StarOffice™ 7 Office Suite

A Sun™ ONE Software Offering

Handbok för Basic-programmering

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
USA 650-960-1300

Artikelnummer 817-3921-10
2003, Revision A

Copyrights and Trademarks

Innehållsförteckning

KAPITEL 1

Introduktion

Den här handboken ger en introduktion till programmering med StarOffice Basic 7.0, och pekar på möjliga sätt att använda StarOffice Basic i StarOffice. För att få ut så mycket som möjligt av boken bör du vara bekant med andra programmeringsspråk.

De många exempel som beskrivs hjälper dig att snabbt utveckla egna StarOffice Basic-program.

Handboken innehåller ett antal migrationstips för Microsoft Visual Basic-programmerare och programmerare som har arbetat med tidigare versioner av StarOffice Basic. Dessa anges av en liten symbol i kanten av sidan. Handbokens appendix innehåller en förteckning över alla migrationstips där du snabbt kan navigera till det tips som du vill läsa.

Om StarOffice Basic

Programmeringsspråket StarOffice Basic är utvecklat speciellt för StarOffice och är väl integrerat med Office-paketet.

Som namnet antyder tillhör programmeringsspråket Basic-familjen. Om du tidigare har arbetat med andra Basic-språk – särskilt Visual Basic och Visual Basic for Applications (VBA) från Microsoft – kommer du snabbt att vänja dig vid StarOffice Basic. Stora delar av grunderna i StarOffice Basic är kompatibla med Visual Basic.

Programmeringsspråket StarOffice Basic kan delas in i fyra komponenter:

- **Språket StarOffice Basic**: Definierar de grundläggande språkliga beståndsdelarna i t.ex. variabeldeklarationer, loopar och funktioner.
- **Runtimebiblioteket**: Innehåller standardfunktioner som inte har någon direkt anknytning till StarOffice, t.ex. funktioner för redigering av tal, strängar, datumvärden och filer.
- **StarOffice API (Application Programming Interface)**: Ger åtkomst till StarOffice-dokument och tillåter att dessa skapas, sparas, ändras och skrivs ut.
- **Dialogruteredigeraren**: Skapar personliga dialogrutor och områden där du kan lägga till kontrollelement och händelsehanterare.

Kompatibilitet mellan StarOffice Basic och VBA gäller både språket StarOffice Basic och runtimebiblioteket. StarOffice API och dialogruteredigeraren är *inte* kompatibla med VBA (standardisering av gränssnitten hade gjort många av funktionerna i StarOffice omöjliga).

Tänkta användare av StarOffice Basic

Användningsområdena för StarOffice Basic börjar där de vanliga funktionerna i StarOffice slutar. Det innebär att rutinaktiviteter kan automatiseras i StarOffice Basic, länkar kan upprättas till andra

program – exempelvis till en databasserver – och komplexa aktiviteter kan utföras med en knapptryckning som startar fördefinierade skript.

StarOffice Basic erbjuder fullständig åtkomst till alla StarOffice-funktioner, stöder alla funktioner, ändrar dokumenttyper och erbjuder alternativ för att skapa personliga dialogrutor.

Använda StarOffice Basic

StarOffice Basic kan användas av alla StarOffice-användare utan några ytterligare program eller hjälpmedel.
Även med standardinstallation innehåller StarOffice Basic alla nödvändiga komponenter för att skapa användardefinierade Basic-makron, inklusive:

- **IDE** (Integrated Development Environment) som innehåller en redigerare som du kan använda för att mata in och testa makron.
- **Tolken** som behövs för att köra StarOffice Basic-makron.
- **Gränssnitten** för olika StarOffice-program som möjliggör direktåtkomst till Office-dokument.

Handbokens struktur

I de första tre kapitlen presenteras StarOffice Basic för läsaren:

- Kapitel 2: Språket StarOffice Basic
- Kapitel 3: Runtimebiblioteket i StarOffice Basic
- Kapitel 4: Introduktion till StarOffice API

De här kapitlen ger en översikt över StarOffice Basic och bör läsas av alla som har för avsikt att skriva StarOffice Basic-program.

De återstående kapitlen beskriver mer detaljerat de enskilda komponenterna i StarOffice API och kan läsas vid behov.

- Kapitel 5: Arbeta med StarOffice-dokument
- Kapitel 6: Textdokument
- Kapitel 7: Tabelldokument
- Kapitel 8: Teckningar och presentationer
- Kapitel 9: Diagram
- Kapitel 10: Databasåtkomst
- Kapitel 11: Dialogrutor
- Kapitel 12: Formulär

Mer information

De komponenter i StarOffice API som diskuteras i den här handboken är utvalda efter sin praktiska användbarhet för StarOffice Basic-programmerare. Som regel diskuteras bara vissa delar av gränssnitten. Du kan få en mer detaljerad beskrivning av API på Internet-adressen:

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

I Developer's Guide beskrivs StarOffice API mer detaljerat än i den här handboken, men den är främst avsedd för Java- och C++-programmerare. Den som redan är bekant med StarOffice Basic-programmering kan hitta ytterligare information i Developer's Guide om StarOffice Basic och StarOffice-programmering. Du kan ladda ned Developer's Guide från Internet-adressen:

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

Programmerare som hellre vill arbeta direkt med Java eller C++ än med StarOffice Basic bör konsultera StarOffice Developer's Guide i stället för den här handboken. StarOffice-programmering med Java eller C++ är betydligt mer komplicerat än programmering med StarOffice Basic.

KAPITEL 2

Språket StarOffice Basic

StarOffice Basic tillhör familjen Basic-språk. Många delar av StarOffice Basic är identiska med Microsoft Visual Basic for Applications och Microsoft Visual Basic. Den som är bekant med dessa språk kan snabbt lära sig använda StarOffice Basic.

Programmerare med erfarenhet av andra språk – t.ex. Java, C++ eller Delphi – kan också lätt lära sig använda StarOffice Basic. StarOffice Basic är ett fullt utvecklat procedurellt programmeringsspråk som inte längre använder elementära kontrollstrukturer i stil med `GoTo` och `GoSub`.

Du kan också utnyttja fördelarna med objektorienterad programmering eftersom ett gränssnitt i StarOffice Basic tillåter användning av externa objektbibliotek. Hela StarOffice API är baserat på sådana gränssnitt som beskrivs mer detaljerat senare.

Det här kapitlet innehåller en översikt över de viktigaste elementen och beståndsdelarna i språket StarOffice Basic, liksom det ramverk i vilket program och bibliotek är orienterade mot StarOffice Basic.

Översikt över ett StarOffice Basic-program

StarOffice Basic är ett tolkspråk. Till skillnad från C++ eller Turbo Pascal skapar inte StarOffice-kompilatorn körbara eller självdekomprimerande filer som kan köras automatiskt. I stället kan du köra ett StarOffice Basic-program genom att trycka på en knapp. Koden genomsöks först efter uppenbara fel och körs sedan rad för rad.

Programrader

Basic-tolkens radorienterade funktionssätt medför en av de viktigaste skillnaderna mellan Basic och andra programmeringsspråk. Medan placeringen av en hård radbrytning i källkoden i exempelvis Java-, C++- eller Delphi-program saknar betydelse, så utgör varje rad i ett Basic-program en självständig enhet. Funktionsanrop, matematiska uttryck och andra språkelement som funktions- och loophuvuden måste avslutas på samma rad som de börjar.

Om det inte finns tillräckligt med utrymme eller om raderna blir väldigt långa kan flera rader länkas samman med understrykningstecken ("_"). Följande exempel visar hur fyra rader i ett matematiskt uttryck kan länkas:

```
LongExpression = (Expression1 * Expression2) + _
      (Expression3 * Expression4) + _
```

```
      (Expression5 * Expression6) + _
      (Expression7 * Expression8)
```

Understrykningstecknet måste alltid vara det sista tecknet i en länkad rad och får inte följas av mellanslag eller tabb, eftersom koden då genererar ett fel.

Förutom att länka ihop enskilda rader så kan StarOffice Basic-programmerare använda kolontecken för att dela upp en rad i flera avsnitt och få plats med flera uttryck. Tilldelningarna

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

kan t.ex. skrivas i följande form:

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

Kommentarer

Förutom den programkod som ska köras kan ett StarOffice Basic-program också innehålla kommentarer som förklarar programmets enskilda beståndsdelar och ger viktig information som kan användas för att analysera programkoden senare, t.ex. i samband med felsökning.

StarOffice Basic erbjuder två metoder för att infoga kommentarer i programkoden:

- Alla tecken som följer efter en apostrof behandlas som kommentarer:

  ```
  Dim A    ' Det här är en kommentar gällande variabel A
  ```

- Nyckelordet `Rem`, följt av kommentaren.

  ```
  Rem Den här kommentaren föregås av nyckelordet Rem.
  ```

En kommentar upptar vanligen alla tecken ända till slutet av raden. StarOffice Basic tolkar sedan den följande raden som en reguljär instruktion igen. Om en kommentar sträcker sig över flera rader måste varje kommentarsrad anges som kommentar.

```
Dim B       ' Den här kommentaren till variabel B är ganska lång
            ' och sträcker sig över flera rader. Kommentarstecknet
            ' måste därför upprepas för varje
            ' rad.
```

Platshållare

Ett StarOffice Basic-program kan innehålla dussintals, hundratals eller t.o.m. tusentals ***platshållare*** (namn på variabler, konstanter, funktioner o.s.v.)
När du bestämmer namn på en platshållare gäller följande:

- Platshållare kan bara innehålla latinska bokstäver, tal och understrykningstecken ("_").
- Det första tecknet i en platshållare måste vara en bokstav eller ett understrykningstecken.

- Platshållare kan inte innehålla specialtecken som t.ex. ä, â, î eller ß.
- Största tillåtna längd på en platshållare är 255 tecken.
- Ingen skillnad görs mellan versaler och gemener (stora och små bokstäver). Platshållaren `OneTestVariable` definierar t.ex. samma variabel som `onetestVariable` och `ONETESTVARIABLE.`

  Den här regeln har dock ett undantag: det görs skillnad mellan versala och gemena tecken för UNO-API-konstanter. Mer information om UNO finns i kapitel 4.

Reglerna för att skapa platshållare är annorlunda i StarOffice Basic jämfört med VBA. Till skillnad från VBA tillåter StarOffice Basic t.ex. inte specialtecken i platshållare, eftersom de kan orsaka problem i internationella projekt.

Här följer några exempel på korrekta och felaktiga platshållare:

```
Efternamn            ' Korrekt
Efternamn5           ' Korrekt (talet 5 är inte första tecknet)
Övriga namn          ' Felaktigt (mellanslag är inte tillåtna)
DéjàVu                     ' Felaktigt (specialtecken som é och à är inte tillåtna)
5Efternamn           ' Felaktigt (första tecknet får inte vara ett tal)
Förnamn,andra        ' Felaktigt (kommatecken och punkt är inte tillåtna)
```

Arbeta med variabler

Implicit variabeldeklaration

Basic-språk är utformade för att vara lättanvända. Ett resultat är att du i StarOffice Basic kan skapa en variabel genom enkel användning och utan en explicit deklaration. Det innebär att en variabel existerar från det ögonblick du tar med den i koden. I följande avsnitt deklareras upp till tre nya variabler, beroende på vilka variabler som redan finns:

```
a = b + c
```

Att deklarera variabler implicit är vanskligt och kan resultera i att en ny variabel introduceras oavsiktligt på grund av t.ex. skrivfel. I stället för att generera ett felmeddelande initierar tolken skrivfelet som en ny variabel med värdet 0. Att hitta fel av den här typen i kod kan vara mycket svårt.

Explicit variabeldeklaration

För att förhindra att fel orsakas av implicit variabeldeklaration innehåller StarOffice Basic en växel som heter:

```
Option Explicit
```

Den måste finnas i den första programraden i varje modul och säkerställer att ett felmeddelande genereras om någon av de använda variablerna inte är deklarerad. Växeln `Option Explicit` bör ingå i alla Basic-moduler.

Kommandot för en explicit variabeldeklaration är i sin enklaste form följande:

```
Dim MyVar
```

Det här exemplet deklarerar en variabel med namnet `MyVar` och typvarianten. En variant är en universell variabel som kan registrera alla upptänkliga värden som t.ex. strängar, hela tal, flytande decimaltal och logiska värden. Här följer några exempel på variabler av typen Variant:

```
MyVar = "Hallå världen"             ' Tilldelning av en sträng
MyVar = 1                           ' Tilldelning av ett heltal
MyVar = 1.0                         ' Tilldelning av ett flytande decimaltal
MyVar = True                        ' Tilldelning av ett logiskt värde
```

De variabler som deklareras här ovan kan dessutom användas för olika variabeltyper i samma program. Även om detta innebär stor flexibilitet så är det bäst att begränsa en variabel till en variabeltyp. När StarOffice Basic påträffar en felaktigt definierad variabeltyp i ett visst sammanhang genereras ett felmeddelande.

Använd följande format när du skapar en typbunden variabeldeklaration:

```
Dim MyVar As Integer    ' Deklaration av en variabel av typen Integer
```

Variabeln deklareras som heltalstyp och kan registrera heltalsvärden. Du kan också använda följande format för att deklarera en variabel av heltalstyp:

```
Dim MyVar%              ' Deklaration av en variabel av typen Integer
```

Instruktionen `Dim` kan registrera flera variabeldeklarationer:

```
Dim MyVar1, MyVar2
```

Om du vill att variablerna ska tilldelas en permanent typ måste du göra en separat tilldelning för varje variabel:

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

Om du inte deklarerar typ för en variabel så tilldelar StarOffice Basic variabeln en varianttyp. I följande variabeldeklaration blir t.ex. `MyVar1` en variant och `MyVar2` blir ett heltal:

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

Följande avsnitt visar de variabeltyper som är tillgängliga i StarOffice Basic, och beskriver hur de kan användas och deklareras.

Strängar

Strängar är tillsammans med tal de viktigaste grundelementen i StarOffice Basic. En sträng består av en serie enskilda tecken i följd. Datorn sparar strängarna internt som en talsekvens där varje tal representerar ett specifikt tecken.

Från en uppsättning ASCII-tecken till Unicode

Teckenuppsättningar matchar tecken i en sträng med motsvarande kod (tal och tecken) i en tabell som beskriver hur datorn ska skriva strängen på bildskärmen eller med en skrivare.

Teckenuppsättningen ASCII

Teckenuppsättningen ASCII består av koder som representerar tal, tecken och specialsymboler med en byte. ASCII-koderna 0 till 127 motsvarar alfabetet och vanliga symboler (som punkt, parentes och kommatecken), samt en del särskilda styrkoder för bildskärm och skrivare. Teckenuppsättningen ASCII används allmänt som standardformat för att överföra textdata mellan datorer.

ASCII innehåller emellertid inte en fullständig uppsättning europeiska specialtecken, som t.ex. â, ä och î, eller andra teckenformat som t.ex. det kyrilliska alfabetet.

Teckenuppsättningen ANSI

Microsoft har baserat sin Windows-produkt på teckenuppsättningen ANSI (American National Standards Institute) som har utökats gradvis med tecken som saknas i teckenuppsättningen ASCII.

Teckenuppsättningar

Teckenuppsättningarna i ISO 8859 har etablerat en länge efterfrågad internationell standard. De första 128 tecknen i teckenuppsättningen ISO motsvarar teckenuppsättningen ASCII. ISO-standarden introducerar nya teckenuppsättningar ***(kodsidor)*** så att fler språk kan visas korrekt. En följd är emellertid att samma teckenvärde kan representera olika tecken på olika språk.

Unicode

Unicode ökar längden på ett tecken till fyra byte och kombinerar olika teckenuppsättningar för att skapa en standard som kan återge så många som möjligt av världens språk. Version 2.0 av Unicode stöds nu av många program – inklusive StarOffice och StarOffice Basic.

Strängvariabler

StarOffice Basic sparar strängar som strängvariabler i Unicode. En strängvariabel kan lagra upp till 65 535 tecken. StarOffice Basic sparar internt det tillhörande Unicode-värdet för varje tecken. Hur mycket arbetsminne som krävs för en strängvariabel beror på strängens längd.

Exempel på deklaration av en strängvariabel:

```
Dim Variabelnamn As String
```

Du kan också skriva deklarationen som:

```
Dim Variabelnamn$
```

När du portar VBA-program bör du se till att inte överskrida den största tillåtna stränglängden i StarOffice Basic (65 535 tecken).

Specifikation av explicita strängar

När du tilldelar en strängvariabel en explicit sträng omger du strängen med citationstecken (").

```
Dim MyString As String
MyString = " Det här är ett test"
```

Du kan dela upp en sträng på två rader genom att lägga till ett plustecken i slutet av den första raden:

```
Dim MyString As String
MyString =    "Den här strängen är så lång att den" + _
              "måste delas upp på två rader."
```

Om du vill infoga ett citationstecken (") i en sträng skriver du det två gånger på den aktuella platsen:

```
Dim MyString As String
MyString = "ett ""-citationstecken."       ' skapar ett "-citationstecken
```

Tal

StarOffice Basic stöder bearbetning av fem grundläggande typer av tal:

- Integer
- Long Integer
- Float
- Double
- Currency

Integer-variabler

Variabler av typen Integer, eller heltalsvariabler, kan lagra alla heltal mellan -32 768 och 32 767. En heltalsvariabel kan kräva upp till två byte minne. Typdeklarationstecknet för en heltalsvariabel är %. Beräkningar som använder heltalsvariabler är mycket snabba och särskilt användbara i loopräknare. Om du tilldelar en heltalsvariabel ett flytande decimaltal avrundas talet uppåt eller nedåt till nästa heltal.

Exempel på deklarationer för heltalsvariabler:

```
Dim Variabelnamn As Integer
Dim Variabelnamn%
```

Long Integer-variabler

Variabler av typen Long Integer, eller långt heltal, kan lagra alla heltal mellan 2 147 483 648 och 2 147 483 647. En lång heltalsvariabel kan kräva upp till fyra byte minne. Typdeklarationstecknet för ett långt heltal är &. Beräkningar som använder långa heltalsvariabler är mycket snabba och särskilt användbara i loopräknare. Om du tilldelar en lång heltalsvariabel ett flytande decimaltal avrundas talet uppåt eller nedåt till nästa heltal.

Exempel på deklarationer för långa heltalsvariabler::

```
Dim Variabelnamn as Long
Dim Variabelnamn&
```

Single-variabler

Variabler av typen Single kan lagra alla positiva och negativa flytande decimaltal mellan 3,402823 x 10^{38} och 1,401298 x 10^{-45}. En variabel av typen Single kan kräva upp till fyra byte minne. Typdeklarationstecknet för en variabel av typen Single är !.

Variabler av typen Single användes ursprungligen för att minska den bearbetningstid som krävs för de mer exakta Double-variablerna. Denna tidsaspekt är dock inte längre relevant, vilket minskar behovet av Single-variabler.

Exempel på deklarationer för variabler av typen Single:

```
Dim Variabelnamn as Single
Dim Variabelnamn!
```

Double-variabler

Variabler av typen Double kan lagra alla positiva och negativa flytande decimaltal mellan 1,79769313486232 x 10^{308} och 4,94065645841247 x 10^{-324}. En variabel av typen Double kan kräva upp till åtta byte minne. Double-variabler används för beräkningar med höga krav på precision. Typdeklarationstecknet är #.

Exempel på deklarationer av Double-variabler:

```
Dim Variabelnamn As Double
Dim Variabelnamn#
```

Currency-variabler

Variabler av typen Currency skiljer sig från andra variabeltyper genom sitt sätt att hantera värden. Decimaltecknet är fast och följs av fyra decimaler. Variabeln kan innehålla upp till 15 siffror före decimaltecknet. En Currency-variabel kan lagra alla värden mellan –922 337 203 685 477,5808 och +922 337 203 685 477,5807 och kräver upp till åtta byte minne. Typdeklarationstecknet för en variabel av typen Currency är @.

Currency-variabler är framför allt avsedda för affärsmässiga beräkningar som kan ge upphov till oväntade avrundningsfel på grund av flytande decimaltal.

Exempel på deklarationer av Currency-variabler:

```
Dim Variabelnamn As Currency
Dim Variabelnamn@
```

Specifikation av explicita tal

Tal kan visas på flera sätt, t.ex. i decimalformat eller i exponentialform, och t.o.m. med en annan bas än decimalsystemet. Följande regler gäller numeriska tecken i StarOffice Basic:

Heltal

Den enklaste metoden är att arbeta med heltal. De visas i källtexten utan tusentalsavgränsare:

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

Talen kan föregås av plustecken (+) eller minustecken (-), med eller utan mellanslag emellan:

```
Dim A As Integer
Dim B As Float

A = +121
B = -243
```

Decimaltal

När du skriver decimaltal använder du punkt (.) som decimaltecken. Den här regeln gör det möjligt att överföra källtext från ett land till ett annat utan konvertering.

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

A = 1223.53       ' avrundas
B = -23446.46     ' avrundas
C = +3532.76323
```

Du kan också använda plustecken (+) och minustecken (-) som prefix för decimaltal, med eller utan mellanslag.

Om en Integer-variabel tilldelas ett decimaltal avrundar StarOffice Basic siffran uppåt eller nedåt.

Exponentiellt skrivsätt

I StarOffice Basic kan tal anges med exponentiellt skrivsätt. Du kan t.ex. skriva 1,5e-10 för talet $1,5 \times 10^{-10}$ (0,00000000015). Bokstaven "e" kan vara stor eller liten och ha eller inte ha ett plustecken (+) som prefix..

Här följer några korrekta och felaktiga exempel på tal skrivna i exponentiellt format:

```
Dim A As Double

A = 1.43E2           ' Korrekt
A = + 1.43E2         ' Korrekt (mellanslag mellan plustecken och tal)
A = - 1.43E2         ' Korrekt (mellanslag mellan minustecken och tal)
A = 1.43E-2          ' Korrekt (negativ exponent)

A = 1.43E -2         ' Felaktigt (mellanslag i talet är inte tillåtna)
A = 1,43E-2          ' Felaktigt (komma är inte tillåtet som decimaltecken)
A = 1.43E2.2         ' Felaktigt (exponenten måste vara ett heltal)
```

Observera att de första och tredje felaktiga exemplen inte genererar något felmeddelande trots att variablerna returnerar felaktiga värden. Uttrycket

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

tolkas som 1,43 minus 2, det vill säga -0,57. Det värde som avsågs var dock $1,43 * 10^2$ (lika med 0,0143). I värdet

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

ignorerar StarOffice Basic den del av exponenten som kommer efter decimaltecknet och tolkar uttrycket som

```
A = 1.43E2
```

Hexadecimala värden

Det hexadecimala systemet (med basen 16) har den fördelen att ett tvåsiffrigt tal motsvarar exakt en byte. Detta möjliggör hantering av tal på ett sätt som ligger närmare datorns arkitektur. I det hexadecimala systemet används talen 0 till 9 och bokstäverna A till F som tal. Ett A står för decimaltalet 10, medan bokstaven F representerar decimaltalet 15. I StarOffice Basic kan du använda heltalsnumrerade hexadecimalvärden, så länge de föregås av &H.

```
Dim A As Longer

A = &HFF      ' Hexadecimalvärdet FF, motsvarar decimalvärdet 255
A = &H10      ' Hexadecimalvärdet 10, motsvarar decimalvärdet 16
```

Oktala värden

StarOffice Basic kan också hantera det oktala systemet (med basen 8) som använder talen 0 till 7, så länge du använder heltal som föregås av &O.

```
Dim A As Longer

A = &O77      ' Oktala värdet 77, motsvarar decimalvärdet 63
A = &O10      ' Oktala värdet 10, motsvarar decimalvärdet 8
```

True och False

Boolean-variabler

Logiska variabler av typen Boolean kan bara innehålla ett av två värden: `True` och `False`. De lämpar sig för binära specifikationer som bara kan anta ett av de namngivna lägena. Ett Boolean-värde sparas internt som ett heltalsvärde på två byte, där 0 motsvarar `False` och alla andra värden motsvarar `True`. Det finns inget typdeklarationstecken för variabler av typen Boolean. Deklarationen kan bara göras med tillägget ***As Boolean***.

Exempel på deklaration av en Boolean-variabel:

```
Dim Variabelnamn As Boolean
```

Datum- och tidsuppgifter

Date-variabler

Variabler av typen Date kan innehålla datum- och tidsvärden. När datumvärden sparas använder StarOffice Basic ett internt format som tillåter jämförelser och matematiska operationer med datum- och tidsvärden. Det finns inget typdeklarationstecken för variabler av typen Date. Deklarationen kan bara göras med tillägget ***As Date***.

Exempel på deklaration av en Date-variabel:

```
Dim Variabelnamn As Date
```

Datafält

Förutom enkla variabler (***skalärer***) stöder StarOffice Basic även datafält (***matriser***). Ett datafält innehåller flera variabler som nås via ett index.

Enkla matriser

En matrisdeklaration är snarlik en enkel variabeldeklaration, men till skillnad från variabeldeklarationen följs matrisens namn av parenteser som innehåller specifikationer av antalet element. Uttrycket

```
Dim MyArray(3)
```

deklarerar en matris som har fyra variabler av datatypen Variant, nämligen `MyArray(0)`, `MyArray(1)`, `MyArray(2)` och `MyArray(3)`.

Du kan också deklarera typspecifika variabler i en matris. Följande rad deklarerar t.ex. en matris med fyra variabler av typen Integer.

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

I de ovanstående exemplen börjar matrisens index alltid med standardstartvärdet noll. Som alternativ kan också ett giltighetsområde med start- och slutvärde anges för datafältsdeklarationen. I följande exempel deklareras ett datafält som har sex heltalsvärden och kan nås med indexen 5 till 10:

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

Indexvärdena behöver inte vara positiva. Följande exempel visar också en korrekt deklaration, men med negativa datafältsbegränsningar.

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

Här deklareras ett heltalsdatafält med 6 värden som kan nås med indexen -10 till -5.

Det finns tre begränsningar som du måste ta hänsyn till när du definierar datafältsindex:

- Minsta möjliga index är -32 768.
- Största möjliga index är 32 767.
- Högsta tillåtna antal element (inom en datafältsdimension) är 16 368.

Ibland gäller andra begränsningsvärden för datafältsindex i VBA. Detsamma gäller även högsta tillåtna antal element per dimension. De värden som gäller där hittar du i dokumentationen till VBA.

Angivet värde för startindex

Startindex för ett datafält börjar vanligen med värdet 0. Du kan ändra startindex för alla datafältsdeklarationer till värdet 1 med hjälp av anropet:

```
Option Base 1
```

Anropet måste finnas i huvudet på en modul om det ska gälla alla matrisdeklarationer i modulen. Anropet påverkar emellertid inte de UNO-sekvenser som är definierade i StarOffice API vars index ***alltid*** börjar med 0. För största tydlighet bör du undvika att använda Option Base 1.

Antalet element i en matris påverkas inte om du använder Option Base 1, utan det är bara startindex som ändras. Deklarationen

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

skapar 4 heltalsvariabler som kan beskrivas med uttrycken `MyInteger(1)`, `MyInteger(2)`, `MyInteger(3)` och `MyInteger(4)`.

I StarOffice Basic påverkar uttrycket Option Base 1 inte antalet element i en matris som det gör i VBA. Det är bara startindex som flyttas i StarOffice Basic. Medan deklarationen `MyInteger(3)` skapar tre heltalsvärden i VBA med indexen 1 till 3, så skapar samma deklaration i StarOffice Basic fyra heltalsvärden med indexen 1 till 4.

Flerdimensionella datafält

Förutom endimensionella datafält stöder StarOffice Basic också flerdimensionella datafält. De olika dimensionerna avgränsas från varandra med kommatecken. Exemplet

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

definierar en heltalsmatris med två dimensioner som vardera har 6 index (kan nås med indexen 0 till 5). Hela matrisen kan registrera sammanlagt 6 × 6 = 36 heltalsvärden.

Även om det går att definiera hundratals dimensioner i StarOffice Basic-matriser så begränsas antalet av hur mycket minne som finns tillgängligt.

Dynamiska ändringar av storleken på datafält

Ovanstående exempel är baserade på datafält med en angiven storlek. Du kan också definiera matriser där datafältens storlek ändras dynamiskt. Du kan t.ex. definiera en matris som innehåller alla ord i en text som börjar med bokstaven A. Eftersom antalet sådana ord är okänt från början måste du kunna ändra fältets gränser efter behov. Det gör du i StarOffice Basic med följande anrop:

```
ReDim MyArray(10)
```

Till skillnad från VBA där du bara kan dimensionera dynamiska matriser med Dim MyArray(), så kan du i StarOffice Basic ändra både statiska och dynamiska matriser med ReDim.

I följande exempel ändras storleken på den ursprungliga matrisen så att den kan registrera 11 eller 21 värden:

```
Dim MyArray(4) As Integer           ' Deklaration med fem element

' ...

ReDim MyArray(10) As Integer        ' Öka till 11 element

' ...

ReDim MyArray(20) As Integer        ' Öka till 21 element
```

När du återställer storleken på en matris kan du använda något av de alternativ som beskrivs i tidigare avsnitt. Ett alternativ är att deklarera flerdimensionella datafält och / eller ange explicita start- och slutvärden. När storleken på ett datafält ändras går alla data ***förlorade***. Om du vill behålla de ursprungliga värdena använder du kommandot `Preserve`:

```
Dim MyArray(10) As Integer                          ' Definierar den ursprungliga
                                                    ' storleken

' ...

ReDim Preserve MyArray(20)   As Integer             ' Ökning av
                                                    ' datafält medan
                                                    ' innehållet bevaras
```

När du använder `Preserve` bör du se till att antalet dimensioner och variabeltypen förblir desamma.

Till skillnad från VBA där bara den övre gränsen av den sista dimensionen i ett datafält kan ändras när du använder `Preserve`, så kan du i StarOffice Basic ändra även andra dimensioner.

Om du använder `ReDim` med `Preserve` så måste du använda samma datatyp som anges i den ursprungliga datafältsdeklarationen.

Område och livslängd för variabler

En variabel i StarOffice Basic har en begränsad livslängd och ett begränsat område från vilket den kan läsas och användas i andra programfragment. Hur länge en variabel finns kvar och varifrån den kan kommas åt beror på dess angivna plats och typ.

Lokala variabler

Variabler som deklareras i en funktion eller procedur kallas lokala variabler:

```
Sub Test
      Dim MyInteger As Integer
```

```
      ' ...
End Sub
```

Lokala variabler är bara giltiga så länge funktionen eller proceduren körs och återställs sedan till noll. När funktionen sedan anropas är inte de värden som tidigare genererades tillgängliga.

För att behålla tidigare värden måste du definiera variablerna som `Static`:

```
Sub Test
      Static MyInteger As Integer

      ' ...

End Sub
```

Till skillnad från VBA tillåter inte StarOffice Basic att namnet på en lokal variabel används samtidigt både som global och privat variabel i modulhuvudet. När du importerar ett VBA-program till StarOffice Basic måste du ändra alla dubblerade variabelnamn.

Allmänna domänvariabler

Allmänna domänvariabler definieras i huvudet på en modul med nyckelordet `Dim`. De här variablerna är tillgängliga för alla moduler i deras bibliotek:

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
```

Värdet på variabeln `A` ändras inte av `Test`-funktionen, men ökar med ett i `Flip`-funktionen och minskar med ett i `Flop`-funktionen. Båda dessa ändringar av variabeln är globala.

Du kan också använda nyckelordet `Public` i stället för `Dim` när du deklarerar en allmän domänvariabel:

```
Public A As Integer
```

En variabel i en allmän domän är bara tillgänglig så länge det tillhörande makrot körs, varefter variabeln återställs.

Globala variabler

Till funktionen är globala variabler snarlika variabler i allmänna domäner, men deras värden behålls även efter att det tillhörande makrot har körts. Globala variabler deklareras i huvudet på en modul med nyckelordet `Global`:

```
Global A As Integer
```

Privata variabler

`Private`-variabler är bara tillgängliga i den modul där de definieras. Använd nyckelordet `Private` för att definiera variabeln:

```
Private MyInteger As Integer
```

Om flera moduler innehåller en `Private`-variabel med samma namn skapar StarOffice Basic en särskild variabel för varje förekomst av namnet. I följande exempel har både modulen `A` och `B` en `Private`-variabel som heter `C`. Funktionen `Test` ställer först in `Private`-variabeln i modul `A` och sedan `Private`-variabeln i modul `B`.

Modul A:

```
Private C As Integer

Sub Test
      SetModuleA            ' Bestämmer variabel C från modul A
      SetModuleB            ' Bestämmer variabel C från modul B

      ShowVarA              ' Visar variabel C från modul A (= 10)
      ShowVarB              ' Visar variabel C från modul B (= 20)
End Sub

Sub SetmoduleA
      A = 10
End Sub

Sub ShowVarA
      MsgBox C              ' Visar variabel C från modul A
End Sub
```

Modul B:

```
Private C As Integer

Sub SetModuleB
      A = 20
End Sub

Sub ShowVarB
      MsgBox C              ' Visar variabel C från modul B
End Sub
```

Konstanter

I StarOffice Basic använder du nyckelordet `Const` för att deklarera en konstant.

```
Const A = 10
```

Om du vill kan du också ange konstanttypen i deklarationen:

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

Operatorer

StarOffice Basic förstår vanliga matematiska operatorer, logiska operatorer och jämförelseoperatorer.

Matematiska operatorer

Matematiska operatorer kan användas med alla typer av tal, och operatorn + kan också användas för att länka strängar.

`+` Addera tal och datumvärden, länka strängar

`-` Subtrahera tal och datumvärden

`*` Multiplicera tal

`/` Dividera tal

`\` Dividera tal med heltal som resultat (avrundat)

`^` Upphöja ett tal till en potens

`REST` moduloperation (beräkna resten av en division)

Logiska operatorer

Med logiska operatorer kan du länka element enligt reglerna för Booleansk algebra. Om operatorerna används på logiska värden ger länken det sökta resultatet direkt. Om de används i samband med heltalsvärden och långa heltalsvärden så görs länkningen på bitnivå.

`OCH` Och-länkning

`ELLER` Eller-länkning

`EXKLUSIVT ELLER` Exklusivt-eller-länkning

`INTE` Negering

`EQV` Likhetstest (båda delarna `True` eller `False`)

`IMP` Implikation (om det första uttrycket är sant måste också det andra uttrycket vara sant)

Jämförelseoperatorer

Jämförelseoperatorer kan användas på alla grundläggande variabeltyper (tal, datumuppgifter, strängar och logiska värden).

=	Likhet mellan tal, datumvärden och strängar
<>	Skillnader mellan tal, datumvärden och strängar
>	Större-än-kontroll av tal, datumvärden och strängar
>=	Större-än-eller-lika-med-kontroll av tal, datumvärden och strängar
<	Mindre-än-kontroll av tal, datumvärden och strängar
<=	Mindre-än-eller-lika-med-kontroll av tal, datumvärden och strängar

Jämförelseoperatorn `Like` i VBA stöds inte i StarOffice Basic.

Förgrening

Använd förgreningsuttryck för att stoppa körningen av ett kodblock tills ett visst villkor är uppfyllt.

If...Then...Else

Det vanligaste förgreningsuttrycket är `If`-uttrycket som demonstreras i följande exempel:

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

Tilldelningen `B = 2` inträffar bara när värdet på variabeln `A` är större än tre. En variant av `If`-uttrycket är `If/Else`-uttrycket:

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

I det här exemplet tilldelas variabeln `B` värdet 2 om `A` är större än 3, annars tilldelas `B` värdet 0.

För mer komplexa uttryck kan du göra en kaskad av `If`-uttrycket, som i följande exempel:

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

Om värdet på variabeln `A` är lika med noll tilldelas `B` värdet `0`. Om `A` är mindre än `3` (men inte lika med noll) så tilldelas `B` värdet `1`. I samtliga andra fall (d.v.s. om `A` är större än eller lika med `3`), så tilldelas `B` värdet `2`.

Select...Case

Instruktionen `Select...Case` är ett alternativ till att göra en kaskad av `If`-uttrycket och används när du vill kontrollera ett värde mot olika villkor.

```
Select Case DayOfWeek
Case 1:
      NameOfWeekday = "Söndag"
Case 2:
      NameOfWeekday = "Måndag"
Case 3:
      NameOfWeekday = "Tisdag"
Case 4:
      NameOfWeekday = "Onsdag"
Case 5:
      NameOfWeekday = "Torsdag"
Case 6:
      NameOfWeekday = "Fredag"
Case 7:
      NameOfWeekday = "Lördag"
End Select
```

I det här exemplet motsvarar veckodagens namn ett tal, så att `DayOfWeek`-variabeln tilldelas värdet 1 för `söndag`, `2 för måndag` och så vidare.

Kommandot `Select` är inte begränsat till enkla 1:1-tilldelningar. Du kan också ange jämförelseoperatorer eller listor med uttryck i en `Case`-förgrening. Följande exempel visar de viktigaste syntaxvarianterna:

```
Select Case Var
Case 1 To 5

      ' ... Var ligger mellan talen 1 och 5

Case 6, 7, 8

      ' ... Var är 6, 7 eller 8

Case Var > 8 And Var < 11

      ' ... Var är större än 8 och mindre än 11

Case Else

      ' ... alla övriga fall

End Select
```

Loopar

En loop kör ett kodblock det antal omgångar som anges. Loopar kan också köras ett odefinierat antal omgångar.

For...Next

Loopen `For...Next` har ett fast antal omgångar. Loopräknaren definierar hur många gånger loopen ska köras. I följande exempel,

```
Dim I

For I = 1 To 10

      ' ...  Inre delen av loopen

Next I
```

är variabeln `I` loopräknare, med initialvärdet 1. Räknaren använder inkrementet 1 i slutet av varje omgång. När variabel I är lika med 10 stannar loopen.

Om du vill att loopräknaren ska använda ett annat inkrement än 1 i slutet av varje omgång använder du funktionen `Step`:

```
Dim I

For I = 1 To 10 Step 0.5

      ' ... Inre delen av loopen

Next I
```

I ovanstående exempel ökar räknaren med 0,5 i slutet av varje omgång och loopen körs 19 gånger.

Du kan också använda negativa stegvärden:

```
Dim I

For I = 10 To 1 Step -1

      ' ... Inre delen av loopen

Next I
```

I det här exemplet börjar räknaren på 10 och minskas med 1 i slutet av varje omgång tills räknaren når värdet 1.

Med instruktionen `Exit For` kan du avsluta en `For`-loop i förväg. I följande exempel avbryts loopen under den femte omgången:

```
Dim I

For I = 1 To 10

      If I = 5 Then
            Exit For
      End If

      ' ... Inre delen av loopen

Next I
```

Loopvarianten `For Each...Next` i VBA stöds inte i StarOffice Basic.

Do...Loop

Loopvarianten `Do...Loop` är inte kopplad till ett fast antal omgångar. I stället körs `Do...Loop`-loopen tills ett visst villkor är uppfyllt. Det finns fyra varianter av `Do...Loop` (i följande exempel representerar `A > 10` valfritt villkor):

1. Varianten `Do While...Loop`-loop

   ```
   Do While A > 10
       ' ... loopens text
   Loop
   ```

   kontrollerar om villkoren fortfarande är uppfyllda före varje omgång och kör bara loopen om så är fallet.

2. Varianten `Do Until...Loop`-loop

   ```
   Do Until A > 10
         ' ... loopens text
   Loop
   ```

   kör loopen tills villkoret *inte längre* är uppfyllt.

3. Varianten `Do...Loop While`-loop

   ```
   Do
         ' ... loopens text
   Loop While A > 10
   ```

   kontrollerar bara villkoret efter den första loopomgången och avbryter om villkoret är ***uppfyllt***.

4. Varianten `Do...Loop Until`-loop

   ```
   Do
         ' ... loopens text
   Loop Until A > 10
   ```

   kontrollerar också villkoret efter den första omgången, men kör loopen tills villkoret ***inte längre*** är uppfyllt.

Precis som loopen `For...Next` så innehåller också `Do...Loop` ett kommando för att avsluta. Kommandot `Exit Do` kan avsluta en loop när som helst.

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

      ' ... loopens text
While A > 10
```

Programmeringsexempel: Sortera med inbäddade loopar

Det finns många sätt att använda loopar, t.ex. för att söka igenom listor, returnera värden eller köra komplexa matematiska aktiviteter. Följande exempel är en algoritm som använder loopar för att sortera en lista efter namn.

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

      Entry(1) = "Pernilla"
      Entry(2) = "Kurt"
      Entry(3) = "Thomas"
      Entry(4) = "Mikael"
      Entry(5) = "David"
      Entry(6) = "Kattis"
      Entry(7) = "Sussi"
      Entry(8) = "Edvard"
      Entry(9) = "Christina"
      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
```

Värdena byts ut parvis flera gånger innan de slutligen är sorterade i stigande ordning. Som bubblor flyttas variablerna gradvis till rätt position. Av det skälet kallas algoritmen också en ***bubbelsortering***.

Procedurer och funktioner

Procedurer och funktioner är centrala delar av strukturen i ett program. De utgör ett ramverk när du delar upp komplexa problem i flera underordnade aktiviteter.

Procedurer

En ***procedur*** kör en aktivitet utan att det resulterar i ett explicit värde. Dess syntax är

```
Sub Test

      ' ... här följer själva procedurens kod

End Sub
```

I exemplet definieras en procedur som kallas `Test` vars kod går att komma åt var som helst i programmet. Anropet görs genom att procedurens namn anges på den aktuella platsen i programmet.

```
Test
```

Funktioner

En ***funktion*** kombinerar ett block av program som ska köras som en logisk enhet, precis som en procedur. Till skillnad från proceduren genererar dock funktionen ett returvärde.

```
Function Test

      ' ... här följer själva funktionens kod

      Test = 123
End Function
```

Returvärdet tilldelas med enkel tilldelning. Tilldelningen behöver inte placeras i slutet, utan kan göras var som helst i funktionen.

Ovanstående funktion kan anropas inuti ett program enligt följande:

```
Dim A

A = Test
```

Koden definierar en variabel `A` som resultatet av funktionen `Test` tilldelas.

Returvärdet kan skrivas över flera gånger inuti funktionen. Precis som vid klassisk variabeltilldelning returnerar funktionen i det här exemplet det värde den senast blev tilldelad.

```
Function Test

      Test = 12

      ' ...

      Test = 123

End Function
```

I det här exemplet är funktionens returvärde `123`.

Om en tilldelning stoppas returnerar funktionen värdet `noll` (talet 0 för numeriska värden och tomt för strängar).

En funktions returvärde kan vara av vilken typ som helst. Typen deklareras likadant som i en variabeldeklaration:

```
Function Test As Integer

      ' ... här följer själva funktionens kod

End Function
```

Om specifikationen av ett explicit värde stoppas så tilldelas returvärdets typ som variant.

Avbryta procedurer och funktioner i förväg

I StarOffice Basic kan du använda kommandona `Exit Sub` och `Exit Function` om du vill avbryta en procedur eller funktion i förväg, t.ex. för felhantering. Dessa kommandon stoppar proceduren eller funktionen och återställer programmet till den punkt där proceduren och/eller funktionen startades.

Följande exempel visar en procedur som avbryter implementeringen när variabeln `ErrorOccured` har värdet `True`.

```
Sub Test
      Dim ErrorOccured As Boolean

      ' ...

      If ErrorOccured Then
            Exit Sub
      End If

      ' ...

End Sub
```

Skicka parametrar

Funktioner och procedurer kan ta emot en eller flera parametrar. Viktiga parametrar måste omges av parenteser efter funktionens eller procedurens namn. Exemplet

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

definierar en procedur som förväntar ett heltalsvärde A och en sträng B som parametrar.

Parametrar skickas normalt med ***referens*** i StarOffice Basic. Ändringar i variablerna behålls när proceduren eller funktionen avslutas:

```
Sub Test
      Dim A As Integer
      A = 10
      ChangeValue(A)
      ' Parametern A har nu värdet 20
End Sub
Sub ChangeValue(TheValue As Integer)
       TheValue = 20
End Sub
```

I det här exemplet skickas variabeln A som definieras i funktionen `Test` som en parameter till funktionen `ChangeValue`. Värdet ändras sedan till 20 och skickas till `TheValue`, som behålls när funktionen avslutas.

Du kan också skicka en parameter som ett ***värde*** om du inte vill att efterföljande ändringar av parametern ska påverka det ursprungligen skickade värdet. När du anger att en parameter ska överföras som ett värde bör du se till att nyckelordet `ByVal` föregår variabeldeklarationen i funktionshuvudet.

Om vi i ovanstående exempel ersätter funktionen `ChangeValue` med funktionen

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

så förblir den överordnade variabeln A opåverkad av ändringen. Efter anropet till funktionen `ChangeValue` behåller variabel A värdet `10`.

Metoden för att skicka parametrar till procedurer och funktioner i StarOffice Basic är nästan identisk med den i VBA. Parametrarna skickas som standard med referens. Om du vill skicka parametrar som värden använder du nyckelordet `ByVal` . I VBA kan du också använda nyckelordet `ByRef` för att bestämma att en parameter ska skickas med referens. StarOffice Basic stöder inte det här nyckelordet eftersom det redan är standardprocedur i StarOffice Basic.

Funktioner och procedurer i StarOffice Basic tillhör som regel kategorin `Public`. Nyckelorden `Public` och `Private` som används i VBA stöds inte i StarOffice Basic.

Valfria parametrar

Funktioner och procedurer kan bara anropas om alla nödvändiga parametrar skickas under anropet.

I StarOffice Basic kan du definiera parametrar som ***valfria***, vilket innebär att om motsvarande värden inte ingår i ett anrop så överför StarOffice Basic en tom parameter. I exemplet

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

End Sub
```

är parametern A obligatorisk, medan parametern B är valfri.

Funktionen `IsMissing` kontrollerar om en parameter är skickad eller har utelämnats.

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

      ' Kontrollera om parameter B finns med
      If Not IsMissing (B) Then
             B_Local = B           ' Parameter B finns med
      Else
             B_Local = 0           ' Parameter B saknas -> standardvärde 0
      End If

      ' ... Starta själva funktionen

End Sub
```

Exemplet testar först om parametern B har skickats, och skickar om nödvändigt samma parameter till den interna variabeln `B_Local`. Om motsvarande parameter inte finns med skickas ett standardvärde (i det här fallet värdet `0`) till `B_ Local` i stället för den skickade parametern.

Alternativet i VBA för att definiera standardvärden för valfria parametrar stöds inte i StarOffice Basic.

Nyckelordet `ParamArray` i VBA stöds inte i StarOffice Basic.

Rekursion

Rekursion är nu tillgänglig i StarOffice Basic. En rekursiv procedur eller funktion har förmågan att anropa sig själv tills ett grundläggande villkor har uppfyllts. När funktionen anropas med det grundläggande villkoret returneras ett resultat.

I följande exempel används en rekursiv funktion för att beräkna fakulteten av talen `42`, `-42` och `3,14`:

```
Sub Main
   Msgbox CalculateFactorial(  42 )    ' Visar 1,40500611775288E+51
   Msgbox CalculateFactorial( -42 )    ' Visar "Ogiltigt tal för fakultet!"
   Msgbox CalculateFactorial( 3.14 )   ' Visar "Ogiltigt tal för fakultet!"
End Sub

Function CalculateFactorial( Number )
   If Number < 0 Or Number <> Int( Number ) Then
       CalculateFactorial = "Ogiltigt tal för fakultet!"
   ElseIf Number = 0 Then
       CalculateFactorial = 1
   Else
       ' Det här är det rekursiva anropet:
       CalculateFactorial = Number * CalculateFactorial( Number - 1 )
   Endif
End Function
```

Exemplet returnerar fakulteten för talet `42` genom att rekursivt anropa funktionen `CalculateFactorial` tills den når det grundläggande villkoret `0! = 1`.

Observera att rekursionsnivån i StarOffice Basic för närvarande är begränsad till 500.

Felhantering

Korrekt felhantering är ett av de mest tidskrävande inslagen i programmering. StarOffice Basic innehåller en uppsättning verktyg som förenklar felhantering.

Instruktionen On Error

Instruktionen `On Error` är nyckeln till all felhantering:

```
Sub Test
      On Error Goto ErrorHandler

      ' ... utför en aktivitet under vilken ett fel kan uppstå

Exit Sub

ErrorHandler:

      ' ... individuell kod för felhantering
```

```
End Sub
```

Raden `On Error Goto ErrorHandler` definierar hur StarOffice Basic ska agera om ett fel uppstår. Delen `Goto ErrorHandler` ser till att StarOffice Basic avslutar den aktuella programraden och sedan kör `ErrorHandler`-koden.

Kommandot Resume

Kommandot `Resume Next` gör att programmet fortsätter från raden efter den där felet inträffade, när koden i felhanteraren har körts:

```
ErrorHandler:

      ' ... individuell kod för felhantering

      Resume Next
```

Använd kommandot `Resume Proceed` för att ange en punkt där programmet ska fortsätta efter felhantering:

```
ErrorHandler:

      ' ... individuell kod för felhantering
      Resume Proceed

Proceed:

      ' ... programmet fortsätter här efter felet
```

Om du vill fortsätta med ett program utan att ett felmeddelande visas när fel inträffar använder du följande format:

```
Sub Test
      On Error Resume Next

      ' ... utför en aktivitet under vilken ett fel kan uppstå

End Sub
```

Använd kommandot `On Error Resume Next` med försiktighet eftersom dess effekt är global. Mer information finns i ***Tips för strukturerad felhantering***.

Frågor om felinformation

Vid felhantering är det praktiskt att ha tillgång till en beskrivning av felet och veta var och varför felet uppstod.

- Variabeln `Err` innehåller det antal fel som har inträffat.
- Variabeln `Error$` innehåller en beskrivning av felet.

- Variabeln `Erl` innehåller numret på den rad där felet inträffade.

Anropet

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

visar hur felinformation kan visas i ett meddelandefönster.

Medan VBA sammanfattar felmeddelandena i ett statistiskt objekt som kallas `Err`, så innehåller StarOffice Basic variablerna `Err`, `Error$` och `Erl`.

Statusinformationen är giltig tills programmet påträffar kommandot `Resume` eller `On Error`, varvid informationen återställs.

I VBA återställer metoden `Err.Clear` i objektet `Err` felstatusen när ett fel har inträffat. Detta utförs i StarOffice Basic med kommandona `On Error` eller `Resume`.

Tips för strukturerad felhantering

Både definitionskommandot, `On Error`, och returkommandot `Resume`, är varianter av begreppet `Goto`.

Om du vill strukturera koden snyggt och förhindra att fel uppstår när du använder det här begreppet, bör du inte använda hoppkommandon utan att granska dem.

Var försiktig när du använder kommandot `On Error Resume Next` eftersom det ignorerar alla öppna felmeddelanden.

Den bästa lösningen är att bara använda en typ av felhantering i ett och samma program, hålla isär felhanteringen från själva programkoden och inte hoppa tillbaka till ursprungskoden efter att ett fel inträffar.

Här följer ett exempel på en felhanteringsprocedur:

```
Sub Example

      ' Definiera felhanterare i början av funktionen
      On Error Goto ErrorHandler

      ' ... Här skrivs själva programkoden

      ' Inaktivera felhantering
      On Error Goto 0

      ' Slut på reguljär programimplementering
Exit Sub

' Startpunkt för felhantering
ErrorHandler:

      ' Kontrollera om felet var väntat
      If Err = ExpectedErrorNo Then
```

```
            ' ... Processfel
      Else
            ' ... Varning för oväntat fel
      End If

      On Error Goto 0                              ' Inaktivera felhantering
End Sub
```

Den här proceduren börjar med definitionen av en felhanterare, följd av själva programkoden. I slutet av programkoden inaktiveras felhanteringen av anropet `On Error Goto 0` och procedurimplementeringen avslutas med kommandot `Exit Sub` (ej att förväxla med `End Sub`).

I exemplet kontrolleras först om antalet fel stämmer med det förväntade antalet (som är lagrat i den imaginära konstanten `ExpectedErrorNo`), varefter felet hanteras. Om ännu ett fel inträffar genererar systemet en varning. Det är viktigt att kontrollera antalet fel så att oväntade fel kan upptäckas.

Anropet `On Error Goto 0` i slutet av koden återställer statusinformationen för felet (felkoden i systemvariablerna `Err`) så att eventuella fel som inträffar senare tydligt kan identifieras.

KAPITEL 3

Runtimebiblioteket i StarOffice Basic

Följande avsnitt presenterar de centrala funktionerna i runtimebiblioteket.

Konverteringsfunktioner

Det händer ofta att en variabel av en typ måste omvandlas till en variabel av en annan typ.

Implicit och explicit typkonvertering

Det enklaste sättet att ändra en variabel från en typ till en annan är att använda en tilldelning.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

I det här exemplet är variabeln A en sträng och variabeln B är ett heltal. StarOffice Basic säkerställer att variabel B konverteras till en sträng vid tilldelning till variabel A. Den här konverteringen är mycket mer avancerad än den kan verka: heltalet B blir kvar i arbetsminnet i form av ett tal på två byte. A däremot är en sträng, och datorn sparar ett värde på en eller två byte för varje tecken (varje tal). Därför måste, innan innehållet kopieras från B till A, B konverteras till det interna formatet för A.

Till skillnad från de flesta programmeringsspråk utför Basic typkonvertering automatiskt. Detta kan dock få ödesdigra konsekvenser. Vid närmare granskning visar sig följande kodsekvens

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

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

som vid ett första påseende kan verka oproblematisk, i själva verket vara något av en fälla. Basic-tolken beräknar först resultatet av additionsprocessen och konverterar det sedan till en sträng vilken som resultat producerar strängen 2.
Om å andra sidan Basic-tolken först konverterar startvärdena B och C till en sträng och använder plusoperatorn på resultatet, så produceras strängen 11.

Detsamma gäller vid användning av variantvariabler:

```
Dim A
Dim B
Dim C

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

Eftersom variantvariabler kan innehålla både tal och strängar är det oklart om variabel A är tilldelad talet 2 eller strängen 11.

De nämnda felkällorna för implicit typkonvertering kan bara undvikas med disciplinerad programmering som t.ex. inte använder variantdatatypen (vilket rekommenderas).

För att undvika andra fel som orsakas av implicit typkonvertering erbjuder StarOffice Basic en uppsättning konverteringsfunktioner som du kan använda för att definiera när datatypen i en operation ska konverteras:

- **CStr(Var)** – konverterar alla datatyper till en sträng.
- **CInt(Var)** – konverterar alla datatyper till ett heltalsvärde.
- **CLng(Var)** – konverterar alla datatyper till ett långt värde.
- **CSng(Var)** – konverterar alla datatyper till ett enskilt värde.
- **CDbl(Var)** – konverterar alla datatyper till ett dubbelvärde.
- **CBool(Var)** – konverterar alla datatyper till ett logiskt värde.
- **CDate(Var)** – konverterar alla datatyper till ett datumvärde.

Du kan använda konverteringsfunktionerna för att definiera hur StarOffice Basic ska utföra följande typkonverteringsoperationer:

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

B = 1
C = 1:

A = CInt(B + C)             ' B och C adderas först, konverteras sedan
                              (producerar talet 2)
A = CStr(B) + Cstr(C)       ' B och C konverteras först till en sträng, och
                            ' kombineras sedan (producerar strängen "11")
```

Under den första additionen i exemplet adderar StarOffice Basic först heltalsvariablerna och konverterar sedan resultatet till en kedja med tecken. A tilldelas strängen 2. I det andra fallet konverteras heltalsvariablerna först till två strängar och länkas sedan till varandra via tilldelningen. A tilldelas därför strängen 11.

De numeriska konverteringsfunktionerna `CSng` och `CDbl` accepterar också decimaltal. Den symbol som definieras i motsvarande landsspecifika inställningar måste användas som decimaltecken. På samma sätt använder metoderna `CStr` de aktuella landsspecifika inställningarna vid formatering av tal, datum- och tidsuppgifter.

Funktionen `Val` skiljer sig från metoderna `Csng, Cdbl` och `Cstr`. Den konverterar en sträng till ett tal, men förväntar alltid att en punkt ska användas som decimaltecken.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)        '  Konverteras korrekt oavsett landsspecifika inställningar
```

Kontrollera innehållet i variabler

I vissa fall kan inte datum konverteras:

```
Dim A As String
Dim B As Date

A = "test"
B = A                   ' Skapar felmeddelande
```

I det visade exemplet är tilldelningen av strängen `test` i en datumvariabel meningslös, så Basic-tolken rapporterar ett fel. Detsamma gäller om du försöker tilldela en Booleansk variabel en sträng:

```
Dim A As String
Dim B As Boolean

A = "test"
B = A                   ' Skapar felmeddelande
```

Basic-tolken rapporterar återigen ett fel.

De här felmeddelandena går att undvika om du kontrollerar programmet före en tilldelning och tar reda på om innehållet i den variabel som ska tilldelas matchar målvariabelns typ. StarOffice Basic tillhandahåller följande testfunktioner som löser problemet:

- **`IsNumeric(Value)`** – kontrollerar om ett värde är ett tal.
- **`IsDate(Value)`** – kontrollerar om ett värde är ett datum
- **`IsArray(Value)`** – kontrollerar om ett värde är en matris

De här funktionerna är särskilt användbara vid kontroll av inmatade användaruppgifter. Du kan t.ex. kontrollera om en användare har angett ett giltigt tal eller datum.

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

Om variabeln `UserInput` i exemplet ovan innehåller ett giltigt numeriskt värde tilldelas det variabeln `ValidInput`. Om inte `UserInput` innehåller ett giltigt tal tilldelas `ValidInput` värdet `0` och ett felmeddelande returneras.

Det finns testfunktioner för att kontrollera tal, datuminformation och matriser i Basic, men ingen motsvarande funktion för att kontrollera logiska värden. Funktionaliteten kan emellertid åstadkommas med funktionen `IsBoolean`:

```
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
```

Funktionen `IsBoolean` definierar en intern `Dummy`-hjälpvariabel av Booleansk typ, och försöker tilldela det skickade värdet denna. Om tilldelningen lyckas returnerar funktionen `True`. Om den misslyckas genereras ett runtime-fel som avbryter testfunktionen och returnerar ett fel.

Om en sträng i StarOffice Basic innehåller ett ickenumeriskt värde och tilldelas ett tal, så skapar StarOffice Basic inget felmeddelande utan skickar värdet `0` till variabeln. Den här proceduren skiljer sig från VBA. Där utlöses ett fel och programimplementeringen avbryts om en sådan tilldelning görs.

Strängar

Arbeta med teckenuppsättningar

När StarOffice Basic hanterar strängar används teckenuppsättningen Unicode. Med funktionerna `Asc` och `Chr` går det att fastställa det Unicode-värde som tillhör ett tecken, och/eller hitta motsvarande tecken för ett Unicode-värde. Följande uttryck tilldelar variabeln `Code` olika Unicode-värden:

```
Code = Asc("A")             ' Latinska bokstaven A (Unicode-värde 65)
Code = Asc("€")             ' Eurotecken (Unicode-värde 8364)
Code = Asc("л")             ' Kyrillisk bokstav л (Unicode-värde 1083)
```

I motsatt riktning säkerställer uttrycket

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

att strängen `MyString` initieras med värdet `13`, som står för en hård radbrytning.

Kommandot `Chr` används ofta i Basic-språken för att infoga kontrolltecken i en sträng. Tilldelningen

```
MyString = Chr(9) + "Det här är ett test" + Chr(13)
```

säkerställer därför att texten föregås av ett tabbtecken (Unicode-värde `9`) och att en hård sidbrytning (Unicode-värde `13`) läggs till efter texten.

Komma åt delar av en sträng

StarOffice Basic innehåller fyra funktioner som returnerar delar av strängar:

- **`Left(MyString, Length)`** – returnerar de första `Length` tecknen i `MyString`.
- **`Right(MyString, Length)`** – returnerar de sista `Length` tecknen i `MyString`.
- **`Mid(MyString, Start, Length)`** – returnerar de första `Length` tecknen i `MyString` med början vid `Start`-positionen.
- **`Len(MyString)`** – returnerar antalet tecken i `MyString`.

Här följer några exempel på anrop av namngivna funktioner:

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

MyString = "Det här är ett litet test"

MyResult = Left(MyString,5)         ' Ger strängen "Det h"
MyResult = Right(MyString, 5)       ' Ger strängen " test"
MyResult = Mid(MyString, 8, 5)      ' Ger strängen "är et"
MyLength = Len(MyString)            ' Ger värdet 4
```

Sök och ersätt

StarOffice Basic innehåller funktionen `InStr` som söker efter en del av en sträng inuti en sträng:

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

Parametern `SearchString` anger vilken sträng som ska sökas inuti strängen `MyString`. Funktionen returnerar ett tal som innehåller den position där `SearchString` först förekommer inuti `MyString`. Om du vill leta efter andra matchningar för strängen erbjuder funktionen också möjlighet att ange en valfri startposition där StarOffice Basic börjar sökningen. Funktionens syntax är i så fall:

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

I ovanstående exempel gör `InStr` ingen skillnad mellan stora och små bokstäver. Du kan ändra sökningen så att `InStr` blir versalkänslig genom att lägga till parametern `0`, enligt följande exempel:

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

Med hjälp av ovanstående funktioner för redigering av strängar kan programmerare söka och ersätta en sträng i en annan sträng:

```
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, Source, Search)
            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
```

Funktionen söker igenom den överförda `Search`-strängen i en loop med hjälp av `InStr` i termens ursprungliga `Source`. Om den påträffar sökordet tar den delen före uttrycket och skriver den till

returbufferten `Result`. Den lägger till det nya avsnittet `Part` på platsen för sökordet `Search`. Om inga fler matchningar påträffas för sökordet bestämmer funktionen den återstående delen av strängen och lägger till den i returbufferten. Den sträng som skapas på det här sättet returneras som ersättningsprocessens resultat.

Eftersom ersättning av delar av en teckenföljd tillhör de mest använda funktionerna har funktionen `Mid` i StarOffice Basic utökats så att aktiviteten utförs automatiskt. I följande exempel

```
Dim MyString As String

MyString  = "Detta var min text"
Mid(MyString, 7, 3, "är")
```

ersätts tre tecken med strängen `är` från den sjunde positionen i strängen `MyString`.

Formatera strängar

Funktionen `Format` formaterar tal som en sträng. För att göra detta förväntar funktionen att ett `Format`-uttryck anges, som sedan används som mall vid formatering av tal. Varje platshållare i mallen säkerställer att objektet formateras på motsvarande sätt i det utmatade värdet. De fem viktigaste platshållarna i en mall är ***noll*** (0), ***fyrkant*** (#), ***punkt*** (.), ***komma*** (,) och ***dollartecken*** ($).

Tecknet ***noll*** i mallen säkerställer att ett tal alltid placeras vid motsvarande punkt. Om inget tal anges visas 0 på dess plats.

En ***punkt*** står för det decimaltecken som är definierat i operativsystemets landsspecifika inställningar.

Exemplet nedan visar hur tecknen ***noll*** och ***punkt*** kan definiera de siffror som följer efter decimaltecknet i ett uttryck:

```
MyFormat = "0.00"

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

På samma sätt kan nollor läggas till framför ett tal för att uppnå önskad längd.

```
MyFormat = "0000.00"

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

Ett ***komma*** representerar det tecken som operativsystemet använder som tusentalsavgränsare, och ***fyrkant*** står för en siffra eller plats som bara visas om det krävs av inmatningssträngen.

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

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

På platsen för ***dollartecken***-platshållaren visar funktionen `Format` den valutasymbol som är definierad av systemet:

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

MyString = Format(-1579.8, MyFormat)      ' Ger "-1,579,80 €"
MyString = Format(1579.8, MyFormat)       ' Ger "1,579,80 €"
MyString = Format(0.4, MyFormat)          ' Ger "0,40 €"
MyString = Format(0.434, MyFormat)        ' Ger "0,43 €"
```

De formatinstruktioner som används i VBA för att formatera datum- och tidsuppgifter stöds inte i StarOffice Basic.

Datum och tid

StarOffice Basic innehåller datatypen `Date` som sparar datum- och tidsuppgifter i binärt format.

Ange datum- och tidsinformation i programkod

Du kan tilldela en datumvariabel ett datum genom tilldelning av en enkel sträng:

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

Den här tilldelningen fungerar korrekt eftersom StarOffice Basic automatiskt konverterar ett datumvärde som är definierat som en sträng till en datumvariabel. Det är emellertid en typ av tilldelning som kan orsaka fel, eftersom datum- och tidsvärden definieras och visas olika i olika länder.

Eftersom StarOffice Basic använder operativsystemets landsspecifika inställningar när en sträng konverteras till ett datumvärde, så fungerar ovanstående uttryck korrekt endast om de landsspecifika inställningarna matchar stränguttrycket.

För att undvika det här problemet bör funktionen `DateSerial` användas för att tilldela en datumvariabel ett fast värde:

```
Dim MyVar As Date

MyDate = DateSerial (2001, 1, 1)
```

Funktionsparametern måste ha ordningen år, månad, dag. Funktionen säkerställer att variabeln tilldelas korrekt värde oavsett de landsspecifika inställningarna.

Funktionen `TimeSerial` formaterar tidsuppgifter på samma sätt som funktionen `DateSerial` formaterar datum:

```
Dim MyVar As Date

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

Deras parametrar bör anges i ordningen timmar, minuter, sekunder.

Ta fram datum- och tidsuppgifter

Följande funktioner är motsatsen till funktionerna `DateSerial` och `TimeSerial`:

- **`Day(MyDate)`** – returnerar dag i månaden från `MyDate`
- **`Month(MyDate)`** – returnerar månaden från `MyDate`
- **`Year(MyDate)`** – returnerar året från `MyDate`
- **`Weekday(MyDate)`** – returnerar veckodagens nummer från `MyDate`
- **`Hour(MyTime)`** – returnerar timtalet från `MyTime`
- **`Minute(MyTime)`** – returnerar minuttalet från `MyTime`
- **`Second(MyTime)`** – returnerar sekundtalet från `MyTime`

De här variablerna tar fram datum- och / eller tidsavsnitt från en angiven `Date`-variabel. Exemplet

```
Dim MyDate As Date

' ... Initiering av MyDate

If Year(MyDate) = 2003 Then

      ' ... Angivet datum infaller under år 2003
End If
```

kontrollerar om det datum som sparades i `MyDate` infaller under år 2003. På samma sätt kontrollerar exemplet

```
Dim MyTime As Date

' ... Initiering av MyTime

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

```
      ' ... Angivet datum infaller under år 2003
End If
```

kontrollerar om `MyTime` infaller mellan klockan `12` och `14`.

Funktionen `Weekday` returnerar numret på veckodagen för det överförda datumet:

```
Dim MyDate As Date
Dim MyWeekday As String

' ... initiera MyDate

Select Case WeekDay(MyDate)
case 1
      MyWeekday = "Söndag"
case 2
      MyWeekday = "Måndag"
case 3
      MyWeekday = "Tisdag"
case 4
      MyWeekday = "Onsdag"
case 5
      MyWeekday = "Torsdag"
case 6
      MyWeekday = "Fredag"
case 7
      MyWeekday = "Lördag"
End Select
```

Obs! Söndag räknas som första dagen i veckan.

Hämta systemdatum och systemtid

Följande funktioner är tillgängliga i StarOffice Basic för att hämta systemdatum och systemtid:

- **`Date`** – returnerar aktuellt datum
- **`Time`** – returnerar aktuellt klockslag
- **`Now`** – returnerar aktuell tidpunkt (datum och klockslag som ett kombinerat värde)

Filer och kataloger

En av de grundläggande aktiviteterna för ett program är att arbeta med filer. StarOffice API innehåller en uppsättning objekt som du kan använda för att skapa, öppna och ändra Office-dokument. De beskrivs i detalj i kapitel 4. Oavsett detta behöver du ibland ha direktåtkomst till filsystemet, söka igenom kataloger och redigera textfiler. I runtime-biblioteket från StarOffice Basic finns flera grundläggande funktioner för dessa aktiviteter.

Vissa DOS-specifika fil- och katalogfunktioner finns inte längre kvar i StarOffice 7.x, eller också är deras funktion begränsad. Funktionerna `ChDir`, `ChDrive` och `CurDir` stöds t.ex. inte längre.
Vissa DOS-specifika egenskaper används inte längre i funktioner som förväntar filegenskaper som

parametrar (t.ex. för att skilja dolda filer från systemfiler). Den här ändringen var nödvändig för att uppnå största möjliga grad av plattformsoberoende för StarOffice.

Administrera filer

Söka igenom kataloger

Funktionen `Dir` i StarOffice Basic söker igenom kataloger efter filer och underordnade kataloger. När den först begärs måste en sträng med sökvägen till de kataloger som ska sökas igenom tilldelas `Dir` som första parameter. Den andra parametern i `Dir` anger den fil eller katalog som ska sökas. StarOffice Basic returnerar namnet på den första påträffade katalogposten. När nästa post ska hämtas bör funktionen `Dir` begäras utan parametrar. Om funktionen `Dir` inte hittar fler poster returnerar den en tom sträng.

Följande exempel visar hur funktionen `Dir` kan användas för att begära alla filer som finns i en katalog. Proceduren sparar de enskilda filnamnen i variabeln `AllFiles` och visar detta i en meddelanderuta.

```
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
```

Siffran `0` som används som andra parameter i funktionen `Dir` säkerställer att `Dir` bara returnerar namn på filer och att kataloger ignoreras. Följande parametrar kan anges här:

- `0`: returnerar normala filer
- `16`: underordnade kataloger

Följande exempel är nästan identiskt med det ovanstående, men funktionen `Dir` överför värdet 16 som en parameter, vilket returnerar de underordnade katalogerna i en mapp i stället för filnamnen.

```
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
```

När den anropas i StarOffice Basic, till skillnad från i VBA, returnerar funktionen `Dir` med parametern `16` bara de underordnade katalogerna i en mapp. (I VBA returnerar funktionen också namnen på standardfilerna så att ytterligare kontroll behövs för att bara hämta katalogerna.)

Alternativen i VBA för att söka igenom kataloger efter filer med egenskaperna *dold*, s*ystemfil, arkiverad* och *volymnamn* existerar inte i StarOffice Basic eftersom motsvarande filsystemfunktioner inte är tillgängliga i alla operativsystem.

De angivna sökvägarna i `Dir` kan använda platshållarna `*` och `?` i både VBA och StarOffice Basic. I StarOffice Basic kan platshållaren `*` emellertid bara vara det sista tecknet i ett filnamn och/eller filnamnstillägg, vilket inte är fallet i VBA.

Skapa och ta bort kataloger

StarOffice Basic innehåller funktionen `MkDir` som används för att skapa kataloger.

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

Den här funktionen skapar kataloger och underordnade kataloger. Alla kataloger som behövs i en hierarki skapas också, om det behövs. Om t.ex. bara katalogen `C:\SubDir1` existerar så skapar anropet

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

både katalogen `C:\SubDir1\SubDir2` och katalogen `C:\SubDir1\SubDir2\SubDir3`.

Funktionen `RmDir` tar bort kataloger.

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

Om katalogen innehåller underordnade kataloger eller filer tas ***dessa också bort.*** Du bör därför vara försiktig när du använder `RmDir`.

I VBA relaterar funktionerna `MkDir` och `RmDir` bara till den aktuella katalogen. I StarOffice Basic däremot kan `MkDir` och `RmDir` användas för att skapa eller ta bort katalognivåer.

I VBA genererar `RmDir` ett felmeddelande om en katalog innehåller en fil. I StarOffice Basic tas katalogen *och alla dess filer* bort.

Kopiera, byta namn på, ta bort och kontrollera existensen hos filer

Anropet

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

skapar en kopia av källfilen `Source` under namnet på aktuell `Destination`.

Med hjälp av funktionen

```
Name OldName As NewName
```

kan du byta namn på filen `OldName` till `NewName`. Nyckelordssyntaxen `As` och det faktum att inget kommatecken används går tillbaka på Basic-språkets rötter.

Anropet

```
Kill(Filename)
```

tar bort filen `Filename`. Om du vill ta bort en katalog (inklusive dess filer) använder du funktionen `RmDir`.

Funktionen `FileExists` kan användas för att kontrollera om en fil existerar:

```
If FileExists(Filename) Then
      MsgBox "Filen existerar."
End If
```

Läsa och ändra filegenskaper

När du arbetar med filer är det ibland viktigt att kunna bestämma filens egenskaper, när filen senast ändrades och filens längd.

Anropet

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

returnerar vissa egenskaper för en fil. Returvärdet tillhandahålls som en bitmask där följande värden är möjliga:

- 1 : skrivskyddad fil
- 16 : namn på en katalog

Exemplet

```
Dim FileMask As Integer
Dim FileDescription As String

FileMask = GetAttr("test.txt")

If (FileMask AND 1) > 0 Then
       FileDescription = FileDescription & " skrivskyddad "
End IF

If (FileMask AND 16) > 0 Then
       FileDescription = FileDescription & " katalog "
End IF

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

MsgBox FileDescription
```

bestämmer bitmasken för filen `test.txt` och kontrollerar om den är skrivskyddad och om den är en katalog. Om ingetdera är fallet tilldelas `FileDescription` strängen "normal".

De flaggor som används i VBA för att söka efter filegenskaperna *dold*, *systemfil, arkiverad* och *volymnamn* stöds inte i StarOffice Basic eftersom de är Windows-specifika och inte tillgängliga eller bara delvis tillgängliga i andra operativsystem.

Funktionen `SetAttr` tillåter att egenskaperna i en fil ändras. Anropet

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

kan därför användas för att ge en fil skrivskyddad status. Befintlig skrivskyddad status kan tas bort med följande anrop:

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

Datum och tid för det senaste tillägget i en fil kommer från funktionen `FileDateTime`. Datumet formateras här i enlighet med de landsspecifika inställningar som används i systemet.

```
FileDateTime("test.txt")      ' Anger datum och tid för det senaste tillägget i filen.
```

Funktionen `FileLen` bestämmer längden på en fil i byte (som ett långt heltalsvärde).

```
FileLen("test.txt")           ' Anger filens längd i byte
```

Skriva och läsa textfiler

StarOffice Basic erbjuder en hel uppsättning metoder för att läsa och skriva filer. Följande beskrivningar gäller textfiler (***inte*** textdokument).

Skriva textfiler

Innan du kommer åt en textfil måste den öppnas. Till detta behövs en ledig ***fildescriptor*** som entydigt identifierar filen för senare filåtkomst.

Funktionen `FreeFile` används för att skapa en ledig fildescriptor. Fildescriptorn används som en parameter för instruktionen `Open` som öppnar filen. Om du vill öppna en fil så att den kan anges som en textfil är `Open`-anropet:

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

`Filename` är en sträng som innehåller filens namn. `FileNo` är den fildescriptor som är skapad av funktionen `FreeFile`.

När filen är öppnad kan instruktionen `Print` beskrivas rad för rad:

```
Print #FileNo, "Det här är en testrad."
```

`FileNo` står också för fildescriptorn här. Den andra parametern anger den text som ska sparas som en rad i textfilen.

När skrivprocessen är avslutad måste filen stängas med ett `Close`-anrop:

```
Close #FileNo
```

Även här bör fildescriptorn anges.

Följande exempel visar hur en textfil öppnas, beskrivs och stängs:

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

Filename = "c:\data.txt"                          ' Definiera filnamn
FileNo = Freefile                                 ' Bestämma ledig fildescriptor

Open Filename For Output As #FileNo               ' Öppna fil (skrivläge)
Print #FileNo, "Det här är en textrad"            ' Spara rad
Print #FileNo, "Det här är ännu en textrad"       ' Spara rad
Close #FileNo                                     ' Stäng fil
```

Läsa textfiler

Textfiler läses på samma sätt som de skrivs. Instruktionen `Open` som används för att öppna filen innehåller uttrycket `For Input` i stället för `For Output`, och i stället för kommandot `Print` som skriver data bör instruktionen `Line Input` användas för att läsa data.

När en textfil anropas används slutligen instruktionen

```
eof(FileNo)
```

för att kontrollera om slutet av filen är nådd.

Följande exempel visar hur en textfil kan läsas in:

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

' Definiera filnamn
Filename = "c:\data.txt"

' Bestämma ledig fildescriptor
FileNo = Freefile

' Öppna fil (lässläge)
Open Filename For Input As FileNo

' Kontrollera om filens slut är nått.
Do While not eof(FileNo)

      ' Läs rad
      Line Input #FileNo, CurrentLine
      If CurrentLine <>"" then
             Msg = Msg & CurrentLine & Chr(13)
      end if

Loop

' Stäng fil
Close #FileNo

Msgbox Msg
```

De enskilda raderna hämtas i en `Do While`-loop, sparas i variabeln `Msg` och visas till slut i en meddelanderuta.

Meddelanderutor och inmatningsfält

StarOffice Basic innehåller funktionerna `MsgBox` och `InputBox` för enkel användarkommunikation.

Visa meddelande

`MsgBox` visar en enkel informationsruta som kan ha en eller flera knappar. I sin enklaste form

```
MsgBox "Det här är information!"
```

innehåller `MsgBox` bara text och en OK-knapp.

Informationsrutans utseende kan ändras med en parameter. Parametern erbjuder möjlighet att lägga till ytterligare knappar, definiera en förtilldelad knapp och lägga till en informationssymbol. Värdena för att välja knappar är:

- 0 – OK-knapp
- 1 – OK-knapp och Avbryt-knapp
- 2 – Avbryt-knapp och Upprepa-knapp.
- 3 – Ja-knapp, Nej-knapp och Avbryt-knapp.
- 4 – Ja-knapp och Nej-knapp.
- 5 – Upprepa-knapp och Avbryt-knapp.

Om du vill ange en knapp som standardknapp lägger du till något av följande värden i parametervärdet från listan med knappalternativ. Om du t.ex. vill skapa Ja-, Nej- och Avbryt-knappar (värde 3) där Avbryt är standard (värde 512), så blir parametervärdet 3 + 512 = 515.

- 0 – Första knappen är standardvärde
- 256 – Andra knappen är standardvärde
- 512 – Tredje knappen är standardvärde

Slutligen är följande informationssymboler tillgängliga och kan också visas genom att lämpliga parametervärden läggs till:

- 16 – Stopptecken
- 32 – Frågetecken
- 48 – Utropstecken
- 64 – Tipsikon

Anropet

```
MsgBox "Vill du fortsätta?",  292
```

visar en informationsruta som innehåller Ja- och Nej-knappar (värde 4), där den andra knappen (Nej) anges som standardvärde (värde 256), och rutan dessutom innehåller ett frågetecken (värde 32), 4+256+32=292

Om en informationsruta innehåller flera knappar bör ett returvärde sökas för att bestämma vilken knapp som har valts. Följande returvärden är tillgängliga i det här fallet:

- 1 – OK
- 2 – Avbryt
- 4 – Upprepa
- 5 – Ignorera
- 6 – Ja
- 7 – Nej

I föregående exempel kan returvärdena kontrolleras enligt följande:

```
If MsgBox ("Vill du fortsätta?",  292) = 6 Then
      ' Ja-knappen har valts
Else
      ' Nej-knappen har valts
End IF
```

Förutom informationstexten och parametern som arrangerar informationsrutan så tillåter `MsgBox` även en tredje parameter som definierar texten i rutans rubrik:

```
MsgBox "Vill du fortsätta?",  292, "Rutans rubrik"
```

Om ingen rubrik anges för rutan är standardvärdet "soffice".

Inmatningsfält för sökning av enkla strängar

Funktionen `InputBox` söker enkla strängar från användaren. Den är därmed ett enklare alternativ till att konfigurera dialogrutor. `InputBox` tar emot tre standardparametrar:

- en informationstext
- en rubrik för rutan
- ett standardvärde som kan läggas till i inmatningsfältet

```
InputVal = InputBox("Ange värde:", "Test", "standardvärde")
```

Som returvärde visar `InputBox` den sträng som matas in av användaren.

Övriga funktioner

Beep

Funktionen `Beep` spelar upp ett systemljud som kan fungera som varning när en användare vidtar en felaktig åtgärd. `Beep` har inga parametrar:

```
Beep  ' genererar ett informationsljud
```

Shell

Externa program kan startas med `Shell`-funktionen.

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

`Pathname` definierar sökvägen till det program som ska köras. `Windowstyle` definierar det fönster där programmet ska startas. Följande värden är möjliga:

- 0 – Programmet ges fokus och startar i ett dolt fönster.
- 1 – Programmet ges fokus och startar i ett normalstort fönster.
- 2 – Programmet ges fokus och startar i ett minimerat fönster.
- 3 – Programmet ges fokus och startar i ett maximerat fönster.
- 4 – Programmet startar i ett normalstort fönster utan att ges fokus.
- 6 – Programmet startar i ett minimerat fönster, fokus blir kvar i det aktuella fönstret.
- 10 – Programmet startar i helskärmsläge.

Den tredje parametern, `Param,` tillåter att kommandoradsparametrar överförs till det program som ska startas.

Wait

Funktionen `Wait` avbryter programkörningen under en angiven tidsrymd. Vänteperioden anges i millisekunder. Kommandot

```
Wait 2000
```

anger ett avbrott på 2 sekunder (2000 millisekunder).

Environ

Funktionen `Environ` returnerar operativsystemets miljövariabler. Olika typer av data sparas här, beroende på system och konfiguration. Anropet

```
Dim TempDir

TempDir=Environ ("TEMP")
```

bestämmer miljövariablerna för den temporära katalogen i operativsystemet.

KAPITEL 4

Introduktion till StarOffice API

StarOffice API är ett universellt programmeringsgränssnitt för åtkomst till StarOffice. Du kan använda StarOffice API för att skapa, öppna, ändra och skriva ut StarOffice-dokument. Gränssnittet ger dig möjlighet att utöka funktionaliteten i StarOffice genom att skapa egna makron och dialogrutor.

StarOffice API går inte bara att använda med StarOffice Basic, utan även med andra programmeringsspråk som Java och C++. Detta möjliggörs av tekniken ***Universal Network Objects*** (UNO), som tillhandahåller ett gränssnitt för diverse olika programmeringsspråk.

Det här kapitlet handlar om hur StarOffice kan användas i StarOffice Basic med hjälp av UNO. Huvudbegreppen i UNO beskrivs utifrån en StarOffice Basic-programmerares synvinkel. Detaljerad information om hur du arbetar med de olika delarna av StarOffice API finns i de följande kapitlen.

UNO (Universal Network Objects)

StarOffice tillhandahåller ett programmeringsgränssnitt i form av UNO (Universal Network Objects). Det är ett objektorienterat programmeringsgränssnitt som StarOffice delar upp i olika objekt som i sin tur säkerställer programkontrollerad åtkomst till Office-paketet.

Eftersom StarOffice Basic är ett procedurellt programmeringsspråk har flera språkliga begrepp behövt läggas till för att möjliggöra användning av UNO.

För att kunna använda ett UNO-objekt i StarOffice Basic behöver du en variabeldeklaration för det tillhörande objektet. Deklarationen görs med instruktionen `Dim` (se kapitel 2). Typbeteckningen `Object` bör användas för att deklarera en objektvariabel:

```
Dim Obj As Object
```

Anropet deklarerar objektvariabeln `Obj`.

Den skapade objektvariabeln måste sedan initieras så att den kan användas. Detta kan göras med `createUnoService`-funktionen:

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

Det här anropet tilldelar variabeln `Obj` en referens till det nyligen skapade objektet. `com.sun.star.frame.Desktop` påminner om en objekttyp. I UNO-terminologi kallas den emellertid för en ***"tjänst"*** och inte en typ. I enlighet med UNO-filosofin beskrivs `Obj` som en ***referens till ett objekt som stöder tjänsten*** `com.sun.star.frame.Desktop`. Termen "tjänst" i StarOffice Basic motsvarar därför termerna ***type*** och ***class*** som används i andra programmeringsspråk.

Det finns dock en viktig skillnad: ett UNO-objekt kan stödja flera tjänster samtidigt. Vissa UNO-tjänster stöder i sin tur andra tjänster, så att du kan få tillgång till en hel uppsättning tjänster via ett objekt. Det är möjligt att t.ex. det ovannämnda objektet, som är baserat på tjänsten `com.sun.star.frame.Desktop`, också innehåller andra tjänster för att ladda dokument och för att avsluta programmet.

Medan strukturen hos ett objekt i VBA definieras av den klass det tillhör, så definieras strukturen i StarOffice Basic av de tjänster objektet stöder. Ett VBA-objekt tilldelas alltid exakt en klass. Ett StarOffice Basic-objekt kan däremot stödja flera tjänster.

Egenskaper och metoder

Ett objekt i StarOffice Basic innehåller en uppsättning egenskaper och metoder som kan anropas via objektet.

Egenskaper

Egenskaper påminner om egenskaperna i ett objekt, t.ex. `Filename` och `Title` för ett `Document`-objekt.

Egenskaperna anges med en enkel tilldelning:

```
Document.Title = "Programmerarhandbok för StarOffice 7.0"
Document.Filename = "progman.sxv"
```

En egenskap har en typ som definierar vilka värden den kan registrera, precis som en vanlig variabel.
De ovanstående egenskaperna `Filename` och `Title` är av strängtyp.

Verkliga egenskaper och imiterade egenskaper

De flesta egenskaper hos ett objekt i StarOffice Basic definieras som sådana i UNO-beskrivningen av tjänsten. Förutom dessa "verkliga" egenskaper finns det också egenskaper i StarOffice Basic som består av två metoder på UNO-nivå. Den ena används för att söka värdet för egenskapen och den andra för att ställa in det (metoderna `get` och `set`). Egenskapen har imiterats virtuellt från två metoder. Teckenobjekt i UNO tillhandahåller t.ex. metoderna `getPosition` och `setPosition` genom vilka den tillhörande nyckelpunkten kan anropas och ändras. StarOffice Basic-programmerare kan komma åt värdena via egenskapen `Position`. Oavsett detta är även de ursprungliga metoderna tillgängliga (i vårt exempel `getPosition` och `setPosition`).

Metoder

Metoder kan beskrivas som funktioner som relaterar direkt till ett objekt, och genom vilka objektet anropas. Ovanstående `Document`-objekt kan t.ex. tillhandahålla en `Save`-metod som kan anropas enligt följande:

```
Document.Save()
```

Metoder kan liksom funktioner innehålla parametrar och returvärden. Syntaxen för sådana metodanrop är orienterad mot klassiska funktioner. Anropet

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

anger också parametern `True` för dokumentobjektet när Save-metoden begärs.
När metoden är slutförd sparar `Save` ett returvärde i variabeln `Ok`.

Modul, tjänster och gränssnitt

StarOffice erbjuder hundratals tjänster. För att dessa tjänster ska vara enklare att överblicka är de kombinerade i moduler. Moduler har ingen annan funktionell betydelse för StarOffice Basic-programmerare. När du anger namn på en tjänst är det bara modulnamnet som är av någon betydelse eftersom det också måste ingå i namnet. Det fullständiga namnet på en tjänst består av uttrycket `com.sun.star` som anger att det är en StarOffice-tjänst, följt av modulens namn, t.ex. `frame`, och slutligen själva tjänstens namn, t.ex. `Desktop`. Det fullständiga namnet i detta exempel blir då:

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

Förutom termerna modul och tjänst introducerar UNO termen ***"gränssnitt".*** Termen kan vara bekant för Java-programmerare men används inte i Basic.

Ett gränssnitt kombinerar flera metoder. En tjänst i UNO stöder inte metoder i ordets snävaste mening, utan snarare gränssnitt, vilka i sin tur innehåller olika metoder. Med andra ord tilldelas tjänsten metoderna (som kombinationer) i gränssnitt. Den detaljen kan vara av intresse särskilt för Java- och C++-programmerare, eftersom gränssnittet behövs för att begära en metod i dessa språk. I StarOffice Basic är detta dock irrelevant. Här anropas metoderna direkt via det aktuella objektet.

För att förstå API kan det emellertid vara användbart att ha tilldelningen av metoder till olika gränssnitt nära till hands, eftersom många gränssnitt används i de olika tjänsterna. Om du är bekant med ett gränssnitt kan du överföra kunskapen från en tjänst till en annan.

Vissa centrala gränssnitt används så ofta att de visas igen i slutet av det här kapitlet, utlösta av olika tjänster.

Verktyg för att arbeta med UNO

Frågan kvarstår om vilka objekt – eller tjänster om vi ska hålla oss till UNO-terminologi – som stöder vilka egenskaper, metoder och gränssnitt, och hur dessa kan fastställas. Utöver den här handboken kan du få mer information om objekt från följande källor: metoden `supportsService`, felsökningsmetoderna, Developer's Guide och API-referensen.

Metoden supportsService

Ett antal UNO-objekt stöder metoden `supportsService` som du kan använda för att avgöra om ett objekt stöder en viss tjänst eller inte. Anropet

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

avgör t.ex. om objektet `TextElement` stöder tjänsten `com.sun.star.text.Paragraph`.

Egenskaper för felsökning

Varje UNO-objekt i StarOffice Basic ***vet*** vilka egenskaper, metoder och gränssnitt det redan innehåller. Det innehåller också egenskaper som returnerar den här informationen i form av en lista. Dessa egenskaper är:

`DBG_properties` – returnerar en sträng som innehåller alla egenskaper för ett objekt

`DBG_methods` – returnerar en sträng som innehåller alla metoder för ett objekt

`DBG_supportetInterfaces` – returnerar en sträng som innehåller alla gränssnitt som stöder ett objekt.

Följande programkod visar hur `DBG_properties` och `DBG_methods` kan användas i riktiga program. Först skapas tjänsten `com.sun.star.frame.Desktop` och sedan visas de egenskaper och metoder som stöds i meddelanderutor.

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

MsgBox Obj.DBG_properties
MsgBox Obj.DBG_methods
```

När du använder `DBG_properties`, bör du tänka på att funktionen returnerar alla egenskaper som en viss tjänst teoretiskt kan stödja. Det finns emellertid inga garantier för att dessa tjänster också kan användas av objektet i fråga. Innan du anropar egenskaper bör du därför använda funktionen `IsEmpty` för att kontrollera om de faktiskt är tillgängliga.

API-referens

Mer information om tillgängliga tjänster, deras gränssnitt, metoder och egenskaper finns i API-referensen för StarOffice API. Den finns på www.openoffice.org på adressen:

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

Översikt över några centrala gränssnitt

Vissa gränssnitt i StarOffice finns på många ställen i StarOffice API. De definierar uppsättningar av metoder för abstrakta aktiviteter som kan användas för olika problem. Här följer en översikt över de vanligaste av dessa gränssnitt.

Objektens ursprung förklaras senare i den här handboken. Här behandlas bara vissa abstrakta aspekter av objekt för vilka StarOffice API erbjuder några centrala gränssnitt.

Skapa sammanhangsberoende objekt

StarOffice API erbjuder två alternativ för att skapa objekt. Det ena kan hittas i funktionen `createUnoService` som nämns i början av det här kapitlet. `createUnoService` skapar ett objekt som kan användas universellt. Sådana objekt och tjänster kallas också ***sammanhangsoberoende tjänster***.

Förutom dessa sammanhangsoberoende tjänster finns det också ***sammanhangsberoende tjänster*** vars objekt bara går att använda tillsammans med ett annat objekt. Ett ritobjekt i ett tabelldokument kan t.ex. bara existera tillsammans med detta dokument.

Gränssnittet com.sun.star.lang.XMultiServiceFactory

Sammanhangsberoende objekt skapas vanligen med en objektmetod som objektet är beroende av. Metoden `createInstance` som definieras i gränssnittet `XMultiServiceFactory` används framför allt i dokumentobjekt.

Ovan nämnda ritobjekt kan t.ex. skapas enligt följande med hjälp av ett tabellobjekt:

```
Dim RectangleShape As Object

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

En styckeformatmall i ett textdokument skapas på samma sätt:

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

Namngiven åtkomst till underordnade objekt

Gränssnitten XNameAccess och XNameContainer används i objekt som innehåller underordnade objekt vilka kan kommas åt med naturliga språknamn.

Medan XNamedAccess tillåter åtkomst till enskilda objekt så tar XNameContainer hand om infogning, ändring och borttagning av element.

Gränssnittet com.sun.star.container.XNameAccess

Ett exempel på hur XNameAccess kan användas är ett tabellobjekt i en tabell. Det kombinerar alla sidor i tabellen. De enskilda sidorna kan kommas åt med metoden getByName från XNameAccess:

```
Dim Sheets As Object
Dim Sheet As Object

Sheets = Spreadsheet.Sheets
Sheet = Sheets.getByName("Tabell1")
```

Metoden getElementNames ger en översikt över namnen på alla element. Som resultat returneras ett datafält som innehåller namnen. Följande exempel visar hur namnen på alla element i en tabell på så sätt kan bestämmas och visas i en loop:

```
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
```

Metoden hasByName i gränssnittet XNameAccess tar reda på om ett underordnat objekt med ett visst namn existerar i huvudobjektet. Följande exempel visar ett meddelande som informerar användaren om huruvida objektet Spreadsheet innehåller en sida med namnet Tabell1.

```
Dim Sheets As Object

Sheets = Spreadsheet.Sheets
If Sheets.HasByName("Tabell1") Then
      MsgBox " Tabell1 finns"
Else
      MsgBox " Tabell1 finns inte"
End If
```

Gränssnittet com.sun.star.container.XNameContainer

Gränssnittet `XNameContainer` sköter infogning, borttagning och ändring av underordnade element i ett huvudobjekt. De aktuella funktionerna är `insertByName`, `removeByName` och `replaceByName`.

Följande är ett praktiskt exempel på detta. Här anropas ett textdokument som innehåller ett `StyleFamilies`-objekt vilket används för att skapa styckeformatmallar (ParagraphStyles) av det tillgängliga dokumentet.

```
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")
```

Raden `insertByName` infogar formatmallen `NewStyle` under samma namn i objektet `ParagraphStyles`. Raden `replaceByName` ändrar objektet bakom `ChangingStyle` till `NewStyle`. Slutligen tar anropet `removeByName` bort objektet bakom `OldStyle` från `ParagraphStyles`.

Indexbaserad åtkomst till underordnade objekt

Gränssnitten `XIndexAccess` och `XIndexContainer` används i objekt som innehåller underordnade objekt och kan nås med ett index.

`XIndexAccess` tillhandahåller metoder för åtkomst av enskilda objekt. `XIndexContainer` tillhandahåller metoder för att infoga och ta bort element.

Gränssnittet com.sun.star.container.XIndexAccess

`XIndexAccess` tillhandahåller metoderna `getByIndex` och `getCount` för att anropa underordnade objekt. `getByIndex` förser ett objekt med ett visst index. `getCount` returnerar antalet tillgängliga objekt.

```
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)
      ' Redigera tabell
Next I
```

Exemplet visar en loop som går igenom alla tabellelement ett efter ett och sparar en referens till vart och ett i objektvariabeln `Sheet`. När du arbetar med index bör du notera att `getCount` returnerar antalet element. Elementen i `getByIndex` är emellertid numrerade med början på `0`. Loopens räknarvariabel går därför från `0` till `getCount()-1`.

Gränssnittet com.sun.star.container.XIndexContainer

Gränssnittet `XIndexContainer` tillhandahåller funktionerna `insertByIndex` och `removeByIndex`. Parametrarna är strukturerade på samma sätt som motsvarande funktioner i `XNameContainer`.

Iterativ åtkomst till underordnade objekt

I vissa fall kan ett objekt innehålla en lista över underordnade objekt som inte går att nå med vare sig ett namn eller ett index. I sådana lägen är gränssnitten `XEnumeration` och `XenumerationAccess` användbara. De tillhandahåller en mekanism där alla underordnade element i ett objekt kan skickas steg för steg, utan att direktadressering måste användas.

Gränssnitten com.sun.star.container.XEnumeration och XenumerationAccess

Huvudobjektet måste tillhandahålla gränssnittet `XEnumerationAccess` som bara innehåller en `createEnumeration`-metod. Detta returnerar ett hjälpobjekt som i sin tur tillhandahåller gränssnittet `XEnumeration` med metoderna `hasMoreElements` och `nextElement`. Via dem har du sedan tillgång till de underordnade objekten.

Följande exempel stegar igenom alla stycken i en text:

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

Exemplet skapar först hjälpobjektet `ParagraphEnumeration`. Detta returnerar gradvis textens enskilda stycken i en loop. Loopen avbryt så snart metoden `hasMoreElements` returnerar värdet `False` som signalerar att slutet på texten har nåtts.

KAPITEL 5

Arbeta med StarOffice-dokument

StarOffice API är strukturerat för att så många delar som möjligt ska kunna användas universellt i olika aktiviteter. Det inbegriper gränssnitt och tjänster för att skapa, öppna, spara, konvertera och skriva ut dokument och för malladministration. Eftersom dessa funktionsområden är tillgängliga i alla typer av dokument förklaras de först i det här kapitlet.

StarDesktop

När du arbetar med dokument är det två tjänster som används mest:

- Tjänsten `com.sun.star.frame.Desktop` som är snarlik kärntjänsten i StarOffice. Den tillhandahåller funktionerna för ramobjektet i StarOffice, under vilket alla dokumentfönster klassificeras. Det går också att skapa, öppna och importera dokument med den här tjänsten.
- Den grundläggande funktionaliteten hos enskilda dokumentobjekt tillhandahålls av tjänsten `com.sun.star.document.OfficeDocument`. Här finns metoder för att spara, exportera och skriva ut dokument.

Tjänsten `com.sun.star.frame.Desktop` öppnas automatiskt när StarOffice startas. För att åstadkomma detta skapar StarOffice ett objekt som kan nås via det globala namnet `StarDesktop`.

Det viktigaste gränssnittet i `StarDesktop` är `com.sun.star.frame.XComponentLoader`. Det täcker i princip metoden `loadComponentFromURL` som skapar, importerar och öppnar dokument.

Namnet på objektet `StarDesktop` går tillbaka till StarOffice 5, där alla dokumentfönster var inbäddade i ett gemensamt program som hette StarDesktop. I den aktuella versionen av StarOffice används inte längre StarDesktop synligt. Namnet StarDesktop har emellertid behållits för ramobjektet i StarOffice, eftersom det tydligt anger att det är ett huvudobjekt för hela programmet.

Objektet `StarDesktop` intar positionen som efterträdare till objektet `Application` i StarOffice 5, som tidigare användes som rotobjekt. Till skillnad från det gamla `Application`-objektet är det emellertid huvudansvarigt för att öppna nya dokument. De funktioner som finns i det gamla `Application`-objektet för att kontrollera bildskärmsvisningen av StarOffice (t.ex. `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) används inte längre.

Medan du kommer åt det aktiva dokumentet i Word via `Application.ActiveDocument` och i Excel via `Application.ActiveWorkbook`, så använder StarOffice `StarDesktop` för denna aktivitet. Du kommer åt det aktiva dokumentobjektet i StarOffice 7 via egenskapen `StarDesktop.CurrentComponent`.

Grundläggande information om dokument i StarOffice

När du arbetar med StarOffice-dokument kan det vara bra att ta itu med vissa grundläggande frågor rörande dokumentadministration i StarOffice. Här ingår hur filnamn ska struktureras för StarOffice-dokument, liksom i vilket format filer ska sparas.

Filnamn i URL-notation

Eftersom StarOffice är utformat som ett plattformsoberoende program används URL-notation (som är oberoende av operativsystem), enligt definitionen i Internet Standard RFC 1738 för filnamn. Standardfilnamn som använder det här systemet inleds med prefixet

`file:///`

följt av den lokala sökvägen. Om filnamnet innehåller underordnade kataloger avgränsas dessa med ett enkelt *framåtlutat* snedstreck, inte med ett bakvänt snedstreck som vanligen används under Windows. Följande sökväg refererar till filen `test.sxw` i katalogen `doc` på enhet `C:`.

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

Om du vill konvertera lokala filnamn till en URL så tillhandahåller StarOffice funktionen `ConvertToUrl`.
Om du vill konvertera en URL till ett lokalt filnamn så tillhandahåller StarOffice funktionen `ConvertFromUrl`:

```
MsgBox ConvertToUrl("C:\doc\test.sxw")
            ' ger file:///C:/doc/test.sxw

MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
            '  ger (under Windows) c:\doc\test.sxw
```

Exemplet konverterar ett lokalt filnamn till en URL som visas i en meddelanderuta. Sedan konverteras en URL till ett lokalt filnamn vilket också visas.

Internet Standard RFC 1738, som det här är baserat på, tillåter användning av tecknen `0-9`, `a-z` och `A-Z`. Alla andra tecken infogas med escape-tecken i URL:erna. För att åstadkomma detta konverteras de till sina hexadecimala värden i teckenuppsättningen ISO 8859-1 (ISO-Latin) och föregås av ett procenttecken. Därför förvandlas t.ex. ett mellanslag i ett lokalt filnamn till `%20` i URL:en.

Filformatet XML

Sedan version 6.0 erbjuder StarOffice ett XML-baserat filformat. Med hjälp av XML kan användare också öppna och redigera filer i andra program.

Komprimera filer

Eftersom XML är baserat på vanliga textfiler blir de resulterande filerna ofta mycket stora. StarOffice komprimerar därför filerna och sparar dem som en ZIP-fil.

Med ett metodalternativ av typen `storeAsURL` kan användaren spara de ursprungliga XML-filerna direkt. Se storeAsURL-metodalternativen på sidan 83.

Skapa, öppna och importera dokument

Dokument öppnas, importeras och skapas med metoden

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

Den första parametern i `loadComponentFromURL` anger den tillhörande filens URL.

Som andra parameter förväntar `loadComponentFromURL` ett namn på fönstrets ramobjekt som StarOffice skapar internt för administrativa ändamål. Det fördefinierade namnet `_blank` anges vanligen här, och säkerställer att StarOffice skapar ett nytt fönster. Som ett alternativ kan också `_hidden` anges och säkerställa att motsvarande dokument laddas men förblir osynligt.

Med de här parametrarna kan användare öppna ett StarOffice-dokument, eftersom de sista två parametrarna kan tilldelas platshållare (dummyvärden):

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

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

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

Anropet ovan öppnar filen text.sxw och visar filen i ett nytt fönster.

Det går att öppna hur många objekt som helst på det här sättet med StarOffice Basic och sedan redigera de returnerade dokumentobjekten.

`StarDesktop.loadComponentFromURL` ersätter metoderna `Documents.Add` och `Documents.Open` från den gamla versionen av StarOffice API.

Ersätta innehållet i dokumentfönstret

De reserverade namnen `_blank` och `_hidden` för parametern `Frame` gör att StarOffice skapar ett nytt fönster för varje anrop till `loadComponentFromURL`. I vissa situationer ska innehållet i det befintliga fönstret ersättas. Om detta är fallet ska fönstrets ramobjekt innehålla ett explicit namn. Observera att detta namn inte får börja med ett understreck. Dessutom måste parametern `SearchFlags` anges så att motsvarande ramverk skapas, om det inte redan finns. Motsvarande konstant för `SearchFlags` är:

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

Följande exempel visar hur innehållet i ett öppet fönster kan ersättas med parametern frame och SearchFlags:

```
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 "Klicka på OK för att visa det andra dokumentet."

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

I exemplet öppnas först filen test.sxw i ett nytt fönster med ramnamnet MyFrame. När användaren har bekräftat i meddelanderutan ersätts innehållet i fönstret med test2.sxw.

Metoden loadComponentFromURL

Den fjärde parametern i anropet till funktionen loadComponentFromURL är ett PropertyValue -datafält som anger olika alternativ för att öppna och skapa dokument. Datafältet måste innehålla en struktur av typen PropertyValue för varje alternativ, där alternativet namn ingår som en sträng, samt det associerade värdet.

loadComponentFromURL stöder följande alternativ:

- **AsTemplate (Boolean)** – om sant laddas ett nytt namnlöst dokument från den givna URL:en. Om falskt laddas mallfiler för redigering.
- **CharacterSet (String)** – definierar vilken teckenuppsättning som ett dokument baseras på.
- **FilterName (String)** – anger ett särskilt filter för funktionen loadComponentFromURL. De möjliga filternamnen är definierade i filen \share\config\registry\instance\org\openoffice\office\TypeDetection.xml.
- **FilterOptions (String)** – anger ytterligare alternativ för filter.
- **JumpMark (String)** – när ett dokument har öppnats, flyttas markören till positionen som anges av JumpMark.
- **Password (String)** – anger ett lösenord för en skyddad fil.
- **ReadOnly (Boolean)** – laddar ett skrivskyddat dokument.

I följande exempel visas hur en kommaseparerad textfil i StarOffice Calc kan öppnas med alternativet `FilterName`.

```
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())
```

Datafältet `FileProperties` anger exakt ett värde eftersom den endast innehåller ett alternativvärde.
Egenskapen `Filtername` definierar om StarOffice ska använda ett StarOffice Calc-textfilter för att öppna filer.

Skapa nya dokument

StarOffice skapar automatiskt ett nytt dokument om den fil som URL:en anger är en dokumentmall.

Om du vill skapa ett tomt standarddokument kan du också använda en `private:factory`-URL.

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

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

Det här kodexemplet skapar ett tomt StarOffice-textdokument.

Dokumentobjekt

Funktionen `loadComponentFromURL` från föregående avsnitt returnerar ett dokumentobjekt. Objektet ingår i tjänsten `com.sun.star.document.OfficeDocument` , som i sin tur innehåller två grundläggande gränssnitt:

- gränssnittet `com.sun.star.frame.XStorable` , som används för att spara dokument och
- gränssnittet `com.sun.star.view.XPrintable` , som innehåller metoder för att skriva ut dokument.

Det funktionella definitionsområdet för dokumentobjekt har med några få undantag inte ändrats i och med StarOffice 6. Dokumentobjekten innehåller till exempel fortfarande metoder för att spara och skriva ut dokument. Däremot har namn och parametrar ändrats för metoderna.

Spara och exportera dokument

StarOffice-dokument kan sparas direkt via dokumentobjektet. Metoden `store` i gränssnittet `com.sun.star.frame.XStorable` kan användas för följande ändamål:

```
Doc.store()
```

Detta anrop fungerar om dokumentet redan har tilldelats ett minnesområde. Detta gäller inte för nya dokument. I detta fall används metoden `storeAsURL` . Metoden är också definierad i `com.sun.star.frame.XStorable` och kan användas för att definiera ett dokuments plats:

```
Dim URL As String
Dim Dummy()

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

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

Förutom de ovanstående metoderna, innehåller `com.sun.star.frame.XStorable` även hjälpmetoder som kan användas när dokument ska sparas. Dessa metoder är:

- **`hasLocation()`** – anger om dokumentet har tilldelats en URL.
- **`isReadonly()`** – anger om ett dokument är skrivskyddat.
- **`isModified()`** – anger om ett dokument har ändrats sedan det sparades senast.

Koden för att spara ett dokument kan byggas ut med dessa alternativ så att dokumentet endast sparas om objektet har ändrats och så att användaren endast får en fråga om filnamn om det behövs.

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

I exemplet kontrolleras om dokumentet har ändrats sedan det sparades senast. Dokumentet sparas endast om detta är fallet. Om dokumentet redan har tilldelats en URL och om det inte är skrivskyddat, sparas det med den befintliga URL:en. Om det inte har en URL eller om dokumentet öppnas som skrivskyddat, sparas det med en ny URL.

storeAsURL-metodalternativen

Liksom metoden `loadComponentFromURL`, kan vissa alternativ också anges med ett `PropertyValue`-datafält till metoden `storeAsURL`. Alternativen anger hur StarOffice ska spara ett dokument. `storeAsURL` stöder följande alternativ:

- **`CharacterSet (String)`** – definierar vilken teckenuppsättning som ett dokument baseras på.
- **`FilterName (String)`** – anger ett särskilt filter för funktionen `loadComponentFromURL`. De möjliga filternamnen är definierade i filen `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **`FilterOptions (String)`** – anger ytterligare alternativ för filter.
- **`Overwrite (Boolean)`** – anger att om filen redan finns ska den skrivas över utan att användaren tillfrågas.
- **`Password (String)`** – anger lösenordet för en skyddad fil.
- **`Unpacked (Boolean)`** – spara dokumentet (okomprimerat) i underkataloger.

I följande exempel visas hur alternativet `Overwrite` kan användas tillsammans med `storeAsURL`:

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

' ... Initialize Doc

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

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

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

I exemplet sparas dokumentet `Doc` med det givna filnamnet, även om det redan finns en fil med det namnet.

Skriva ut dokument

Dokument skrivs ut på samma sätt som de sparas, direkt via dokumentobjektet. Metoden `Print` i gränssnittet `com.sun.star.view.Xprintable` kan användas för detta ändamål.
I sin enklaste form ser `print`-anropet ut så här:

```
Dim Dummy()

Doc.print(Dummy())
```

Liksom för metoden `loadComponentFromURL` är parametern `Dummy` ett `PropertyValue` -datafält som används för att ange olika utskriftsalternativ.

Alternativen för metoden print

Metoden `print` förväntar sig ett`PropertyValue` -datafält som innehåller inställningar i utskriftsdialogrutan i StarOffice.

- **`CopyCount (Integer)`** – anger hur många kopior som ska skrivas ut.
- **`FileName (String)`** – skriver ut dokumentet i den angivna filen.
- **`Collate (Boolean)`** – anger att skrivaren ska skriva ut sidorna i sorterad ordning.
- **`Sort (Boolean)`** – sorterar sidorna när flera kopior skrivs ut. (`CopyCount` > 1).
- **`Pages (String)`** – innehåller en lista på de sidor som ska skrivas ut (samma syntax som i utskriftsdialogrutan).

I följande exempel visas hur flera sidor i ett dokument kan skrivas ut med alternativet `Pages:`

```
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())
```

Skrivarval och inställningar

Gränssnittet `com.sun.star.view.XPrintable` innehåller egenskapen `Printer,` som anger den aktuella skrivaren. Egenskapen anges med ett `PropertyValue`-datafält med följande värden:

- **`Name (String)`** – anger skrivarens namn.
- **`PaperOrientation (Enum)`** – anger sidorienteringen (`com.sun.star.view.PaperOrientation.PORTRAIT` för stående format, `com.sun.star.view.PaperOrientation.LANDSCAPE` för liggande format).
- **`PaperFormat (Enum)`** – anger pappersformatet (till exempel com.sun.star.view.PaperFormat.A4 for DIN A4 eller `com.sun.star.view.PaperFormat.Letter` för US Letter).
- **`PaperSize (Size)`** – anger pappersstorleken i hundradels millimetrar.

I följande exempel visas hur den aktuella skrivaren och pappersstorleken kan ändras med egenskapen `Printer`.

```
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      ' corresponds to 20 cm
PaperSize.Height = 20000     ' corresponds to 20 cm

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

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

Doc.Printer = PrinterProperties()
```

I exemplet definieras ett objekt med namnet `PaperSize` av typen `com.sun.star.awt.Size`. Detta krävs för att ange pappersstorleken. Dessutom skapas två datafält för två `PropertyValue`-poster med namnet `PrinterProperties`. Detta datafält initierats sedan med de värden som ska tilldelas egenskapen `Printer`. I UNO-termer är skrivaren inte en verklig egenskap, utan en ***imiterad*** egenskap.

Mallar

Mallar är namngivna listor som innehåller formateringsattribut. De kan användas i alla StarOffice-program och används för att underlätta konsekvent formatering. Om användaren ändrar ett attribut i en mall, kommer programmet automatiskt att justera alla dokumentavsnitt som innehåller attributet. Användaren kan till exempel ändra teckensnittet för alla rubriker på nivå 1 genom en enda ändring i dokumentet. Beroende på vilken typ dokumentet är av, kan StarOffice hantera flera olika typer av mallar.

StarOffice Writer stöder

- teckenformatmallar
- styckeformatmallar
- ramformatmallar
- sidformatmallar
- numreringsmallar

StarOffice Calc stöder

- cellformatmallar
- sidformatmallar

StarOffice Impress stöder

- teckenelementmallar
- presentationsmallar

I StarOffice-terminologi kallas de olika typerna av mallar för `StyleFamilies` i enlighet med `com.sun.star.style.StyleFamily`-tjänsten som de bygger på.
De olika `StyleFamily`-objekten kan kommas åt via dokumentobjektet:

```
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")
```

I exemplet används egenskapen `StyleFamilies` i ett tabelldokument för att skapa en lista med alla tillgängliga cellformatmallar.

De individuella mallarna kan också nås direkt med ett index:

```
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
```

I exemplet ovan har en slinga lagts till som visar namnen på alla cellformatmallar i meddelanderutor.

Mer om olika formateringsalternativ

Varje typ av mall innehåller flera olika individuella formateringsegenskaper. Här följer en översikt över de viktigaste formateringsegenskaperna och i vilket avsnitt de beskrivs:

- Teckenegenskaper, kapitel 6, Textdokument, `com.sun.star.style.CharacterProperties`-tjänsten
- Styckeegenskaper, kapitel 6, Textdokument, `com.sun.star.text.Paragraph`-tjänsten
- Cellegenskaper, kapitel 7, Tabelldokument, `com.sun.star.table.CellProperties`-tjänsten
- Sidegenskaper, kapitel 7, Tabelldokument, `com.sun.star.style.PageStyle`-tjänsten
- Teckenelementegenskaper, kapitel 7, Tabelldokument, Diverse tjänster

Formategenskaperna är inte begränsade till det program för vilket de förklaras. Egenskaperna kan användas i alla program. Till exempel kan de flesta sidegenskaper som beskrivs i kapitel 7 inte bara användas i StarOffice Calc, men också i StarOffice Writer.

Mer information om att arbeta med mallar finns i avsnittet ***Standardvärden för tecken- och styckeegenskaper*** i kapitel 6, Textdokument.

KAPITEL 6

Textdokument

Förutom vanliga strängar, innehåller textdokument även formateringsinformation. Informationen kan finnas var som helst i texten. Strukturen för formatering bygger på tabeller. Tabellerna innehåller förutom endimensionella strängar även tvådimensionella fält. De flesta ordbehandlingsprogram kan placera ritobjekt, textramar och andra objekt mitt i texten. Objekten kan finnas utanför textflödet och kan placeras var som helst på sidan.

I detta kapitel beskrivs de grundläggande gränssnitten och tjänsterna för textdokument. Det första avsnittet behandlar uppbyggnaden av textdokument och beskriver hur ett StarOffice Basic-program kan användas för att manipulera ett StarOffice-dokument. Avsnittet fokuserar på stycken, delar av stycken och deras formatering.

Det andra avsnittet fokuserar på att arbeta effektivt med textdokument. StarOffice innehåller flera hjälpobjekt för att uppnå detta, till exempel objektet `TextCursor`, förutom de som beskrevs i det första avsnittet.

Det tredje avsnittet handlar om att arbeta med andra objekt än text. Det beskriver tabeller, textramar, fältkommandon, bokmärken, innehållskataloger och andra objekt.

Information om hur du skapar, öppnar, sparar och skriver ut dokument finns i kapitel 5, eftersom den informationen inte bara gäller textdokument, men även andra typer av dokument.

Strukturen i ett textdokument

Ett textdokument innehåller fyra kategorier av information:

- själva texten
- mallar för att formatera tecken, stycken och sidor
- andra element än text, till exempel tabeller, grafik och ritobjekt
- globala inställningar för textdokumentet

I detta avsnitt beskrivs olika alternativ för text och textformatering.

Uppbyggnaden av StarOffice 7.x API för StarOffice Writer skiljer sig på några punkter från tidigare versioner. Den gamla API-modellen byggde på att koden arbetade mot objektet `Selection`, vilket nära stämmer överens med det sätt som slutanvändaren arbetar med texten, till exempel att markera text med musen.

I StarOffice 7.x API har denna koppling mellan användar- och programmeringsgränssnitt ersatts. Programmeraren har nu tillgång till alla delar i programmet och kan arbeta med objekt direkt från olika delavsnitt i dokumentet samtidigt. Det gamla `Selection`-objektet finns inte längre.

Stycken och delar av ett stycke

Ett textdokument består huvudsakligen av en sekvens stycken. Styckena saknar namn och index och det går därför inte att ***direkt*** komma åt individuella stycken. Styckena kan däremot räknas upp med objektet `Enumeration` som beskrivs i kapitel 4. Uppräkningen gör att styckena kan redigeras.

När du arbetar med objektet `Enumeration` bör du emellertid vara medveten om följande: objektet returnerar inte bara stycken, men också tabeller (i StarOffice Writer är en tabell egentligen en särskild sorts stycke). Innan du kan använda det returnerade objektet bör du därför kontrollera om objektet stöder tjänsten `com.sun.star.text.Paragraph` för stycken eller tjänsten `com.sun.star.text.TextTable` för tabeller.

I följande exempel används en slinga för att gå igenom innehållet i ett textdokument, och använder ett meddelande i varje iteration som informerar användaren om objektet är ett stycke eller en tabell.

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

' Create document object
Doc = StarDesktop.CurrentComponent

' Create enumeration object
Enum = Doc.Text.createEnumeration

' loop over all text elements
While Enum.hasMoreElements
      TextElement = Enum.nextElement

      If TextElement.supportsService("com.sun.star.text.TextTable") Then
            MsgBox "Det aktuella blocket innehåller en tabell."
      End If

      If TextElement.supportsService("com.sun.star.text.Paragraph") Then
            MsgBox "Det aktuella blocket innehåller ett stycke."
      End If
Wend
```

I exemplet skapas objektet `Doc` som refererar till det aktuella StarOffice-dokumentet. Med hjälp av objektet `Doc`, skapas sedan ett objekt av typen `Enumeration` som itererar genom de individuella delarna i texten (stycken och tabeller) och tilldelar det aktuella elementet till objektet `TextElement`. Exemplet använder metoden `supportsService` för att kontrollera om `TextElement` är ett stycke eller en tabell och visar sedan motsvarande meddelande.

Stycken

Tjänsten `com.sun.star.text.Paragraph` ger åtkomst till innehållet i ett stycke. Texten i stycket kan hämtas och ändras med egenskapen `String` enligt följande:

```
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, "jag", "hon")
            TextElement.String = Replace(TextElement.String, "mig", "henne")
            TextElement.String = Replace(TextElement.String, "min", "hennes")
      End If
Wend
```

I exemplet öppnas det aktuella textdokumentet och innehållet gås igenom med objektet `Enumeration`. I exemplet används egenskapen `TextElement.String` i alla stycken för att komma åt de relevanta styckena, och ersätter strängarna `jag`, `mig` och `min` med orden `hon`, `henne` och `hennes`.
Funktionen `Replace` som används för att ersätta finns inte i det vanliga definitionsområdet för StarOffice Basic. Detta är en instans av exempelfunktionen som beskrivs i kapitel 3 i avsnittet ***Sök och ersätt***.

Den procedur för att komma åt styckena i en text som beskrivs här är kompatibel med att använda egenskapen `Paragraphs` i objekten `Range` och `Document` i VBA. I VBA kan styckena refereras till efter index (till exempel med anropet `Paragraph(1)`). I StarOffice Basic används i stället objektet `Enumeration` som beskrivs ovan.

Det finns ingen direkt motsvarighet i StarOffice Basic till egenskaperna `Characters`, `Sentences` och `Words` i VBA. Däremot kan du växla till en `TextCursor` som du kan använda för att navigera på tecken-, menings- eller ordnivå (se avsnittet *Textmarkören*).

Styckedelar

Det föregående exemplet kan användas för att ändra texten, men ibland kan formateringen gå förlorad.

Detta beror på att ett stycke består av enskilda underobjekt. Varje underobjekt innehåller formateringsinformation. Om det till exempel finns ett ord mitt i ett stycke som är formaterat med fet stil, kommer stycket att representeras av tre styckedelar i StarOffice: delen innan formateringen, det fetstilta ordet och delen efter formateringen.

Om texten i stycket ändras med styckets `String`-egenskap, kommer StarOffice först att ta bort de gamla styckedelarna och sedan infoga en ny styckedel. Formateringen i de tidigare delarna försvinner.

Du kan undvika det här genom att referera till de enskilda styckedelarna i stället för hela stycket. Stycken innehåller ett `Enumeration`-objekt som kan användas för detta ändamål. I följande exempel visas en dubbel slinga som itererar över alla stycken i ett textdokument och de styckedelar de innehåller och använder samma ersättningsprocess som i det tidigare exemplet:

```
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

' loop over all paragraphs
While Enum1.hasMoreElements
	TextElement = Enum1.nextElement
	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = TextElement.createEnumeration

		' gå igenom alla understycken
		While Enum2.hasMoreElements
			TextPortion = Enum2.nextElement
			MsgBox "'" & TextPortion.String & "'"
			TextPortion.String = Replace(TextPortion.String, "jag", "hon")
			TextPortion.String = Replace(TextPortion.String, "mig", "henne")
			TextPortion.String = Replace(TextPortion.String, "min", "hennes")
		Wend

	End If
Wend
```

I exemplet används en dubbel slinga för att gå igenom dokumentet. Den yttre slingan refererar till styckena i texten. Den inre slingan behandlar de olika delarna i styckena. I exemplet ändras innehållet i dessa stycken med egenskapen `String`, liksom i det tidigare exemplet för stycken. Eftersom styckedelarna redigeras direkt, behålls all formateringsinformation vid ersättningarna.

Formatering

Det finns flera olika sätt att formatera text. Det enklaste sättet är att direkt tilldela textsekvenser formateringsegenskaper. Detta kallas ***direkt formatering.*** Direkt formatering används oftast i kortare dokument eftersom formaten kan anges direkt av användaren med musen. Du kan till exempel markera ett visst ord och kursivera det, eller centrera en rad.

Förutom direkt formatering kan du också formatera text med formatmallar. Detta kallas ***indirekt formatering***. Vid indirekt formatering används en fördefinierad formatmall i ett avsnitt i texten. Om textens layout ska ändras behöver användaren bara ändra formatmallen. StarOffice ändrar sedan formateringen på alla ställen där formatmallen används.

I VBA är ett objekts formateringsegenskaper vanligtvis uppdelat på flera olika underobjekt (till exempel `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). Formatet refereras till med sammansatta uttryck (till exempel `Range.Font.AllCaps`). I StarOffice Basic är formateringsegenskaperna direkt tillgängliga via relevanta objekt (`TextCursor`, `Paragraph` och så vidare). I de följande två avsnitten finns en översikt över de tecken- och styckeegenskaper som finns i StarOffice..

I de tidigare versionerna av StarOffice API formaterades texten huvudsakligen via objektet `Selection` och objektets underobjekt (till exempel `Selection.Font`, `Selection.Paragraph` och `Selection.Border`). I den nya API-versionen finns formategenskaperna i varje objekt (`Paragraph`, `TextCursor` och så vidare) och kan användas direkt. En lista på tillgängliga tecken- och styckeegenskaper finns i följande avsnitt.

Teckenegenskaper

Formategenskaper som gäller enskilda tecken kallas teckenegenskaper. Exempel på teckenegenskaper är teckensnitt och fetstil. Objekt som kan hantera teckenegenskaper stöder tjänsten `com.sun.star.style.CharacterProperties`. StarOffice innehåller flera olika objekt som stöder denna tjänst. Dessa är bland annat `com.sun.star.text.Paragraph` för stycken och `com.sun.star.text.TextPortion` för styckedelar.

Tjänsten `com.sun.star.style.CharacterProperties` har inga gränssnitt, men innehåller egenskaper genom vilka teckenegenskaper kan definieras och anropas. En fullständig lista på alla teckenegenskaper finns i referensmaterialet till StarOffice API. I följande lista finns några av de vanligaste egenskaperna:

- **`CharFontName (String)`** – namnet på det aktuella teckensnittet.
- **`CharColor (Long)`** – teckenfärgen.
- **`CharHeight (Float)`** – teckenhöjden i punkter (pt).
- **`CharUnderline (konstant)`** – typ av understryckning (konstanterna enligt `com.sun.star.awt.FontUnderline`).
- **`CharWeight (konstant)`** – tjocklek (konstanter enligt `com.sun.star.awt.FontWeight`).
- **`CharBackColor (Long)`** – bakgrundsfärg.
- **`CharKeepTogether (Boolean)`** – blockering av automatiska radbrytningar.
- **`CharStyleName (String)`** – namn på teckenformatmall.

Styckeegenskaper

Formateringsinformation som gäller hela stycken i stället för enskilda tecken kallas styckeegenskaper. Exempel på styckeegenskaper är avståndet mellan stycket och papperskanten

samt radavstånd. Styckeegenskaperna finns i tjänsten `com.sun.star.style.ParagraphProperties`.

Styckeegenskaperna finns också i olika objekt. Alla objekt som stöder tjänsten `com.sun.star.text.Paragraph` stöder också styckeegenskaper i tjänsten `com.sun.star.style.ParagraphProperties`.

En fullständig lista på alla styckeegenskaper finns i referensmaterialet till StarOffice API. De vanligaste styckeegenskaperna är:

- **`ParaAdjust (enum)`** – vertikal textorientering (konstanter enligt `com.sun.star.style.ParagraphAdjust`).
- **`ParaLineSpacing (struktur)`** – radavstånd (strukturen enligt `com.sun.star.style.LineSpacing`).
- **`ParaBackColor (Long)`** – bakgrundsfärg.
- **`ParaLeftMargin (Long)`** – vänstermarginal i hundradels millimeter.
- **`ParaRightMargin (Long)`** – högermarginal i hundradels millimeter.
- **`ParaTopMargin (Long)`** – övre marginal i hundradels millimeter.
- **`ParaBottomMargin (Long)`** – nedre marginal i hundradels millimeter.
- **`ParaTabStops (Struct-vektor)`** – typ och placering av tabbar (strukturer med `com.sun.star.style.TabStop`).
- **`ParaStyleName (String)`** – namn på styckeformatmall.

Exempel: enkel HTML-export

Följande exempel visar hur du kan arbeta med formateringsinformation. Exemplet itererar genom ett textdokument och skapar en enkel HTML-fil. Varje stycke sparas med HTML-taggen <P>. Styckedelar med fet stil får HTML-taggen <B> vid exporten.

```
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

' gå igenom alla stycken
While Enum1.hasMoreElements
      TextElement = Enum1.nextElement
      If TextElement.supportsService("com.sun.star.text.Paragraph") Then
            Enum2 = TextElement.createEnumeration
            CurLine = "<P>"

            ' gå igenom alla styckedelar
            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

            ' mata ut raden
            CurLine = CurLine & "</P>"
            Print #FileNo, CurLine

      End If
Wend

' skriv HTML-sidfot
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

Den grundläggande strukturen i exemplet är densamma som i tidigare exempel för att gå igenom styckedelarna i en text. Funktionalitet för att skriva till HTML-filen, att kontrollera formatering för olika textdelar samt för att skapa motsvarande HTML-taggar har lagts till.

Standardvärden för tecken- och styckeegenskaper

Direkt formatering har alltid högre prioritet än ***indirekt*** formatering. Med andra ord har formatering som bygger på en formatmall alltid lägre prioritet än direkt formatering i texten.

Det finns inget enkelt sätt att i användargränssnittet avgöra om ett avsnitt i ett dokument har formaterats direkt eller indirekt. Symbollisterna i StarOffice visar vanliga textegenskaper som teckensnitt, teckenstil och storlek. Däremot visar symbollisterna inte om texten har formaterats direkt eller med en formatmall.

I StarOffice Basic finns metoden `getPropertyState` som kan användas för att kontrollera på vilket sätt en viss formatering använts. Metoden tar namnet på egenskapen som en parameter och returnerar en konstant som ger information om formateringens ursprung. Följande värden, som är definierade i uppräkningen `com.sun.star.beans.PropertyState`, är möjliga:

- **`com.sun.star.beans.PropertyState.DIRECT_VALUE`** – egenskapen är definierad direkt i texten (direkt formatering).
- **`com.sun.star.beans.PropertyState.DEFAULT_VALUE`** – egenskapen är definierad via en formatmall (indirekt formatering).
- **`com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE`** – egenskapen är tvetydigt definierad.
  Detta kan till exempel inträffa om egenskapen fetstil efterfrågas för ett stycke som både innehåller ord i fet stil och ord med normal tjocklek.

I följande exempel visas hur formateringsegenskaper kan redigeras med StarOffice. Exemplet söker igenom en text efter styckedelar som har formaterats direkt med fet stil. Om en motsvarande styckedel påträffas raderas den direkta formateringen med metoden `setPropertyToDefault` och använder formatmallen `MyBold` i motsvarande styckedel.

```
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

' gå igenom alla stycken
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration

              ' gå igenom alla styckedelar
              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
```

Redigera textdokument

I föregående avsnitt finns en beskrivning av olika alternativ för att redigera textdokument, med fokus på `com.sun.star.text.TextPortion` och `com.sun.star.text.Paragraph`, som ger åtkomst till styckedelar samt stycken. Dessa metoder gäller program där textinnehållet ska ändras globalt med en slinga. Det finns däremot många situationer där denna metod inte kan användas. I StarOffice finns tjänsten `com.sun.star.text.TextCursor` som kan användas för mer komplicerade situationer, till exempel när dokument måste genomsökas baklänges eller när enskilda ord eller meningar ska ändras i stället för `TextPortions`.

Textmarkören

Objektet `TextCursor` i StarOffice API motsvarar den synliga textmarkören i användargränssnittet i ett StarOffice-dokument. Markören markerar en viss punkt i ett textdokument och kan flyttas i alla riktningar med olika kommandon. De olika `TextCursor`-objekten i StarOffice Basic är däremot inte samma sak som den synliga markören. De är två helt skilda objekt.

Varning! Terminologin skiljer sig från den som används i VBA: i termer av definitionsområden motsvaras VBA-objektet `Range` av StarOffice-objektet `TextCursor` och inte av StarOffice-objektet `Range`.

Objektet `TextCursor` i StarOffice innehåller till exempel metoder för att navigera i och ändra text som finns i objektet `Range` i VBA (till exempel `MoveStart`, `MoveEnd`, `InsertBefore`, `InsertAfter`). Motsvarigheterna till objektet `TextCursor` i StarOffice finns beskrivna i följande avsnitt.

Navigera i texten

Objektet `TextCursor` i StarOffice Basic fungerar oberoende av den synliga markören i ett textdokument. En programmatisk ändring av positionen för objektet `TextCursor` påverkar inte placeringen av den synliga markören. Det kan finnas flera `TextCursor`-objekt på olika ställen samtidigt i samma dokument, och alla objekt är oberoende av varandra.

Ett `TextCursor`-objekt skapas med anropet `createTextCursor`:

```
Dim Doc As Object
Dim Cursor As Object

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

Objektet `Cursor` som skapas i exemplet stöder tjänsten `com.sun.star.text.TextCursor` som innehåller flera olika metoder för att navigera i textdokument. I följande exempel flyttas först objektet `TextCursor` tio tecken till vänster, och därefter tre tecken till höger:

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

Ett objekt av typen `TextCursor` kan markera ett helt område. Detta kan jämföras med att markera ett ställe i texten med musen. Parametern `False` i funktionsanropet ovan anger att texten inte ska markeras när markören flyttas. Till exempel flyttas objektet `TextCursor` i följande exempel

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

först tio tecken till höger utan att markera, och sedan tre tecken tillbaka med markering. Det område som markeras av `TextCursor` börjar efter det sjunde tecknet i texten och slutar efter det tionde tecknet.

Här följer de huvudsakliga metoder som tjänsten `com.sun.star.text.TextCursor` innehåller för att navigera:

- **`goLeft (Count, Expand)`** – går Count tecken till vänster.
- **`goRight (Count, Expand)`** – går Count tecken till höger.
- **`gotoStart (Expand)`** – går till början av textdokumentet.
- **`gotoEnd (Expand)`** – går till slutet av textdokumentet.
- **`gotoRange (TextRange, Expand)`** – går till ett givet `TextRange`-objekt.
- **`gotoStartOfWord (Expand)`** – går till början av aktuellt ord.
- **`gotoEndOfWord (Expand)`** – går till slutet av aktuellt ord.
- **`gotoNextWord (Expand)`** – går till början av nästa ord.
- **`gotoPreviousWord (Expand)`** – går till början av föregående ord.
- **`isStartOfWord ()`** – returnerar `True` om `TextCursor`-objektet står i början av ett ord.
- **`isEndOfWord ()`** – returnerar `True` om `TextCursor`-objektet står i slutet av ett ord.
- **`gotoStartOfSentence (Expand)`** – går till början av aktuell mening.
- **`gotoEndOfSentence (Expand)`** – går till slutet av aktuell mening.
- **`gotoNextSentence (Expand)`** – går till början av nästa mening.
- **`gotoPreviousSentence (Expand)`** – går till början av föregående mening.
- **`isStartOfSentence ()`** – returnerar `True` om `TextCursor`-objektet står i början av en mening.
- **`isEndOfSentence ()`** – returnerar `True` om `TextCursor`-objektet står i slutet av en mening.
- **`gotoStartOfParagraph (Expand)`** – går till början av aktuellt stycke.
- **`gotoEndOfParagraph (Expand)`** – går till slutet av aktuellt stycke.
- **`gotoNextParagraph (Expand)`** – går till början av nästa stycke.
- **`gotoPreviousParagraph (Expand)`** – går till början av föregående stycke.
- **`isStartOfParagraph ()`** – returnerar `True` om `TextCursor`-objektet står i början av ett stycke.

- **`isEndOfParagraph ()`** – returnerar `True` om `TextCursor`-objektet står i slutet av ett stycke.

Texten är uppdelad i meningar som avgränsas av skiljetecken. En punkt är till exempel en symbol som tolkas som en meningsavgränsare.

Parametern `Expand` är ett värde av typen Boolean som anger om det område som markören förflyttar sig över ska markeras eller inte. Alla navigeringsmetoder returnerar ett värde som anger om navigeringen lyckades eller om det inte finns tillräckligt mycket text för att slutföra navigeringen.

I följande lista finns flera olika metoder för att redigera markerade områden med ett `TextCursor`-objekt som också stöder tjänsten `com.sun.star.text.TextCursor`:

- **`collapseToStart ()`** – nollställer markeringen placerar `TextCursor`-objektet i början av det tidigare markerade området.
- **`collapseToEnd ()`** – nollställer markeringen placerar `TextCursor` -objektet i slutet av det tidigare markerade området.
- **`isCollapsed ()`** – returnerar `True` om `TextCursor`-objektet inte innehåller en markering.

Formatera text med ett TextCursor-objekt

Servicen `com.sun.star.text.TextCursor` stöder alla tecken- och styckeegenskaper som beskrevs i början av detta kapitel.

I följande exempel visas hur dessa egenskaper kan användas i samband med `TextCursor`. Exempelkoden går igenom hela dokumentet och formaterar det första ordet i varje mening med fet stil.

```
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
```

I exemplet skapas först ett dokumentobjekt för den text som just har öppnats. Exemplet går sedan igenom hela texten, mening för mening, markerar det första ordet i varje mening och gör texten fet.

Hämta och ändra textinnehåll

Om ett `TextCursor`-objekt innehåller en markering, representeras texten av egenskapen `String` i `TextCursor`-objektet. I följande exempel används egenskapen `String` för att visa de första orden i en mening i en meddelanderuta:

```
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
```

Det första ordet i varje mening kan ändras på samma sätt med egenskapen `String`:

```
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
```

Om `TextCursor`-objektet innehåller en markering, innebär en tilldelning till egenskapen `String` att textinnehållet ersätts. Om det inte finns en markering, infogas texten vid den aktuella `TextCursor`-positionen.

Infoga styrkoder

I vissa situationer är det dokumentets struktur, och inte själva texten, som behöver ändras. I StarOffice används styrkoder för detta ändamål. Styrkoderna infogas i texten och påverkar dokumentets struktur. De olika styrkoderna är definierade i konstantgruppen `com.sun.star.text.ControlCharacter` . Följande styrkoder finns definierade i StarOffice:

- **`PARAGRAPH_BREAK`** – styckebrytning.
- **`LINE_BREAK`** – radbrytning i ett stycke.
- **`SOFT_HYPHEN`** – möjlig plats för avstavning.
- **`HARD_HYPHEN`** – obligatorisk plats för avstavning.
- **`HARD_SPACE`** – fast mellanslag som inte expanderas eller komprimeras i justerad text.

Om du behöver infoga styrkoder behöver du inte bara en textmarkör, utan även det associerade textdokumentobjektet. I följande exempel infogas ett stycke efter det 20:e tecknet i texten:

```
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)
```

Parametern `False` i anropet till metoden `insertControlCharacter` anger att området som markeras av `TextCursor`-objektet finns kvar efter att stycket infogats. Om parametern `True` används här, kommer `insertControlCharacter` att ersätta den aktuella texten.

Söka efter textavsnitt

I många situationer ska koden söka efter en viss fras och redigera texten på det ställe frasen finns. I alla StarOffice-dokument finns ett särskilt gränssnitt för detta ändamål, och gränssnittet fungerar alltid på samma sätt: Innan sökningen måste en så kallad `SearchDescriptor` skapas. Objektet definierar vad StarOffice ska söka efter i dokumentet. Ett objekt av typen `SearchDescriptor` är ett objekt som stöder tjänsten `com.sun.star.util.SearchDescriptor` och skapas med dokumentmetoden `createSearchDescriptor`:

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

När en `SearchDescriptor` har skapats anges den text som programmet ska söka efter:

```
SearchDesc.searchString="valfri text"
```

I termer av funktionalitet motsvarar `SearchDescriptor` närmast sökdialogrutan i StarOffice. De olika sökalternativen du anger i sökfönstret anger du i stället i `SearchDescriptor`-objektet.

Egenskaperna definieras av tjänsten `com.sun.star.util.SearchDescriptor`:

- **`SearchBackwards (Boolean)`** – söker baklänges genom texten.
- **`SearchCaseSensitive (Boolean)`** – skiljer på stora och små bokstäver under sökningen.
- **`SearchRegularExpression (Boolean)`** – tolkar söksträngen som ett reguljärt uttryck.
- **`SearchStyles (Boolean)`** – söker igenom texten efter en viss styckeformatmall.
- **`SearchWords (Boolean)`** – söker endast efter hela ord.

StarOffice-funktionen `SearchSimilarity` (eller ”likhetssökning”) finns även i StarOffice Basic. Med den här funktionen söker StarOffice efter ett uttryck som ungefär stämmer överens med sökuttrycket. Hur många tecken som kan skilja mellan de två uttrycken kan anges individuellt (för saknade, ändrade respektive ytterligare tecken). Tjänsten `com.sun.star.util.SearchDescriptor` innehåller följande egenskaper:

- **`SearchSimilarity (Boolean)`** – utför en likhetssökning.
- **`SearchSimilarityAdd (Short)`** – antal tecken som får läggas till vid en likhetssökning.
- **`SearchSimilarityExchange (Short)`** – antal tecken som får ersättas vid en likhetssökning.
- **`SearchSimilarityRemove (Short)`** – antal tecken som får tas bort vid en likhetssökning.
- **`SearchSimilarityRelax (Boolean)`** – tar hänsyn till alla avvikelser samtidigt under sökningen.

När en `SearchDescriptor` har definierats kan objektet användas i dokumentet. StarOffice-dokument innehåller metoderna `findFirst` och `findNext` för detta ändamål:

```
Found = Doc.findFirst (SearchDesc)

Do While Found
      ' Sök efter uttrycket
      Found = Doc.findNext( Found.End, Search)
Loop
```

I exemplet används en slinga för att söka efter alla förekomster av uttrycket. Sökningen returnerar ett `TextRange`-objekt som refererar till den funna texten.

Exempel: Likhetssökning

Detta exempel söker igenom dokumentet efter ordet "omsättning" och formaterar varje förekomst med fet stil. Exemplet använder en likhetssökning så att, förutom ordet ”omsättning”, hittas även pluralformen "omsättningar" och den bestämda formen "omsättningen". De matchande uttrycken kan skilja sig med upp till två bokstäver från sökuttrycket.

```
Dim SearchDesc As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

SearchDesc = Doc.createSearchDescriptor
SearchDesc.SearchString="omsättning"
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
```

Den grundläggande uppbyggnaden av sök- och ersättfunktionaliteten i StarOffice är jämförbar med motsvarande funktionalitet i VBA. Båda gränssnitten använder ett objekt, vars egenskaper definierar hur sökningen ska utföras. Detta objekt används sedan i textområdet för att utföra sökningen. Hjälpobjektet i VBA nås via egenskapen `Find` i `Range`-objektet, och i StarOffice Basic skapas objektet med metoden `createSearchDescriptor` eller `createReplaceDescriptor` i dokumentobjektet. Dessutom skiljer sig egenskaperna och metoderna för sökobjektet.

Liksom i det gamla StarOffice API, bygger sök- och ersättfunktionaliteten i denna API-version på dokumentobjektet. Tidigare fanns objektet `SearchSettings` som användes för att definiera sökalternativ. I denna version används en `SearchDescriptor` eller `ReplaceDescriptor` för att automatiskt ersätta text. Dessa objekt definierar inte bara alternativen, men även det aktuella sökuttrycket och eventuell ersättningstext. Objekt skapas från dokumentobjektet, objektets egenskaper fylls i, och sedan används objektet igen på dokumentet som en parameter till sökfunktioner.

Ersätta text

Precis som med sökfunktionen, finns ersättfunktionen i StarOffice även i StarOffice Basic. De två funktionerna fungerar på samma sätt. Ett särskilt objekt som definierar ersättningens parametrar skapas innan ersättningen görs. Objektet är av typen `ReplaceDescriptor` och stöder tjänsten `com.sun.star.util.ReplaceDescriptor`. Alla egenskaper i `SearchDescriptor` ovan stöds även av `ReplaceDescriptor`. Till exempel kan en ersättning vara skiftlägeskänslig eller bygga på likhetssökningar.

I följande exempel visas hur en `ReplaceDescriptor` används för att söka i ett StarOffice-dokument.

```
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
```

Uttrycken för att söka och ersätta anges med egenskaperna `SearchString` och `ReplaceString` i `ReplaceDescriptor`. Själva ersättningen utförs med anropet till metoden `replaceAll` i dokumentobjektet, som ersätter alla förekomster av sökuttrycket.

Exempel: söka och ersätta text med reguljära uttryck

Ersättningsfunktionen i StarOffice är särskilt effektiv när den används tillsammans med reguljära uttryck. Reguljära uttryck är en form av variabla sökuttryck som använder platshållare och specialtecken i stället för konstantvärden.

De reguljära uttryck som stöds av StarOffice beskrivs mer detaljerat i direkthjälpen för StarOffice. Här följer ett par exempel:

- En punkt i ett sökuttryck representerar ett godtyckligt tecken. Sökuttrycket `k.rt` kan därför både stå för `kort` och för `kört`.
- Tecknet `^` markerar början av ett stycke. Alla förekomster av namnet `Peter` som finns i början av ett stycke kan hittas med sökuttrycket `^Peter`.

- Tecknet `$` markerar slutet på ett stycke. Alla förekomster av namnet `Peter` som finns i slutet av ett stycke kan hittas med sökuttrycket `Peter$`.
- Ett `*` innebär att det föregående tecknet kan upprepas flera gånger. Det kan kombineras med punkten som platshållare för vilket tecken som helst. Uttrycket `be.*ning` kan till exempel stå för uttrycken `befattning` och `behållning`.

I följande exempel visas hur alla tomma rader i ett textdokument kan tas bort med hjälp av ett reguljärt uttryck `^$`:

```
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)
```

Textdokument är inte bara text

Hittills har detta kapitel endast behandlat textstycken och deras innehåll. Men textdokument kan också innehålla andra objekt. Dessa objekt kan till exempel vara tabeller, teckningar, fältkommandon och kataloger. Alla dessa objekt kan förankras på en valfri plats i texten.

De gemensamma egenskaperna för alla objekt i StarOffice finns definierade i tjänsten `com.sun.star.text.TextContent`. Tjänsten innehåller följande egenskaper:

- **`AnchorType (Enum)`** – anger förankringstypen för ett `TextContent`-objekt (standardvärden enligt `com.sun.star.text.TextContentAnchorType` , uppräkning).
- **`AnchorTypes (sequence of Enum)`** – uppräkning av alla `AnchorTypes` som stöder ett visst `TextContent`-objekt.
- **`TextWrap (Enum)`** – anger brytningen kring ett `TextContent`-objekt (standardvärden enligt `com.sun.star.text.WrapTextMode` , uppräkning).

De olika `TextContent`-objekten har också vissa gemensamma metoder för att skapa, infoga och ta bort objekt.

- Ett nytt `TextContent`-objekt kan **skapas** med metoden `createInstance` i dokumentobjektet.
- Ett objekt **infogas** med metoden `insertTextContent` i dokumentobjektet.
- `TextContent`-objekt kan **tas bort** med metoden `removeTextContent`.

I följande avsnitt finns flera olika exempel som använder dessa metoder.

Tabeller

I följande exempel skapas en tabell med metoden `createInstance` som beskrivs ovan.

```
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)
```

När tabellen skapats anges antalet rader och kolumner med metoden `initialize`. Tabellen placeras sedan i textdokumentet med `insertTextContent`.

Som framgår av exemplet, tar metoden `insertTextContent` inte bara det `Content`-objekt som ska infogas, men dessutom två andra parametrar:

- ett `Cursor`-objekt som definierar var objektet ska infogas
- en variabel av typen Boolean som anger som `Content`-objektet ska ersätta den aktuella markeringen (`True`) eller om det ska infogas före den aktuella markeringen i texten (`False`)

När tabeller skapas och infogas i ett textdokument, fungerar objektet på ungefär samma sätt i StarOffice som i VBA. Dokumentobjektet och ett `TextCursor`-objekt i StarOffice Basic motsvaras av objektet `Range` i VBA. I VBA används metoden `Document.Tables.Add` för att skapa och definiera tabellen. I StarOffice Basic används i stället `createInstance` för att skapa och initiera tabellen, som sedan infogas i dokumentet med `insertTextContent`.

Vilka tabeller som finns i ett textdokumentet kan kontrolleras med en enkel slinga. Metoden `getTextTables()` i textdokumentet används för detta ändamål:

```
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)

      ' Redigera tabellen
Next I
```

Texttabeller finns tillgängliga i StarOffice 7 via listan `TextTables` i dokumentobjektet. Detta ersätter den tidigare tabellistan i objektet `Selection`. I det föregående exemplet visas hur en texttabell kan skapas. Alternativen för att hämta texttabeller beskrivs i följande avsnitt.

Redigera tabeller

En tabell består av rader. Raderna innehåller i sin tur celler. I formella termer finns det inga tabellkolumner i StarOffice. Kolumnerna uppstår implicit när raderna placeras under varandra. För att förenkla tabellredigering finns vissa metoder i StarOffice som bygger på kolumner. Dessa metoder kan användas om det inte finns sammanfogade celler i tabellen.

Tabellen innehåller flera olika egenskaper som definieras av tjänsten `com.sun.star.text.TextTable`. Här följer en lista av tabellobjektets vanligaste egenskaper:

- **`BackColor (Long)`** – tabellens bakgrundsfärg.
- **`BottomMargin (Long)`** – nedre marginal i hundradels millimeter.
- **`LeftMargin (Long)`** – vänstermarginal i hundradels millimeter.
- **`RightMargin (Long)`** – högermarginal i hundradels millimeter.
- **`TopMargin (Long)`** – övre marginal i hundradels millimeter.
- **`RepeatHeadline (Boolean)`** – tabellöverskriften upprepas på varje sida.
- **`Width (Long)`** – tabellens bredd i hundradels millimeter.

Rows

En tabell består av en lista med rader. I följande exempel visas hur raderna i en tabell kan hämtas och formateras.

```
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
```

I exemplet skapas först en lista med alla rader med metoden `Table.getRows`. Metoderna `getCount` och `getByIndex` hämtar tabellens innehåll och finns i gränssnittet `com.sun.star.table.XtableRows`. Metoden `getByIndex` returnerar ett radobjekt, som stöder tjänsten `com.sun.star.text.TextTableRow`

Här följer de huvudsakliga metoder som gränssnittet `com.sun.star.table.XtableRows` innehåller:

- **`getByIndex(Integer)`** – returnerar ett radobjekt för ett givet index.
- **`getCount()`** – returnerar antalet radobjekt.
- **`insertByIndex(Index, Count)`** – infogar Count antal rader i tabellen på positionen`Index`.
- **`removeByIndex(Index, Count)`** – tar bort Count antal rader från tabellen på positionen `Index`.

Även om metoderna `getByIndex` och `getCount` är tillgängliga i alla tabeller, kan metoderna `insertByIndex` och `removeByIndex` endast användas i tabeller som inte innehåller sammafogade celler.

Tjänsten `com.sun.star.text.TextTableRow` innehåller följande egenskaper:

- **`BackColor (Long)`** – radens bakgrundsfärg.
- **`Height (Long)`** – radhöjden i hundradels millimeter.
- **`IsAutoHeight (Boolean)`** – tabellhöjden anpassas automatiskt efter innehållet.
- **`VertOrient (konstant)`** – vertikal orientering av textramen, definierar vertikal orientering av texten i tabellen (värden i enlighet med `com.sun.star.text.VertOrientation`)

Kolumner

Kolumner hämtas på samma sätt som rader, med metoderna `getByIndex`, `getCount`, `insertByIndex` och `removeByIndex` i objektet `Column` som nås via `getColumns`. Kolumnerna kan däremot endast användas i tabeller som inte innehåller sammafogade celler. Det går inte att formatera celler efter kolumn i StarOffice Basic. För att formatera celler måste du formatera de enskilda cellerna separat.

Celler

Varje cell i ett StarOffice-dokument har ett unikt namn. Om StarOffice-markören står i en cell visas cellens namn i statuslisten. Cellen högst upp till vänster har oftast namnet `A1` och cellen längst ned till höger kallas oftast `Xn`, där `X` står för bokstaven för den sista kolumnen och `n` för antalet rader. Cellobjekten finns tillgängliga via metoden `getCellByName()` i tabellobjektet. I följande exempel används en slinga för att gå igenom alla celler i en tabell och skriva in motsvarande rad- och kolumnnamn i cellerna.

```
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 = "row: " & CStr(RowIndex) + ", column: " & CStr(ColIndex)
      Next
Next
```

En tabellcell är jämförbar med vanlig text. Cellen stöder gränssnittet `createTextCursor` för att skapa ett `TextCursor`-objekt.

```
CellCursor = Cell.createTextCursor()
```

Alla formateringsalternativ för enskilda tecken och stycken finns därför också direkt tillgängliga.

I följande exempel genomsöks alla tabeller i ett textdokument, och högerjusterar alla celler med numeriska värden genom att en motsvarande styckeformategenskap används.

```
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
```

I exemplet skapas en `TextTables`-lista som innehåller alla tabeller i texten. StarOffice skapar sedan en lista med motsvarande cellnamn för varje tabell. Dessa gås igenom i en slinga. Om en cell innehåller ett numeriskt värde ändras sedan formateringen. För att ändra formateringen skapas ett `TextCursor`-objekt som refererar till innehållet i tabellcellen, och som sedan får styckeformategenskaperna för tabellcellen.

Textramar

Textramar är `TextContent`-objekt, precis som tabeller och grafik. Ramarna kan innehålla vanlig text, men kan placeras var som helst på sidan och ingår inte i textflödet.

Liksom för övriga `TextContent`-objekt, är det en skillnad mellan att skapa textramarna och på att infoga dem i dokumentet.

```
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)
```

Textramen skapas med metoden `createInstance` i dokumentobjektet. Textramen kan sedan infogas med metoden `insertTextContent` i objektet `Text`. För att göra det måste rätt `com.sun.star.text.TextFrame`-tjänst anges.

Textramens placering anges av ett `Cursor`-objekt, som också används när ramen ska infogas.

Textramar i StarOffice motsvaras av de ramar som används i Word. I VBA används metoden `Document.Frames.Add` för att skapa ramar. I StarOffice används proceduren ovan med en TextCursor samt metoden `createInstance` i dokumentobjektet.

Textramar innehåller egenskaper som anger ramens placering och funktionalitet. De flesta av dessa egenskaper definieras i tjänsten `com.sun.star.text.BaseFrameProperties`, som också stöds av varje TextFrame-tjänst. De vanligaste egenskaperna är:

- **`BackColor (Long)`** – textramens bakgrundsfärg.
- **`BottomMargin (Long)`** – nedre marginal i hundradels millimeter.
- **`LeftMargin (Long)`** – vänstermarginal i hundradels millimeter.
- **`RightMargin (Long)`** – högermarginal i hundradels millimeter.
- **`TopMargin (Long)`** – övre marginal i hundradels millimeter.
- **`Height (Long)`** – textramens höjd i hundradels millimeter.
- **`Width (Long)`** – textramens bredd i hundradels millimeter.
- **`HoriOrient (konstant)`** – horisontell orientering av textramen (enligt `com.sun.star.text.HoriOrientation`).
- **`VertOrient (konstant)`** – vertikal orientering av textramen (enligt `com.sun.star.text.VertOrientation`).

I följande exempel skapas en textram med ovanstående egenskaper:

```
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)
```

I exemplet skapas en `TextCursor` som används som insättningspunkt för textramen. Objektet placeras mellan det första och andra ordet i texten. Textramen skapas med metoden `Doc.createInstance`. Därefter anges textramens egenskaper.

Lägg märke till sambandet mellan `AnchorType` (från tjänsten `TextContent`) och `VertOrient` (från tjänsten `BaseFrameProperties`). `AnchorType` antar värdet `AS_CHARACTER`. Därför infogas textramen direkt i textflödet och fungerar som ett vanligt tecken. Den kan till exempel flyttas till nästa rad om en radbrytning medför det. Värdet `LINE_TOP` för egenskapen `VertOrient` anger att textramens övre kant är på samma nivå som den övre kanten av tecknet.

När objektet initierats, infogas textramen i dokumentet med ett anrop till `insertTextContent`.

För att redigera innehållet i en textram används en `TextCursor`, som även fungerar för textramar.

```
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 = "Det här är ett test!"
```

Exemplet skapar en textram, infogar den i det aktuella dokumentet och öppnar en `TextCursor` för textramen. Markören används för att ange teckensnittet till fet stil och för att centrera stycket. Därefter anges texten ”Detta är ett test!” för textramen.

Fältkommandon

Fältkommandon är `TextContent`-objekt eftersom de innehåller mer funktionalitet än vanlig text. Fältkommandon kan infogas i textdokument med samma metoder som används för andra `TextContent`-objekt:

```
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)
```

I exemplet skapas ett fältkommando med dagens datum i början av det aktuella textdokumentet. Värdet `True` för egenskapen `IsDate` innebär att endast datumet visas, och inte klockslaget. Värdet `False` för egenskapen `IsFixed` gör att datumet uppdateras automatiskt när dokumentet öppnas.

I VBA anges fältets typ av en parameter till metoden `Document.Fields.Add`. I StarOffice Basic anges fältets typ av den tjänst som skapar fältkommandot.

I tidigare versioner fanns flera olika metoder för att komma åt fältkommandon med objektet `Selection` (till exempel `InsertField`, `DeleteUserField`, `SetCurField`).

I StarOffice 7 administreras fältet med en objektorienterad metod. När du ska skapa ett fältkommando skapar du ett fält av rätt typ och initierar fältet med olika egenskaper. Fältkommandot infogas sedan i dokumentet med metoden `insertTextContent`. I exemplet ovan används denna metod. I följande avsnitt beskrivs de vanligaste fälttyperna och deras respektive egenskaper.

Förutom att lägga till fältkommandon, kan du även söka igenom ett dokument efter fältkommandon. I följande exempel visas hur alla fältkommandon i ett textdokument gås igenom i en slinga, och hur varje fälts typ kontrolleras.

```
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 "Tid/datum"
	ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
		MsgBox "Anmärkning"
	Else
		MsgBox "Okänt"
	End If

Wend
```

Utgångspunkten för att hitta fältkommandona är dokumentobjektets `TextFields`-lista. I exemplet skapas ett `Enumeration`-objekt för denna lista, som kan användas för att räkna upp alla fält i en slinga. Vilken typ av fältkommando som hittas kontrolleras genom metoden `supportsService`. Om fältkommandot är ett tid / datum-fält eller en anmärkning, visas motsvarande fälttyp i en meddelanderutan. Om fältkommandot är av en annan typ visas meddelandet ”Okänt”.

Nedan finns en lista på de vanligaste fältkommandona och motsvarande egenskaper. En fullständig lista över alla fältkommandon finns i referensmaterialet i modulen `com.sun.star.text.TextField`. (När du anger namnet på en tjänst måste du skilja mellan stora och små bokstäver, enligt exemplet ovan.)

Antal sidor, ord och tecken

Fältkommandona

- `com.sun.star.text.TextField.PageCount`
- `com.sun.star.text.TextField.WordCount`
- `com.sun.star.text.TextField.CharacterCount`

returnerar antalet sidor, ord respektive tecken i en text. De stöder följande egenskap:

- **`NumberingType (konstant)`** – numreringsformat (i enlighet med konstanter i `com.sun.star.style.NumberingType`).

Aktuell sida

Den aktuella sidans nummer kan infogas i dokumentet med fältkommandot `com.sun.star.text.TextField.PageNumber`. Följande egenskaper kan anges:

- **`NumberingType (konstant)`** – numreringsformat (i enlighet med konstanter i `com.sun.star.style.NumberingType`).
- **`Offset (short)`** – värde som ska läggas till det totala antalet sidor (kan även vara ett negativt värde).

I följande exempel visas hur sidantalet kan infogas i dokumentets sidfot.

```
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("Sidformatmallar")

StdPage = PageStyles("Default")
StdPage.FooterIsOn = True

FooterCursor = StdPage.FooterTextLeft.Text.createTextCursor()
StdPage.FooterTextLeft.Text.insertTextContent(FooterCursor, PageNumber, False)
```

I exemplet skapas först ett fältkommando som stöder tjänsten `com.sun.star.text.TextField.PageNumber`. Eftersom sidhuvud- och sidfotsrader definieras som en del i sidformatmallar i StarOffice, hämtas standardformatmallen från listan som innehåller alla `PageStyles`.

För att göra sidfotsraden synlig anges egenskapen `FooterIsOn` till `True`. Fältkommandot infogas sedan i dokumentet med det textobjekt som hör till sidfotsraden till vänster.

Anmärkningar

Anmärkningsfält (`com.sun.star.text.TextField.Annotation`) visas med en mindre gul symbol i texten. Om du klickar på symbolen öppnas ett textfält där du kan skriva en kommentar om texten. Ett anmärkningsfält har följande egenskaper.

- **`Author (String)`** – författarens namn.
- **`Content (String)`** – kommentarssträng.
- **`Date (Date)`** – datum när kommentaren infördes.

Tid och klockslag

Ett datum- eller tidsfält (`com.sun.star.text.TextField.DateTime`) representerar dagens datum eller aktuell tid. Det stöder följande egenskaper:

- **`IsFixed (Boolean)`** – om `True`, ändras tidsvärdet aldrig, om False uppdateras värdet varje gång dokumentet öppnas.
- **`IsDate (Boolean)`** – om `True`, visar fältet dagens datum, annars visas den aktuella tiden.
- **`DateTimeValue (struktur)`** – aktuellt innehåll i fältet (`com.sun.star.util.DateTime` struktur)
- **`NumberFormat (konstant)`** – det format i vilket klockslaget eller datumet visas.

Kapitelnamn / nummer

Namnet på det aktuella kapitlet är tillgängligt via ett fältkommando av typen `com.sun.star.text.TextField.Chapter`. Formatet definieras av två egenskaper.

- **`ChapterFormat (konstant)`** – anger om kapitelnamnet eller numret visas (i enlighet med `com.sun.star.text.ChapterFormat`)
- **`Level (Integer)`** – anger den kapitelnivå för vilken namnet och / eller numret ska visas. Värdet 0 står för den högsta tillgängliga nivån.

Bokmärken

Bokmärken (tjänsten `com.sun.star.text.Bookmark`) är `TextContent`-objekt. Bokmärken skapas och infogas på samma sätt som beskrivits ovan:

```
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 = "Mina bokmärken"
Doc.Text.insertTextContent(Cursor, Bookmark, True)
```

I exemplet skapas objektet `Cursor`, som markerar den position där bokmärket ska infogas, och sedan själva bokmärkesobjektet (`Bookmark`). Bokmärket tilldelas sedan ett namn och infogas i dokumentet med `insertTextContent` vid insättningspunkten.

Bokmärken i en text kan nås via listan `Bookmarks`. Bokmärkena kan antingen nås via nummer eller namn.

I följande exempel visas hur ett bokmärke kan lokaliseras i texten, varpå text infogas vid bokmärkets position.

```
Dim Doc As Object
Dim Bookmark As Object
Dim Cursor As Object

Doc = StarDesktop.CurrentComponent

Bookmark = Doc.Bookmarks.getByName("Mina bokmärken")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Här är bokmärket"
```

I detta fall används metoden `getByName` för att hitta bokmärket givet bokmärkets namn. Anropet `createTextCursorByRange` skapar sedan objektet `Cursor`, som placeras vid bokmärkets förankringspunkt. Markören infogar sedan texten på detta ställe.

KAPITEL 7

Tabelldokument

I StarOffice Basic finns ett omfattande gränssnitt för programmatisk styrning av tabelldokument. I detta kapitel beskrivs hur du kan använda gränssnitten, metoderna och egenskaperna som hör till tabelldokument.

I det första avsnittet behandlas den grundläggande strukturen i tabelldokument, och visar hur du når och redigerar innehållet i enskilda celler.

Det andra avsnittet behandlar redigering av tabelldokument med fokus på cellområden och alternativ för att söka och ersätta cellinnehåll.

Objektet Range används för att adressera tabellområden och har lagts till i denna API-version.

Strukturen för tabellbaserade dokument

Dokumentobjektet för ett tabelldokument bygger på tjänsten `com.sun.star.sheet.SpreadsheetDocument`. Varje sådant dokument kan innehålla flera tabeller. I denna handbok används termerna ***tabellbaserat dokument*** eller ***tabelldokument*** för hela dokumentet, och termen ***tabelldokument*** (eller ***tabell***) är en enskild tabell i dokumentet.

Notera skillnaden i terminologi mellan VBA och StarOffice Basic. Dokumentobjektet i VBA kallas *Workbook* och de enskilda sidorna *Worksheets*. I StarOffice Basic kallas de *SpreadsheetDocument* och *Sheet*.

Tabeller

Du kan nå de olika tabellerna i ett tabelldokument med listan `Sheets`.

I följande exempel visas hur du når en tabell via tabellens nummer eller namn.

Exempel 1: via nummer (numreringen börjar på 0)

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet  = Doc. Sheets (0)
```

Exempel 2: via namn

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Tabell 1")
```

I det första exemplet nås tabellen via tabellens nummer (numreringen börjar på 0). I det andra exemplet används tabellens namn och metoden `getByName`.

Objektet `Sheet` som returneras av metoden `getByName` stöder tjänsten `com.sun.star.sheet.Spreadsheet`. Förutom ett flertal gränssnitt för att redigera innehållet, innehåller denna tjänst följande egenskaper:

- **`IsVisible (Boolean)`** – om tabellen är synlig.
- **`PageStyle (String)`** – namnet på den sidformatmall som används för tabellen.

Skapa, ta bort och ändra namn på tabeller

Listan `Sheets` för ett t`abell` dokument används också för att skapa, ta bort och ändra namn på enskilda tabeller. I följande exempel används metoden `hasByName` för att kontrollera om tabellen ***MinTabell*** existerar. Gör den det skapas en motsvarande objektreferens med metoden `getByName` varpå referensen sparas i variabeln `Sheet`. Om tabellen inte finns skapas den med anropet `createInstance` och infogas i tabelldokumentet med metoden `insertByName`.

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

If Doc.Sheets.hasByName("MinTabell") Then
      Sheet = Doc.Sheets.getByName("MinTabell")
Else
      Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
      Doc.Sheets.insertByName("MinTabell", Sheet)
End If
```

Metoderna `getByName` och `insertByName` kommer från gränssnittet `com.sun.star.container.XnameContainer` och beskrivs i kapitel 4.

Rader och kolumner

Varje tabell innehåller en lista på rader och kolumner. Dessa finns tillgängliga via egenskaperna `Rows` och `Columns` i tabellobjektet och stöder tjänsterna `com.sun.star.table.TableColumns` och / eller `com.sun.star.table.TableRows`.

I följande exempel skapas två objekt som refererar till den första raden och den första kolumnen i en tabell, och lagrar referenserna i objektvariablerna `FirstCol` och `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)
```

Kolumnobjektet stöder tjänsten `com.sun.star.table.TableColumn` som har följande egenskaper:

- **`Width (long)`** – kolumnbredden i hundradels millimeter.
- **`OptimalWidth (Boolean)`** – anger optimal kolumnbredd.
- **`IsVisible (Boolean)`** – visar en kolumn.
- **`IsStartOfNewPage (Boolean)`** – skapar en sidbrytning före kolumnen vid utskrift.

Kolumnens bredd optimeras endast om egenskapen `OptimalWidth` är `True`. Om en enskild cells bredd ändras, ändras inte bredden för den kolumn som innehåller cellen. I funktionalitetstermer är `OptimalWidth` snarare en metod än en egenskap.

Radobjekten bygger på tjänsten `com.sun.star.table.RowColumn` som har följande egenskaper:

- **`Height (long)`** – radhöjden i hundradels millimeter.
- **`OptimalHeight (Boolean)`** – anger optimal radhöjd.
- **`IsVisible (Boolean)`** – visar raden.
- **`IsStartOfNewPage (Boolean)`** – skapar en sidbrytning före raden vid utskrift.

Om egenskapen `OptimalHeight` för en rad är `True`, ändras radhöjden automatiskt när cellhöjden i en cell ändras. Den automatiska optimeringen kommer att vara aktiv tills raden får en höjd med egenskapen `Height`.

I följande exempel aktiveras automatisk radhöjd för de första fem raderna i tabellen, och döljer den andra kolumnen.

```
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
```

Listorna `Rows` och `Columns` kan nås via index i StarOffice Basic. Till skillnad från VBA har den första kolumnen index 0 (inte index 1).

Infoga och ta bort rader och kolumner

Objekten `Rows` och `Columns` i en tabell kan användas för att nå befintliga rader och kolumner, samt för att infoga och ta bort dem.

```
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)
```

I detta exempel används metoden `insertByIndex` för att infoga en ny kolumn på positionen för den fjärde kolumnen i tabellen (index 3, numreringen börjar på 0). Den andra parametern anger hur många kolumner som ska läggas till (i exemplet en kolum).

Metoden `removeByIndex` tar bort den sjätte kolumnen (index 5). Den andra parametern anger hur många kolumner som ska tas bort.

Metoderna för att infoga och ta bort rader använder objektet `Rows` på samma sätt som vid redigering av kolumnen med `Columns`-objektet.

Celler

En tabell består av en tvådimensionell matris med celler. Varje cell definieras av en X- och Y-position relativt den översta cellen längst till vänster, som har position (0,0).

I följande exempel skapas ett objekt som refererar till den översta cellen längst till vänster, varpå text infogas i cellen:

```
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"
```

Förutom numeriska koordinater har varje cell i tabellen ett namn. Cellen överst längst till vänster (0,0) har namnet `A1`. Bokstaven `A` står för cellens kolumn, och siffran `1` står för raden. Lägg märke till att en cells ***namn*** och ***position*** skiljer sig åt med avseende på indexering; namnen börjar på 1 och positionerna börjar på 0.

I StarOffice kan en tabellcell vara tom eller innehålla text, tal eller formler. Cellens typ avgörs inte av innehållet i cellen, utan av vilken objektegenskap som användes för att ge cellen ett innehåll. Tal kan infogas och hämtas med egenskapen `Value`, text med egenskapen `String` och formler med egenskapen `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"
```

I exemplet infogas ett tal, en sträng och en formel i cellerna A1 till A3.

Egenskaperna `Value`, `String` och `Formula` ersätter metoden `PutCell` för att ange cellers värden.

StarOffice behandlar cellinnehåll som angetts med egenskapen `String` som text, även om strängen innehåller en siffra. Tal som anges på detta sätt vänsterjusteras i cellen, i stället för att högerjusteras. Lägg också märke till skillnaderna mellan text och tal när du använder formler:

```
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
```

Även om cell A1 innehåller värdet 100 och cell A2 innehåller värdet 1000, returnerar formeln A1+A2 värdet 100. Detta beror på att innehållet i cell A2 angetts som en sträng och inte som ett tal.

Om du vill kontrollera om en cell innehåller ett tal eller en sträng, använder du egenskapen `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 "Innehåll: Tom"
Case com.sun.star.table.CellContentType.VALUE
      MsgBox "Innehåll: Värde"
Case com.sun.star.table.CellContentType.TEXT
      MsgBox "Innehåll: Text"
Case com.sun.star.table.CellContentType.FORMULA
      MsgBox "Innehåll: Formel"
End Select
```

Egenskapen `Cell.Type` returnerar ett värde ur uppräkningen `com.sun.star.table.CellContentType` som identifierar cellinnehållets typ. Möjliga värden är:

- **`EMPTY`** – inget värde
- **`VALUE`** – tal
- **`TEXT`** – sträng
- **`FORMULA`** – formel

Infoga, ta bort, kopiera och flytta celler

Förutom att direkt ändra cellinnehåll, innehåller StarOffice Calc ett gränssnitt som du kan använda för att infoga, ta bort, kopiera eller sammanfoga celler. Gränssnittet (`com.sun.star.sheet.XRangeMovement`) finns tillgängligt via tabellobjektet och innehåller fyra metoder för att ändra innehållet i celler.

Metoden `insertCell` används för att infoga celler i en tabell.

```
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)
```

I exemplet infogas ett cellområde som är två rader gånger två kolumner stort vid platsen för den andra kolumnen och raden (båda har index 1) i den första tabellen (index 0) i tabelldokumentet. Alla befintliga värden i cellområdet flyttas nedåt.

Om du vill definiera ett cellområde som ska infogas använder du strukturen `com.sun.star.table.CellRangeAddress`. Följande värden finns i denna stuktur:

- **`Sheet (short)`** – tabellens nummer (numreringen börjar på 0).
- **`StartColumn (long)`** – första kolumnen i cellområdet (numreringen börjar på 0).
- **`StartRow (long)`** – första raden i cellområdet (numreringen börjar på 0).
- **`EndColumn (long)`** – sista kolumnen i cellområdet (numreringen börjar på 0).
- **`EndRow (long)`** – sista raden i cellområdet (numreringen börjar på 0).

Den ifyllda `CellRangeAddress`-strukturen används som den första parametern till metoden `insertCells`. Den andra parametern till `insertCells` innehåller ett värde ur uppräkningen `com.sun.star.sheet.CellInsertMode` som definierar hur de värden som finns framför infogningspositionen ska behandlas. Uppräkningen `CellInsertMode` innehåller följande värden:

- **`NONE`** – de aktuella värdena kvarstår i deras nuvarande position.
- **`DOWN`** – cellerna vid och under infogningspositionen flyttas nedåt.
- **`RIGHT`** – cellerna vid och till höger om infogningspositionen flyttas till höger.
- **`ROWS`** – raderna efter infogningspositionen flyttas nedåt.

- **COLUMNS** – kolumnerna efter infogningspositionen flyttas till höger.

Metoden `removeRange` motsvarar metoden `insertCells`. Denna metod tar bort det område som är definierat i strukturen `CellRangeAddress` från tabellen.

```
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)
```

I detta exempel tas cellområdet B2:C3 bort från tabellen, varpå de underliggande cellerna flyttas två rader uppåt. Alternativet för att ta bort rader på detta sätt anges med ett av följande värden från uppräkningen `com.sun.star.sheet.CellDeleteMode`:

- **NONE** – de aktuella värdena kvarstår i deras nuvarande position.
- **UP** – cellerna vid och under infogningspositionen flyttas uppåt.
- **LEFT** – cellerna vid och till höger om infogningspositionen flyttas till vänster.
- **ROWS** – raderna efter infogningspositionen flyttas uppåt.
- **COLUMNS** – kolumnerna efter infogningspositionen flyttas till vänster.

Gränssnittet `XRangeMovement` innehåller två övriga metoder för att flytta (`moveRange`) eller kopiera (`copyRange`) cellområden. I följande exempel flyttas området B2:C3 så att området börjar vid position A6:

```
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)
```

Förutom strukturen `CellRangeAdress` tar metoden `moveRange` en `com.sun.star.table.CellAddress`-struktur som definierar målområdets position. Metoden `CellAddress` ger följande värden:

- **`Sheet (short)`** – tabellens nummer (numreringen börjar på 0).
- **`Column (long)`** – kolumnens nummer (numreringen börjar på 0).
- **`Row (long)`** – radens nummer (numreringen börjar på 0).

Cellinnehållet i målområdet skrivs alltid över av metoden `moveRange`.
Till skillnad från metoden `InsertCells` används ingen parameter för att flytta underliggande celler med metoden `removeRange`.

Metoden `copyRange` fungerar på samma sätt som metoden `moveRange`, med den skillnaden att `copyRange` infogar en kopia av cellområdet i stället för att flytta det.

I funktionalitetstermer är metoderna `insertCell`, `removeRange` och `copyRange` i StarOffice Basic jämförbara med VBA-metoderna `Range.Insert`, `Range.Delete` och `Range.Copy`. I VBA används metoderna med motsvarande `Range`-objekt, och i StarOffice Basic används de med motsvarande `Sheet`-objekt.

Formatering

Ett tabelldokument har egenskaper och metoder för att formatera celler och sidor.

Cellegenskaper

Det finns flera olika alternativ för att formatera celler, till exempel för att ange teckensnitt och textstorlek. Varje cell stöder tjänsterna `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`, vars huvudsakliga egenskaper beskrivs i kapitel 6 (***Textdokument***). Cellspecifik formatering hanteras av tjänsten `com.sun.star.table.CellProperties`. Tjänstens huvudsakliga egenskaper beskrivs i följande avsnitt.

Du kan använda alla namngivna egenskaper i enskilda celler och cellområden.

> Objektet `CellProperties` i StarOffice API är jämförbart med objektet `Interior` i VBA som också används för att definiera cellspecifika egenskaper.

Bakgrundsfärger och skuggor

Tjänsten `com.sun.star.table.CellProperties` innehåller följande egenskaper för bakgrundsfärger och skuggor:

- **`CellBackColor (Long)`** – tabellcellens bakgrundsfärg.
- **`IsCellBackgroundTransparent (Boolean)`** – anger genomskinlig bakgrundsfärg.
- **`ShadowFormat (struktur)`** – anger skuggan för celler (struktur i enlighet med `com.sun.star.table.ShadowFormat`).

Strukturen `com.sun.star.table.ShadowFormat` och den detaljerade specifikationen för cellskuggor har följande struktur:

- **`Location (enum)`** – skuggans placering (värden från strukturen `com.sun.star.table.ShadowLocation`).
- **`ShadowWidth (Short)`** – skuggans bredd i hundradels millimeter.
- **`IsTransparent (Boolean)`** – anger genomskinlig skugga.
- **`Color (Long)`** – skuggans färg.

Följande exempel skriver talet 1 000 till cell B2, ändrar bakgrundsfärgen till rött med egenskapen `CellBackColor` och skapar en ljusgrå skugga för cellen som flyttas 1 mm till vänster och nedåt.

```
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
```

Justering

I StarOffice finns flera olika funktioner som används för att ändra justeringen av texten i en tabellcell.

Följande egenskaper definierar den horisontella och vertikala textjusteringen:

- **`HoriJustify (enum)`** – textens horisontella justering (värden från `com.sun.star.table.CellHoriJustify`)
- **`VertJustify (enum)`** – textens vertikala justering (värden från `com.sun.star.table.CellVertJustify`)
- **`Orientation (enum)`** – textens orientering (värden från `com.sun.star.table.CellOrientation`)
- **`IsTextWrapped (Boolean)`** – anger automatisk radbrytning i cellen
- **`RotateAngle (Long)`** – textens rotationsvinkel i hundradels grader

I följande exempel visas hur du kan "stapla" innehållet i en cell så att enskilda tecken skrivs ut under varandra högst upp till vänster i cellen. Tecknen roteras inte.

```
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
```

Tal-, datum- och textformat

I StarOffice finns flera olika fördefinierade tids- och datumformat. Varje format har ett internt index som används när formatet används i en cell med egenskapen `NumberFormat`. I StarOffice finns metoderna `queryKey` och `addNew` som du kan använda för att utnyttja befintliga talformat och skapa egna format. Metoderna nås via följande funktionsanrop:

```
NumberFormats = Doc.NumberFormats
```

Ett format anges med en formatsträng som är uppbyggd på ett liknande sätt som formateringsfunktionen i StarOffice Basic. Däremot finns det en avgörande skillnad: kommandoformatet använder engelska förkortningar och decimalkomman eller tecken som tusentalsavgränsare, och objektet NumberFormat använder landsspecifika förkortningar.

I följande exempel formateras cell B2 så att tal visas med tre decimaler och kommatecken som tusentalsavgränsare.

```
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
```

Dialogrutan **Formatera celler** i StarOffice Calc innehåller en översikt över de olika formateringsalternativen för celler.

Sidegenskaper

Sidegenskaper är formateringsalternativ som placerar dokumentinnehåll på en sida samt visuella element som upprepas på flera sidor. Dessa är till exempel

- pappersformat
- sidmarginaler
- sidhuvud och sidfot

Proceduren för att definiera sidformat skiljer sig från andra typer av formatering. Cell-, stycke- och teckenelement kan formateras direkt. Sidformat kan också definieras och användas indirekt med sidformatmallar. Till exempel ingår sidhuvud och sidfot i sidformatmallen.

I följande avsnitt beskrivs de huvudsakliga formateringsalternativen för sidor i tabelldokument. Många av formatmallarna som beskrivs är även tillgängliga för textdokument. Sidegenskaperna som gäller båda typerna av dokument är definierade i tjänsten `com.sun.star.style.PageProperties`. Sidegenskaper som endast gäller tabelldokument definieras i tjänsten `com.sun.star.sheet.TablePageStyle`

> Sidegenskaperna (sidmarginaler, kantlinjer och så vidare) för ett Microsoft Office-dokument definieras av ett `PageSetup`-objekt i `Worksheet` – (Excel) eller `Document`-nivån (Word).
> I StarOffice definieras dessa egenskaper av en sidformatmall som i sin tur är länkad till ett dokument.

Sidbakgrund

Tjänsten `com.sun.star.style.PageProperties` definierar följande egenskaper för en sidbakgrund:

- **`BackColor (long)`** – bakgrundsfärgen
- **`BackGraphicURL (String)`** – URL-adress till en bakgrundsbild
- **`BackGraphicFilter (String)`** – namnet på det filter som ska användas för att läsa in bakgrundsbilden
- **`BackGraphicLocation (Enum)`** – placering av bakgrundsbilden (värdet enligt uppräkningen `com.sun.star.style.GraphicLocation`)
- **`BackTransparent (Boolean)`** – gör bakgrunden genomskinlig

Sidformat

Sidformatet definieras med följande egenskaper i tjänsten `com.sun.star.style.PageProperties`:

- **`IsLandscape (Boolean)`** – liggande format
- **`Width (long)`** – sidans bredd i hundradels millimeter.
- **`Height (long)`** – sidans höjd i hundradels millimeter.
- **`PrinterPaperTray (String)`** – namn på det skrivarfack som du vill använda

Följande exempel anger sidstorleken för sidformatmallen "Standard" till liggande DIN A5 (höjd 148 mm, bredd 210 mm):

```
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("Sidformatmallar")
DefPage = PageStyles.getByName("Standard")

DefPage.IsLandscape = True
DefPage.Width = 21000
DefPage.Height = 14800
```

Sidmarginaler, kantlinjer och skuggor

Tjänsten `com.sun.star.style.PageProperties` innehåller följande egenskaper för att justera sidmarginaler, kantlinjer och skuggor:

- **`LeftMargin (long)`** – bredden på vänster sidmarginal i hundradels millimeter.
- **`RightMargin (long)`** – bredden på höger sidmarginal i hundradels millimeter.
- **`TopMargin (long)`** – bredden på övre sidmarginal i hundradels millimeter.
- **`BottomMargin (long)`** – bredden på nedre sidmarginal i hundradels millimeter.
- **`LeftBorder (struktur)`** – anger vänster kantlinje (`com.sun.star.table.BorderLine`)
- **`RightBorder (struktur)`** – anger höger kantlinje (`com.sun.star.table.BorderLine`)
- **`TopBorder (struktur)`** – anger övre kantlinje (`com.sun.star.table.BorderLine`)
- **`BottomBorder (struktur)`** – anger nedre kantlinje (`com.sun.star.table.BorderLine`)
- **`LeftBorderDistance (long)`** – avstånd mellan vänster sidkant och innehållet i hundradels millimeter
- **`RightBorderDistance (long)`** – avstånd mellan höger sidkant och innehållet i hundradels millimeter
- **`TopBorderDistance (long)`** – avstånd mellan övre sidkant och innehållet i hundradels millimeter
- **`BottomBorderDistance (long)`** – avstånd mellan nedre sidkant och innehållet i hundradels millimeter
- **`ShadowFormat (struktur)`** – anger skuggning av innehållsområdet på en sida (strukturen `com.sun.star.table.ShadowFormat`)

I följande exempel anges vänster och höger marginal för sidformatmallen "Standard" till 1 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("Sidformatmallar")
DefPage = PageStyles.getByName("Standard")

DefPage.LeftMargin = 1000
DefPage.RightMargin = 1000
```

Sidhuvud och sidfot

Sidhuvud och sidfot i ett dokument ingår i sidegenskaperna och definieras av tjänsten `com.sun.star.style.PageProperties`. De olika egenskaperna för att formatera sidhuvud är:

- **`HeaderIsOn (Boolean)`** – sidhuvud aktivt
- **`HeaderLeftMargin (long)`** – avstånd mellan sidhuvud och vänster sidmarginal i hundradels millimeter
- **`HeaderRightMargin (long)`** – avstånd mellan sidhuvud och höger sidmarginal i hundradels millimeter.
- **`HeaderBodyDistance (long)`** – avstånd mellan sidhuvud och brödtexten i dokumentet i hundradels millimeter
- **`HeaderHeight (long)`** – sidhuvudets höjd i hundradels millimeter
- **`HeaderIsDynamicHeight (Boolean)`** – sidhuvudets höjd anpassas efter innehållet
- **`HeaderLeftBorder (struktur)`** – detaljer för vänster kantram runt sidhuvud (strukturen `com.sun.star.table.BorderLine`)
- **`HeaderRightBorder (struktur)`** – detaljer för höger kantram runt sidhuvud (strukturen `com.sun.star.table.BorderLine`)
- **`HeaderTopBorder (struktur)`** – detaljer för övre kantram runt sidhuvud (strukturen `com.sun.star.table.BorderLine`)
- **`HeaderBottomBorder (struktur)`** – detaljer för nedre kantram runt sidhuvud (strukturen `com.sun.star.table.BorderLine`)
- **`HeaderLeftBorderDistance (long)`** – avstånd mellan vänster sidkant och sidhuvudets innehåll i hundradels millimeter
- **`HeaderRightBorderDistance (long)`** – avstånd mellan höger sidkant och sidhuvudets innehåll i hundradels millimeter

- **HeaderTopBorderDistance (long)** – avstånd mellan övre sidkant och sidhuvudets innehåll i hundradels millimeter
- **HeaderBottomBorderDistance (long)** – avstånd mellan nedre sidkant och sidhuvudets innehåll i hundradels millimeter
- **HeaderIsShared (Boolean)** – sidhuvuden på jämna och udda sidor har samma innehåll (se `HeaderText`, `HeaderTextLeft` och `HeaderTextRight`)
- **HeaderBackColor (long)** – sidhuvudets bakgrundsfärg
- **HeaderBackGraphicURL (String)** – URL-adress till en bakgrundsbild
- **HeaderBackGraphicFilter (String)** – namnet på det filter som ska användas för att läsa in bakgrundsbilden
- **HeaderBackGraphicLocation (Enum)** – placering av bakgrundsbilden för sidhuvudet (värden enligt uppräkningen `com.sun.star.style.GraphicLocation`)
- **HeaderBackTransparent (Boolean) –** visar en genomskinlig bakgrund för sidhuvudet
- **HeaderShadowFormat (struktur)** – detaljer för sidhuvudets skugga (strukturen `com.sun.star.table.ShadowFormat`)

Egenskaperna för att formatera sidfot är:

- **FooterIsOn (Boolean)** – sidfot är aktiverad
- **FooterLeftMargin (long)** – avstånd mellan sidfot och vänster sidmarginal i hundradels millimeter
- **FooterRightMargin (long)** – avstånd mellan sidfot och höger sidmarginal i hundradels millimeter
- **FooterBodyDistance (long)** – avstånd mellan sidfot och brödtexten i dokumentet i hundradels millimeter
- **FooterHeight (long)** – sidfotens höjd i hundradels millimeter
- **FooterIsDynamicHeight (Boolean)** – sidfotens höjd anpassas efter innehållet
- **FooterLeftBorder (struktur)** – detaljer om vänster kantlinje runt sidfoten (strukturen `com.sun.star.table.BorderLine`)
- **FooterRightBorder (struktur)** – detaljer om höger kantlinje runt sidfoten (strukturen `com.sun.star.table.BorderLine`)
- **FooterTopBorder (struktur)** – detaljer om övre kantlinje runt sidfoten (strukturen `com.sun.star.table.BorderLine`)
- **FooterBottomBorder (struktur)** – detaljer om nedre kantlinje runt sidfoten (strukturen `com.sun.star.table.BorderLine`)
- **FooterLeftBorderDistance (long)** – avstånd mellan vänster sidkant och sidfotens innehåll i hundradels millimeter

- **FooterRightBorderDistance (long)** – avstånd mellan höger sidkant och sidfotens innehåll i hundradels millimeter
- **FooterTopBorderDistance (long)** – avstånd mellan övre sidkant och sidfotens innehåll i hundradels millimeter
- **FooterBottomBorderDistance (long)** – avstånd mellan nedre sidkant och sidfotens innehåll i hundradels millimeter
- **FooterIsShared (Boolean)** – sidfot på jämna och udda sidor har samma innehåll (se `FooterText`, `FooterTextLeft` och `FooterTextRight`).
- **FooterBackColor (long)** – sidfotens bakgrundsfärg
- **FooterBackGraphicURL (String)** – URL-adress till en bakgrundsbild
- **FooterBackGraphicFilter (String)** – namnet på det filter som ska användas för att läsa in bakgrundsbilden
- **FooterBackGraphicLocation (Enum)** – placering av bakgrundsbilden i sidfoten (värden enligt uppräkningen `com.sun.star.style.GraphicLocation`)
- **FooterBackTransparent (Boolean) –** visar en genomskinlig bakgrund för sidfoten
- **FooterShadowFormat (struktur)** – detaljer för sidfotens skugga (strukturen `com.sun.star.table.ShadowFormat`)

Ändra texten i sidhuvuden och sidfötter

Innehållet i sidhuvuden och sidfoten i ett tabelldokument kan nås via följande egenskaper:

- **LeftPageHeaderContent (Object)** – innehållet i sidhuvud på jämna sidor (tjänsten `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageHeaderContent (Object)** – innehållet i sidhuvud på udda sidor (tjänsten `com.sun.star.sheet.HeaderFooterContent`)
- **LeftPageFooterContent (Object)** – innehållet i sidfot på jämna sidor (tjänsten `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageFooterContent (Object)** – innehållet i sidfot på udda sidor (tjänsten `com.sun.star.sheet.HeaderFooterContent`)

Om du inte behöver göra åtskillnad mellan sidhuvud eller sidfot på jämna och udda sidor (egenskapen `FooterIsShared` är `False`), anger du egenskaperna för sidhuvudet eller sidfoten på udda sidor.

Alla namngivna objekt returnerar ett objekt som stöder tjänsten `com.sun.star.sheet.HeaderFooterContent`. Genom att använda (de icke-genuina) egenskaperna `LeftText`, `CenterText` och `RightText`, kan denna tjänst skapa de tre textelementen för ett sidhuvud eller en sidfot i StarOffice Calc.

I följande exempel skriver värdet "Bara ett test" i vänster textfält i sidhuvudet från mallen "Standard".

```
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("Sidformatmallar")
DefPage = PageStyles.getByName("Standard")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Bara ett test."
DefPage.RightPageHeaderContent = HContent
```

Lägg märke till den sista raden i exemplet: när texten har ändrats måste objektet `TextContent` tilldelas till sidhuvudet igen så att ändringarna genomförs.

Ett annat sätt att ändra texten i ett sidhuvud eller en sidfot finns för textdokument (StarOffice Writer) eftersom dessa objekt består av ett enda textblock. Följande egenskaper är definierade i tjänsten `com.sun.star.style.PageProperties`:

- **`HeaderText (Object)`** – textobjekt med sidhuvudets innehåll (`com.sun.star.text.XText`-tjänsten)
- **`HeaderTextLeft (Object)`** – textobjekt med innehåll i sidhuvud på sidor till vänster (`com.sun.star.text.XText`-tjänsten)
- **`HeaderTextRight (Object)`** – textobjekt med innehåll i sidhuvud på sidor till höger (`com.sun.star.text.XText`-tjänsten)
- **`FooterText (Object)`** – textobjekt med sidfotens innehåll (`com.sun.star.text.XText`-tjänsten)
- **`FooterTextLeft (Object)`** – textobjekt med innehåll i sidfoten på sidor till vänster (`com.sun.star.text.XText`-tjänsten)
- **`FooterTextRight (Object)`** – textobjekt med innehåll i sidfoten på sidor till höger (`com.sun.star.text.XText`-tjänsten)

Följande exempel skapar ett sidhuvud med sidformatmallen "Standard" för textdokument och lägger till texten "Bara ett test" i sidhuvudet.

```
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("Sidformatmallar")
DefPage = PageStyles.getByName("Standard")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Bara ett test."
```

I detta fall kan sidhuvudet nås direkt med egenskapen `HeaderText` i sidformatmallen i stället för objektet `HeaderFooterContent`.

Centrering (endast tabeller)

Tjänsten `com.sun.star.sheet.TablePageStyle` används endast i sidformatmallar i StarOffice Calc och används för att centrera cellområden vid utskrift. Tjänsten innehåller följande egenskaper:

- **`CenterHorizontally (Boolean)`** – tabellens innehåll centreras horisontellt
- **`CenterVertically (Boolean)`** – tabellens innehåll centreras vertikalt

Definition av element för utskrift (endast tabeller)

När du formaterar tabeller kan du ange om element på sidan ska synas eller inte. För detta ändamål innehåller tjänsten `com.sun.star.sheet.TablePageStyle` följande egenskaper:

- **`PrintAnnotations (Boolean)`** – skriver ut cellkommentarer
- **`PrintGrid (Boolean)`** – skriver ut cellrutnätet
- **`PrintHeaders (Boolean)`** – skriver ut rad- och kolumnrubriker
- **`PrintCharts (Boolean)`** – skriver ut diagram i tabellen
- **`PrintObjects (Boolean)`** – skriver ut inbäddade objekt
- **`PrintDrawing (Boolean)`** – skriver ut ritobjekt
- **`PrintDownFirst (Boolean)`** – om innehållet i en tabell sträcker sig över flera sidor skrivs de först ut i lodrät ordning, och sedan i vågrät ordning från höger
- **`PrintFormulas (Boolean)`** – skriver ut formler i stället för beräknade värden
- **`PrintZeroValues (Boolean)`** – skriver ut nollvärden

Redigera tabelldokument effektivt

De tidigare avsnitten behandlade den grundläggande uppbyggnaden av tabelldokument. Detta avsnitt beskriver olika sätt att komma åt enskilda celler eller cellområden.

Cellområden

Förutom ett objekt för enskilda celler (tjänsten `com.sun.star.table.Cell`) innehåller StarOffice även objekt som representerar cellområden. Sådana `CellRange`-objekt skapas med anropet `getCellRangeByName` i tabelldokumentobjektet:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Tabell 1")
CellRange = Sheet.getCellRangeByName("A1:C15")
```

Ett kolon (:) används för att ange ett cellområde i tabelldokumentet. Till exempel representerar A1:C15 alla celler i raderna 1 till 15 i kolumnerna A, B och C.

Platsen för individuella celler i ett cellområde kan avgöras med metoden `getCellByPosition` där koordinaterna för cellen överst till vänster i cellområdet är (0,0). I följande exempel används denna metod för att skapa ett objekt av cell C3.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Tabell 1")
CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

Formatera cellområden

Precis som individuella celler kan du använda formatering i cellområden med tjänsten `com.sun.star.table.CellProperties`. Mer information och exempel på denna tjänst finns i avsnittet ***Formatering***.

Beräkningar med cellområden

Du kan använda metoden `computeFunction` för att utföra matematiska operationer på cellområden. Metoden `computeFunction` tar en konstant som parameter som beskriver den matematiska funktion du vill använda. Konstanterna är definierade i uppräkningen `com.sun.star.sheet.GeneralFunction`. Följande värden är tillgängliga:

- **`SUM`** – summan av alla numeriska värden
- **`COUNT`** – antal värden (även icke-numeriska värden)

- **COUNTNUMS** – antal numeriska värden
- **AVERAGE** – medelvärdet av alla numeriska värden
- **MAX** – största numeriska värde
- **MIN** – minsta numeriska värde
- **PRODUCT** – produkten av alla numeriska värden
- **STDEV** – standardavvikelsen
- **VAR** – variance
- **STDEVP** – standardavvikelse baserat på en total population
- **VARP** – varians baserat på en total population

I följande exempel beräknas medelvärdet av cellområdet A1:C3, varpå resultatet visas i en meddelanderuta:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Tabell 1")
CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Ta bort cellinnehåll

Metoden `clearContents` förenklar processen för att ta bort cellinnehåll och cellområden genom att den tar bort en viss typ av innehåll från ett cellområde.

I följande exempel tas alla strängar och direkt formateringsinformation bort från cellområdet 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)
```

Flaggorna som anges i `clearContents` finns i konstantlistan `com.sun.star.sheet.CellFlags`. Listan innehåller följande konstanter:

- **VALUE** – numeriska värden som inte är formaterade som tids- eller datumvärden
- **DATETIME** – numeriska värden som är formaterade som tids- eller datumvärden
- **STRING** – strängar
- **ANNOTATION** – kommentarer som är länkade till celler
- **FORMULA** – formler
- **HARDATTR** – direkt cellformatering
- **STYLES** – indirekt formatering
- **OBJECTS** – ritobjekt som är kopplade till celler
- **EDITATTR** – teckenformatering som endast gäller för delar av celler

Du kan också summera konstanter för att ta bort flera olika typer av information med `clearContents`.

Söka och ersätta cellinnehåll

Tabelldokument innehåller precis som textdokument en funktion för att söka och ersätta.

Beskrivningsobjektet för att söka och ersätta i tabelldokument skapas inte direkt från dokumentobjektet, utan genom listan `Sheets`. I följande exempel visas hur du kan söka och ersätta i tabelldokument:

```
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 = "är"
ReplaceDescriptor.ReplaceString = "var"

For I = 0 to Doc.Sheets.Count - 1
      Sheet = Doc.Sheets(I)
      Sheet.ReplaceAll(ReplaceDescriptor)
Next I
```

I exemplet används den första sidan i dokumentet för att skapa en `ReplaceDescriptor` som sedan används i alla sidor i slingan.

KAPITEL 8

Teckningar och presentationer

Detta kapitel innehåller en inledning till makrostyrd redigering av teckningar. I det första avsnittet beskrivs uppbyggnaden av teckningar samt grundläggande element som ingår i teckningar. Det andra avsnittet behandlar mer komplicerade redigeringsfunktioner, som grupperingar, rotation och skalning av objekt.

Mer information om att skapa, öppna och spara teckningar finns i kapitel 5, ***Arbeta med StarOffice-dokument***.

Uppbyggnaden av teckningar

StarOffice har ingen begränsning för antalet sidor i ett teckningsdokument. Du kan utforma varje sida separat. Det finns inte heller någon begränsning för hur många teckningselement som kan finnas på en sida.

Situationen kompliceras i viss mån av ***nivåer***. Varje teckningsdokument innehåller som standard nivåerna ***Layout***, ***Controls*** och ***Måttlinjer***. Alla teckningselement läggs till i nivån ***Layout***. Du kan skapa fler nivåer om det behövs. Mer information om nivåer finns i StarOffice Developer's Guide.

Sidor

Sidorna i ett teckningsdokument kan nås via listan `DrawPages`. Du kan nå enskilda sidor antingen via sidans namn eller index. Om ett dokument innehåller en sida med namnet ***Sida 1***, har följande exempel samma funktion.

Exempel 1:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

Exempel 2:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Sida 1")
```

I exempel 1 adresseras sidan med sitt index (indexering börjar på 0). I det andra exemplet används sidans namn och metoden `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
```

I anropet ovan returneras ett sidobjekt som stöder tjänsten `com.sun.star.drawing.DrawPage`. Tjänsten innehåller följande egenskaper:

- **`BorderLeft (Long)`** – vänster sidmarginal i hundradels millimeter
- **`BorderRight (Long)`** – höger sidmarginal i hundradels millimeter
- **`BorderTop (Long)`** – övre marginal i hundradels millimeter
- **`BorderBottom (Long)`** – nedre marginal i hundradels millimeter
- **`Width (Long)`** – sidans bredd i hundradels millimeter
- **`Height (Long)`** – sidans höjd i hundradels millimeter
- **`Number (Short)`** – sidantal (numreringen börjar på 1), skrivskyddat
- **`Orientation (Enum)`** – sidorientering (i enlighet med uppräkningen `com.sun.star.view.PaperOrientation`)

Om dessa inställningar ändras kommer ***samtliga*** sidor i dokumentet att påverkas.

I följande exempel anges storleken för ett nyöppnat teckningsdokument till 20 × 20 centimeter med en sidmarginal på 0,5 centimeter.

```
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
```

Grundläggande egenskaper för teckningsobjekt

Teckningsobjekt är former (rektanglar, cirklar och så vidare), linjer och textobjekt. Alla dessa har gemensamma funktioner och stöder tjänsten `com.sun.star.drawing.Shape`. Tjänsten definierar egenskaperna `Size` och `Position` i ett ritobjekt.

I StarOffice Basic finns också flera andra sätt att ändra egenskaper som formatering och fyllning. Vilka formateringsalternativ som är tillgängliga beror på den aktuella typen av ritobjekt.

I följande exempel skapas och infogas en rektangel i ett teckningsdokument:

```
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)
```

I detta exempel används anropet `StarDesktop.CurrentComponent` för att avgöra vilken typ av dokument som är öppet. Dokumentobjektet returnerar den första sidan i teckningen med anropet `drawPages(0)`.

Strukturerna `Point` och `Size` med samma origo (vänster överkant) och storlek som ritobjektet initieras sedan. Längderna anges i hundradels millimeter.

Programkoden använder sedan anropet `Doc.createInstance` för att skapa en rektangel som anges av tjänsten `com.sun.star.drawing.RectangleShape`. Slutligen infogas ritobjekt i en sida med anropet `Page.add`.

Fyllningsegenskaper

Detta avsnitt innehåller fyra beskrivningar av en tjänst och i varje exempelkod används ett rektangelelement som kombinerar olika typer av formatering. Fyllningsegenskaper kombineras med tjänsten `com.sun.star.drawing.FillProperties`.

StarOffice kan hanterar alla de fyra vanligaste typerna av formatering för en fyllning. Den enklaste varianten är en solid fyllning med en färg. Du kan också använda olika alternativ för att definiera gradienter och skrafferingar. Den fjärde varianter innebär att en bild projiceras i fyllningsområdet.

Fyllningsläget för ett ritobjekt definieras med egenskapen `FillStyle`. De olika värdena är definierade i `com.sun.star.drawing.FillStyle`.

Solida fyllningar

Den huvudsakliga egenskapen för en solid fyllning är

- **`FillColor (Long)`** – fyllningens färg.

För att använda detta läge anger du egenskapen `FillStyle` till `SOLID`.

I följande exempel skapas en rektangel som fylls med färgen röd (RGB-värde 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)
```

Färggradienter

Om du anger egenskapen `FillStyle` till `GRADIENT` kan du använda en färggradient som fyllning.

Om du vill använda en fördefinierad gradient anger du motsvarande namn i egenskapen `FillTransparenceGradientName`. Om du vill definiera en egen färggradient måste du fylla i en `com.sun.star.awt.Gradient`-struktur och ange den som egenskapen `FillGradient`. Egenskapen hanterar följande alternativ:

- **`Style (Enum)`** – typ av gradient, till exempel linjär eller radiell (standardvärden enligt `com.sun.star.awt.GradientStyle`)
- **StartColor (Long)** – gradientens startfärg
- **EndColor (Long)** – gradientens slutfärg
- **Angle (Short)** – gradientens vinkel i tiondels grader
- **XOffset (Short)** – X-koordinat där gradienten börjar, i hundradels millimeter
- **YOffset (Short)** – Y-koordinat där gradienten börjar, i hundradels millimeter
- **StartIntensity (Short)** – intensitet för `StartColor` som en procentandel (i StarOffice Basic kan du även ange värden över 100 procent)
- **EndIntensity (Short)** – intensitet för `EndColor` som en procentandel (i StarOffice Basic kan du även ange värden över 100 procent)
- **StepCount (Short)** – antal färgsteg som ska användas för gradienten

I följande exempel visas hur du kan använda färggradienter med strukturen `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)
```

I detta exempel skapas en linjär gradient (`Style` = `LINEAR`). Gradienten startar med röttStartColor) högst uppe till vänster, och sträcker sig i 45 graders vinkel (`Angle`) till grönt (`EndColor`) längst ned till höger. Färgintensiteten för start- och slutfärger är 150 procent. (`StartIntensity` och `EndIntensity`) vilket innebär att färgerna är mer intensiva än de värden som anges med egenskaperna `StartColor` och `EndColor`. Färggradienten visas med ett hundra enskilda färger (`StepCount`).

Skrafferingar

Om du vill skapa en skraffering anger du egenskapen `FillStyle` till `HATCH`. Programkoden för att definiera en skraffering liknar den för gradienter. I detta fall används strukturen `com.sun.star.drawing.Hatch` för att definiera skrafferingens utseende. Strukturen innehåller följande egenskaper:

- **`Style (Enum)`** – typ av skraffering: enkel, rutad eller rutad med diagonaler (värden enligt `com.sun.star.awt.HatchStyle`)

- **`Color (Long)`** – linjernas färg
- **`Distance (Long)`** – avståndet mellan linjer i hundradels millimeter
- **`Angle (Short)`** – vinkeln för skrafferingen i tiondels grader

I följande exempel visas hur du kan använda en skrafferingsstruktur:

```
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)
```

I exemplet skapas en enkel skraffering (`HatchStyle` = `SINGLE`) vars linjer är roterade 45 grader (`Angle`). Linjerna är mörkgrå (`Color`) och har 0,2 millimeters avstånd (`Distance`).

Bitmappar

Om du vill använda en bitmapp som en fyllning anger du egenskapen `FillStyle` till `BITMAP`. Om bitmappen redan är tillgänglig i StarOffice anger du bitmappens namn i egenskapen `FillBitMapName` och visningsstilen (enkel, sida vid sida eller utsträckt) i egenskapen `FillBitmapMode` (standardvärden enligt `com.sun.star.drawing.BitmapMode`).

Om du vill använda en extern bitmappsfil anger du filens URL i egenskapen `FillBitmapURL`.

I följande exempel skapas en rektangel som är fylld med bitmappen Himmel sida vid sida.

```
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 = "Himmel"
RectangleShape.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT


Page.add(RectangleShape)
```

Genomskinlighet

Du kan justera hur genomskinlig en fyllning ska vara. Det enklaste sättet att ändra ett ritelements genomskinlighet är att använda egenskapen `FillTransparence`.

I följande exempel skapas en röd rektangel med 50 procents genomskinlighet.

```
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)
```

Om du vill göra fyllningen genomskinlig anger du egenskapen `FillTransparence` till 100.

Förutom att använda egenskapen `FillTransparence` kan du även använda tjänsten `com.sun.star.drawing.FillProperties` och den motsvarande egenskapen `FillTransparenceGradient`. Denna metod används för att ange en gradient som definierar genomskinligheten i ett område.

Linjeegenskaper

Alla ritobjekt som kan ha en kantlinje stöder tjänsten `com.sun.star.drawing.LineStyle`. Denna tjänst stöder bland annat följande egenskaper:

- **`LineStyle (Enum)`** – typ av linje (standardvärden enligt `com.sun.star.drawing.LineStyle`)
- **`LineColor (Long)`** – linjefärg
- **`LineTransparence (Short)`** – genomskinlighet
- **`LineWidth (Long)`** – tjocklek i hundradels millimeter
- **`LineJoint (Enum)`** – övergångar till förbindelsepunkter (standardvärden enligt `com.sun.star.drawing.LineJoint`)

I följande exempel skapas en rektangel med solid kant (`LineStyle` = `SOLID`) som är 5 millimeter tjock (`LineWidth`) och är till hälften genomskinlig. Linjens start- och slutpunkt sträcks ut till en gemensam skärningspunkt (`LineJoint` = `MITER`) så att en rätvinklig triangel bildas.

```
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)
```

Förutom ovanstående egenskaper innehåller tjänsten `com.sun.star.drawing.LineStyle` alternativ för att rita streckade och prickade linjer. Mer information finns i referensmaterialet till StarOffice API.

Textegenskaper (ritobjekt)

Tjänsten `com.sun.star.style.CharacterProperties` eller `com.sun.star.style.ParagraphProperties` kan användas för att formatera text i ritobjekt. Formateringen gäller individuella tecken och stycken och beskrivs närmare i kapitel 6 (***Textdokument***).

I följande exempel infogas text i en rektangel och formateras med tjänsten `com.sun.star.style.CharacterProperties`.

```
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 = "Detta är ett test"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

Exemplet använder egenskapen `String` i rektangeln för att infoga texten och egenskaperna `CharWeight` och `CharFontName` från tjänsten `com.sun.star.style.CharacterProperties` för att formatera teckensnittet.

Texten kan endast infogas efter att ritobjektet lagts till på sidan. Du kan också använda tjänsten `com.sun.star.drawing.Text` för att placera och formatera text i ritobjekt. Några av de vanligaste egenskapern är:

- **`TextAutoGrowHeight (Boolean)`** – anpassar ritelementets höjd efter den text det innehåller
- **`TextAutoGrowWidth (Boolean)`** – anpassar ritelementets bredd till den text det innehåller
- **`TextHorizontalAdjust (Enum)`** – horisontell placering av texten i ritelementet (standardvärden enligt `com.sun.star.drawing.TextHorizontalAdjust`)
- **`TextVerticalAdjust (Enum)`** – vertikal placering av texten i ritelementet (standardvärden enligt `com.sun.star.drawing.TextVerticalAdjust`)
- **`TextLeftDistance (Long)`** – avstånd till vänster mellan ritelementet och texten i hundradels millimeter
- **`TextRightDistance (Long)`** – avstånd till höger mellan ritelementet och texten i hundradels millimeter
- **`TextUpperDistance (Long)`** – övre avstånd mellan ritelementet och texten i hundradels millimeter
- **`TextLowerDistance (Long)`** – nedre avstånd mellan ritelementet och texten i hundradels millimeter

I följande exempel visas hur namngivna egenskaper används.

```
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 = "Detta är ett test"     ' Kan endast ske efter Page.add!

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
```

Det här kodexemplet infogar ett ritelement på sidan och lägger till text högst upp till vänster i ritobjektet med egenskaperna `TextVerticalAdjust` och `TextHorizontalAdjust`. Det minsta avståndet mellan textens kant och ritobjektet anges till tre millimeter.

Skuggningsegenskaper

Du kan lägga till en skuggning i de flesta typer av ritobjekt med tjänsten `com.sun.star.drawing.ShadowProperties`. Tjänstens egenskaper är:

- **`Shadow (Boolean)`** – aktiverar skuggan
- **`ShadowColor (Long)`** – skuggningsfärg
- **`ShadowTransparence (Short)`** – skuggans genomskinlighet
- **`ShadowXDistance (Long)`** – vertikalt avstånd mellan skuggan och ritobjektet i hundradels millimeter
- **`ShadowYDistance (Long)`** – horisontellt avstånd mellan skuggan och ritobjektet i hundradels millimeter

I följande exempel skapas en rektangel med en skugga som är vertikalt och horisontellt justerad med 2 millimeter från rektangeln. Skuggan visas med mörkgrå färg och 50 procents genomskinlighet.

```
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)
```

En översikt över olika ritobjekt

Rektangelformer

Rektanglar (`com.sun.star.drawing.RectangleShape`) stöder följande tjänster för att formatera objekt:

- **Fyllningsegenskaper** – `com.sun.star.drawing.FillProperties`
- **Linjeegenskaper** – `com.sun.star.drawing.LineProperties`
- **Textegenskaper** – `com.sun.star.drawing.Text` (med `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`)
- **Skuggningsegenskaper** – `com.sun.star.drawing.ShadowProperties`
- **CornerRadius (Long)** – radie för avrundning av hörn i hundradels millimeter

Cirklar och ellipser

Tjänsten `com.sun.star.drawing.Tjänsten EllipseShape` har hand om cirklar och ellipser och stöder följande tjänster:

- **Fyllningsegenskaper** – `com.sun.star.drawing.FillProperties`
- **Linjeegenskaper** – `com.sun.star.drawing.LineProperties`
- **Textegenskaper** – `com.sun.star.drawing.Text` (med `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`)
- **Skuggegenskaper** – `com.sun.star.drawing.ShadowProperties`

Med cirklar och ellipser får du utöver de här tjänsterna även följande egenskaper:

- **`CircleKind (Enum)`** - cirkel- eller ellipstyp (standardvärden i överensstämmelse med `com.sun.star.drawing.CircleKind`)
- **`CircleStartAngle (Long)`** - startvinkel i tiondelar av en grad (endast för cirkel- eller ellipssegment)
- **`CircleEndAngle (Long)`** – slutvinkel i tiondelar av en grad (endast för cirkel- eller ellipssegment)

Egenskapen `CircleKind` bestämmer om ett objekt är en hel cirkel, en cirkelbit eller en cirkelsektion. Följande värden är tillgängliga:

- **`com.sun.star.drawing.CircleKind.FULL`** – hel cirkel eller hel ellips
- **`com.sun.star.drawing.CircleKind.CUT`** – cirkelsektion (cirkeldel vars gränsytor är direkt länkade till varandra)
- **`com.sun.star.drawing.CircleKind.SECTION`** – cirkelbit
- **`com.sun.star.drawing.CircleKind.ARC`** – vinkel (cirkellinjen ej inkluderad)

I följande exempel skapas en cirkelbit med en vinkel på 70 grader (beräknad från skillnaden mellan startvinkeln på 20 grader och slutvinkeln på 90 grader).

```
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)
```

Linjer

I StarOffice finns tjänsten `com.sun.star.drawing.LineShape` för linjeobjekt. Linjeobjekt stöder alla allmänna formateringstjänster med undantag för områden. Nedan följer alla egenskaper som associeras med tjänsten `LineShape`:

- **Linjeegenskaper** – `com.sun.star.drawing.LineProperties`
- **Textegenskaper** – `com.sun.star.drawing.Text` (med `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`)
- **Skuggegenskaper** – `com.sun.star.drawing.ShadowProperties`

I följande exempel skapas och formateras en linje med hjälp av de namngivna egenskaperna. Linjens originalvärde anges i egenskapen `Placering`, medan koordinaterna som listas i egenskapen `Storlek` anger linjens slutpunkt.

```
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)
```

Polypolygonformer

StarOffice stöder även komplexa polygonformer via tjänsten `com.sun.star.drawing.PolyPolygonShape`. En ***PolyPolygon*** är inte en enkel polygon utan en multipel polygon. Flera oberoende listor som innehåller hörnpunkter kan därför anges och kombineras för att skapa ett komplett objekt.

Alla formateringsegenskaper för ritobjekt finns för rektangulära former liksom för polypolygoner.

- **Fyllningsegenskaper** – `com.sun.star.drawing.FillProperties`
- **Linjeegenskaper** – `com.sun.star.drawing.LineProperties`
- **Textegenskaper** – `com.sun.star.drawing.Text` (med `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`)
- **Skuggegenskaper** – `com.sun.star.drawing.ShadowProperties`

Tjänsten `PolyPolygonShape` har dessutom en egenskap som du kan bestämma koordinaterna för en polygon med:

- `PolyPolygon (Fält)` – innehåller koordinaterna för polygonen (dubbelmatris med punkterna för typen `com.sun.star.awt.Point`)

I följande exempel visas hur du kan definiera en triangel med tjänsten `PolyPolygonShape` .

```
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 måste inträffa innan koordinaterna har angetts

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())
```

Eftersom polygonens punkter definieras som absoluta värden behöver du inte ange storleken för polygonens startposition. I stället måste du skapa en matris av punkterna och paketera denna matris i en andra matris (använd anropet `Array(Coordinates())`och tilldela sedan polygonen matrisen. Innan motsvarande anrop kan göras måste polygonen infogas i dokumentet.

Tack vare dubbelmatrisen i definitionen kan du skapa komplexa former genom att sammanfoga flera polygoner. Du kan t.ex. skapa en rektangel och sedan infoga en annan rektangel inuti den för att skapa ett hål i den ursprungliga rektangeln.

```
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 måste inträffa innan koordinaterna har angetts

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())
```

I StarOffice används en enkel regel som tar hänsyn till vilka områden som är fyllda och vilka områden som är hål: den yttre formens kant är alltid polypolygonens yttre kant. Nästa linje inåt är formens inre kant och markerar övergången till det första hålet. Om det finns ytterligare en linje inåt markerar den övergången till ett fyllt område.

Grafik

De sista teckningselement som presenteras här är grafiska objekt som baseras på tjänsten `com.sun.star.drawing.GraphicObjectShape`. Dessa kan användas på vilket grafikobjekt som helst i StarOffice, förutsatt att sättet på hur det visas kan anpassas med med en serie egenskaper.

Grafiska objekt stöder två av de allmänna formateringsegenskaperna:

- **Textegenskaper** – `com.sun.star.drawing.Text` (med `com.sun.star.style.CharacterProperties` och `com.sun.star.style.ParagraphProperties`)
- **Skuggegenskaper** – `com.sun.star.drawing.ShadowProperties`

Ytterligare egenskaper som stöds av grafiska objekt:

- **`GraphicURL (String)`** – grafikens URL
- **`AdjustLuminance (Short)`** – färgernas ljusstyrka, uttryckt i procent (negativa värden är tillåtna)
- **`AdjustContrast (Short)`** – kontrast, uttryckt i procent (negativa värden är tillåtna)
- **`AdjustRed (Short)`** – rödvärde, uttryckt i procent (negativa värden är tillåtna)
- **`AdjustGreen (Short)`** – grönvärde, uttryckt i procent (negativa värden är tillåtna)
- **`AdjustBlue (Short)`** – blåvärde, uttryckt i procent (negativa värden är tillåtna)
- **`Gamma (Short)`** – grafikens gammavärde
- **`Transparens (Short)`** – grafikens transparens uttryckt i procent
- **`GraphicColorMode (enum)`** – färgläge, t.ex. standard, gråskalor, svartvitt (standardvärde enligt `com.sun.star.drawing.ColorMode`)

I följande exempel visas hur du infogar en sida i det grafiska objektets object.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                      ' specifikationer som är betydelselösa eftersom
                                    koordinaterna är bindande
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)
```

Med den här koden infogas det grafiska objektet `test.jpg` och dess utseende anpassas med egenskaperna `Justera`. I det här exemplet avbildas grafiken som 40 procent transparent och inga andra färgkonversioner är aktuella (`GraphicColorMode` = `STANDARD`).

Redigera ritobjekt

Gruppera objekt

I många fall är det användbart att gruppera flera enskilda ritobjekt så att de fungerar som ett enda stort objekt.

I följande exempel kombineras två ritobjekt:

```
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
' skapa kvadratiska teckningselement
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' skapa teckningselement som är cirklar
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)
' kombinera teckningselement av typerna kvadrat och cirkel
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' centrera kombinerade teckningselement
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
```

Med den här koden skapas en rektangel och en cirkel som infogas på sidan. Sedan skapas ett objekt som stöder tjänsten `com.sun.star.drawing.ShapeCollection` och använder metoden `Lägg till` för att lägga till rektangeln och cirkeln i objektet. The `ShapeCollection` läggs till på sidan med hjälp av `Grupp-metoden` och returnerar det faktiska `Grupp-objekt` som kan redigeras som en enskild `form`.

Om du vill formatera enskilda objekt i en grupp använder du formateringen innan du lägger till dem i gruppen. Du kan inte ändra objekt när de har blivit en del av gruppen.

Rotera och skära ritobjekt

Alla ritobjekt som beskrivits i tidigare avsnitt kan roteras och skäras med tjänsten `com.sun.star.drawing.RotationDescriptor`.

Med tjänsten följer de här egenskaperna:

- **`RotateAngle (Long)`** – roteringsvinkel i hundradelar av en grad
- **`ShearAngle (Long)`** – skärningsvinkel i hundradelar av en grad

I följande exempel skapas en rektangel som roteras 30 grader med egenskapen `RotateAngle`:

```
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)
```

I nästa exempel skapas en likadan rektangel som i det tidigare exemplet men den skärs i stället genom 30 grader med egenskapen `ShearAngle`.

```
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)
```

Söka och ersätta

Det går att söka och ersätta i teckningsdokument precis som i textdokument. Funktionen liknar den som används i textdokument och som beskrivs i kapitel 6, ***Textdokument***. I teckningsdokument skapas emellertid inte beskrivningsobjekten för sök- och ersätt direkt via dokumentobjektet utan via den associerade teckennivån. I följande exempel beskrivs ersättningsprocessen i en teckning.

```
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 = "är"
ReplaceDescriptor.ReplaceString = "var"

For I = 0 to Doc.drawPages.Count - 1
      Page = Doc.drawPages(I)
      Page.ReplaceAll(ReplaceDescriptor)
Next I
```

I den här koden används dokumentets första `DrawPage` för att skapa en `ReplaceDescriptor` och sedan används denna beskrivare i en loop för alla sidor i teckningsdokumentet.

Presentationer

StarOffice-presentationer baseras på teckningsdokument. Varje sida i en presentation är en diabild. Du får tillgång till diabilderna på samma sätt som du kommer åt en standardteckning via listan `DrawPages` för dokumentobjektet. `Tjänsten com.sun.star.presentation.PresentationDocument`, har hand om presentationsdokumenten och tillhandahåller även hela tjänsten `com.sun.star.drawing.DrawingDocument`.

Arbeta med presentationer

Utöver teckningsfunktionerna som tillhandahålls av egenskapen `Presentation` innehåller presentationsdokumentet ett presentationsobjekt som ger tillgång till huvudegenskaper och kontrollmekanismer för presentationer. Det här objektet ger t.ex. en `startmetod` som alla presentationer kan startas med.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

Med koden som används i exemplet skapas ett `Doc-objekt` som refererar till det aktuella presentationsdokumentet och som etablerar det associerade presentationsobjektet. start()-metodens objekt används för att starta exemplet och för att köra skärmpresentationen.

Följande metoder finns som presentationsobjekt:

- **`start`** – startar presentationen
- **`end`** – avslutar presentationen
- **`rehearseTimings`** – startar presentationen från början och upprättar sin runtime

Följande egenskaper är också tillgängliga:

- **`AllowAnimations (Boolean)`** – kör animationer i presentationen
- **`CustomShow (String)`** – gör att du kan ge presentationen ett namn som du kan använda som referens i presentationen.
- **`FirstPage (String)`** – namn på den diabild som du vill starta presentationen med
- **`IsAlwaysOnTop (Boolean)`** – visar alltid presentationsfönstret som det första fönstret på skärmen
- **`IsAutomatic (Boolean)`** – kör automatiskt igenom presentationen
- **`IsEndless (Boolean)`** – startar om presentationen från början när den har nått slutet
- **`IsFullScreen (Boolean)`** – startar automatiskt presentationen i helskärmsläge
- **`IsMouseVisible (Boolean)`** – visar musen under presentationen
- **`Pause (long)`** – den tid som en tom skärmbild visas i slutet av presentationen
- **`StartWithNavigator (Boolean)`** – visar Navigator-fönstet när presentationen startar
- **`UsePn (Boolean)`** – visar pekaren under presentationen

KAPITEL 9

Diagram

I StarOffice kan data visas i diagram, vilket skapar grafiska länkar mellan data i form av staplar, cirkeldiagram, linjer eller andra element. Data kan visas som antingen 2D- eller 3D-grafik och de enskilda diagramelementens utseende kan anpassas på sätt som liknar de som används för teckningselement.

Om data finns tillgängliga i tabellform kan de länkas dynamiskt till diagrammet. Alla ändringar som görs i grundläggande data ger omedelbar effekt i det tilldelade diagrammet. I det här kapitlet görs en översikt över programmeringsgränssnittet för diagrammoduler i StarOffice och fokus läggs på användningen av diagram i tabelldokument.

Använda diagram i tabelldokument

Diagram fungerar inte som oberoende dokument i StarOffice utan som objekt som är inbäddade i befintliga dokument.

Diagram som används i text- och teckningsdokument är isolerade från dokumentinnehållet. När de används i tabelldokument däremot finns det en mekanism som gör att det går att länka mellan data i dokumentet och inbäddade diagram. I följande exempel förklaras interaktionen mellan tabelldokument och diagram.

```
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("MittDiagram", Rect, RangeAddress(), True, True)
```

Även om koden i exemplet kan tyckas komplicerad består de centrala processerna bara av tre rader: den första raden skapar dokumentvariabeln `Doc` som ger en referens till det aktuella tabelldokumentet (`Doc` line `= StarDesktop.CurrentComponent`). Sedan skapas med hjälp av koden i exemplet en lista som innehåller alla diagram i det första tabelldokumentet (`Charts` line `= Doc.Sheets(0).Charts`). Slutligen läggs ett nytt diagram till sist i listan med metoden `addNewByName`. Det nya diagrammet blir sedan synligt för användaren.

Den sista raden initierar tilläggsstrukturerna `Rect` och `RangeAddress` som även förses av metoden `addNewByName` som en parameter. `Rect` bestämmer diagrammets position i tabelldokumentet. `RangeAddress` bestämmer området som innehåller de data som ska länkas till diagrammet.

I det föregående exemplet skapades ett stapeldiagram. Om du vill ha en annan typ av grafik måste stapeldiagrammet ersättas explicit:

```
Chart = Charts.getByName("MittDiagram").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

Den första raden definierar det motsvarande diagramobjektet. Den andra raden ersätter det aktuella diagrammet med ett nytt, vilket i exemplet är ett linjediagram.

I Excel görs en distinktion mellan diagram som har infogats som en separat sida i ett Excel-dokument och diagram som är inbäddade på en tabellsida. Därför definieras också två olika åtkomstmetoder för diagrammen. I StarOffice Basic görs inte den här distinktionen eftersom diagram i StarOffice Calc alltid skapas som inbäddade objekt på en tabellsida. Åtkomst till diagrammen sker allltid via listan `Diagram` för det associerade `Tabell`-objektet.

Diagrammens struktur

Diagrammets struktur och listan över de tjänster och gränssnitt som stöds beror på vilken typ av diagram det är. Metoder och egenskaper för Z-axeln är t.ex. bara tillgängliga för 3D-diagram och inte för 2D-diagram. När det gäller cirkeldiagram saknas gränssnitt för att arbeta med axlar.

Enskilda diagramelement

Rubrik, underrubrik och nyckel

Grundläggande element i alla diagram är en rubrik, en underrubrik och en nyckel. Alla diagram har egna objekt för de här elementen. Objektet `Diagram` tillhandahåller följande egenskaper som används för att administrera de här elementen:

- **`HasMainTitle (Boolean)`** – aktiverar rubriken.
- **`Rubrik (Object)`** – objekt meddetaljerad information om diagramrubriken (stöder tjänsten `com.sun.star.chart.ChartTitle`).
- **`HasSubTitle(Boolean)`** – aktiverar underrubriken.
- **`Underrubrik (Object)`** – objekt med detaljerad information om diagrammets underrubrik (stöder tjänsten `com.sun.star.chart.ChartTitle`).

- **`HasLegend (Boolean)`** – aktiverar nyckeln.
- **Förklaring (Object)** – objekt med detaljerad information om nyckeln till diagrammet (stöder tjänsten `com.sun.star.chart.ChartLegendPosition`).

I flera avseenden motsvarar de element som angetts ett teckningselement. Detta beror på att både tjänsten `com.sun.star.chart.ChartTitle` och metoden `com.sun.star.chart.ChartLegendPosition` stöder tjänsten `com.sun.star.drawing.Shape` som utgör den tekniska programbasen för teckningselementen.

Därför kan användare bestämma position och storlek för elementen genom att använda egenskaperna `Storlek` och `Position`.

De andra fyllnings- och linjeegenskaperna (tjänsterna `com.sun.star.drawing.FillProperties` och `com.sun.star.drawing.LineStyle`) liksom teckenegenskaperna (tjänsten `com.sun.star.style.CharacterProperties` tillhandahålls för du ska kunna formatera elementen.

`com.sun.star.chart.ChartTitle` innehåller inte bara de namngivna formategenskaperna utan också två andra egenskaper:

- **`TextRotation (Long)`** – textrotationsvinkel i hundradelar av en grad.
- **`String (String)`** – text som ska visas som rubrik eller underrubrik

Diagramnyckeln (`tjänsten com.sun.star.chart.ChartLegend`) innehåller dessutom följande egenskap:

- **`Justering (Enum)`** – position där nyckeln visas (standardvärde i överensstämmelse med `com.sun.star.chart.ChartLegendPosition`).

I följande exempel skapas ett diagram som får rubriken "Test", underrubriken "Test 2" och en nyckel. Nyckeln har en grå bakgrundsfärg, en teckenstorlek på 7 punkter och är placerad längst ned i diagrammet.

```
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("MittDiagram", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MittDiagram").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
```

Bakgrund

Alla diagram har ett bakgrundsområde och alla områden innehåller ett objekt som det går att komma åt via följande egenskaper för diagramobjektet:

- **`Område (Object)`** – diagrammets bakgrundsområde (stöder tjänsten `com.sun.star.chart.ChartArea`).

Diagrammets bakgrund täcker hela bakgrundsområdet inklusive området under rubriken, underrubriken och diagramnyckeln. Den associerade tjänsten `com.sun.star.chart.ChartArea` stöder linje- och fyllningsegenskaper men ger inga utökade egenskaper.

Diagrammens sid- och basytor

Även om diagrambakgrunden täcker hela diagramområdet begränsas diagrammets sidyta till området direkt bakom dataområdet.

3D-diagram har oftast två sidytor: en bakom dataområdet och en som avgränsare till vänster mot Y-axeln. 3D-diagram brukar också ha en basyta.

- **`Basyta (Object)`** – diagrammets basytepanel (endast 3D-diagram, stöder tjänsten `com.sun.star.chart.ChartArea`).
- **`Sidyta (Object)`** – diagrammets sidytor (endast 3D-diagram, stöder tjänsten `com.sun.star.chart.ChartArea`).

De objekt som angetts stöder tjänsten `com.sun.star.chart.ChartArea` som i sin tur tillhandahåller de vanliga fyllnings- och linjeegenskaperna (tjänsterna `com.sun.star.drawing.FillProperties` och `com.sun.star.drawing.LineStyle`, se kapitel 8).

Åtkomst till diagrammets sidytor och basyta sker via objektet `Diagram` som i sin tur är en del av objektet `Diagram`:

```
Chart.Area.FillBitmapName = "Himmel"
```

I följande exempel visas hur ett grafiskt objekt (kallat Himmel) som redan finns i StarOffice kan användas som diagrambakgrund.

```
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("MittDiagram", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MittDiagram").EmbeddedObject

Chart.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = "Himmel"
```

```
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Axlar

StarOffice känner igen fem olika axlar som kan användas i diagram. I de enklaste diagrammen är detta X- och Y-axlarna. När du arbetar med 3D-diagram behövs ibland en Z-axel. För diagram där värdena i de olika dataraderna varierar kraftigt använder StarOffice ytterligare en X- och Y-axel för skalningsåtgärder.

Den första X-, Y- och Z-axeln

Utöver den aktuella axeln kan det för varje första X-, Y- och Z-axel finnas en rubrik, en beskrivning, ett gitter och ett stödgitter. Det finns alternativ som styr om dessa element ska visas eller döljas. Diagramobjektet innehåller följande egenskaper som kan användas för hantering av de här funktionerna (om det gäller t.ex. en X-axel så är egenskaperna för Y- och Z-axeln strukturerade på samma sätt):

- **`HasXAxis (Boolean)`** – aktiverar X-axeln.
- **`XAxis (Object)`** – objekt med detaljerad information om X-axeln (stöder tjänsten `com.sun.star.chart.ChartAxis`).
- **`HasXAxisDescription (Boolean)`** – aktiverar beskrivningen för X-axeln.
- **`HasXAxisGrid (Boolean)`** – aktiverar huvudgittret för X-axeln.
- **`XMainGrid (Object)`** – objekt med detaljerad information om X-axelns huvudgitter (stöder tjänsten `com.sun.star.chart.ChartGrid`).
- **`HasXAxisHelpGrid (Boolean)`** – aktiverar stödgittret för X-axeln.
- **`XHelpGrid (Object)`** – objekt med detaljerad information om X-axelns stödgitter (stöder tjänsten `com.sun.star.chart.ChartGrid`).
- **`HasXAxisTitle (Boolean)`** – aktiverar X-axelns rubrik.
- **`XAxisTitle (Object)`** – objekt med detaljerad information om X-axelns rubrik (stöder tjänsten `com.sun.star.chart.ChartTitle`).

Den andra X- och Y-axeln

Följande egenskaper är tillgängliga för den andra X- och Y-axeln (egenskaperna exemplifieras med den andra X-axeln):

- **`HasSecondaryXAxis (Boolean)`** – aktiverar den andra X-axeln.
- **`SecondaryXAxis (Object)`** – objekt med detaljerad information om den andra X-axeln (stöder tjänsten `com.sun.star.chart.ChartAxis`).
- **`HasSecondaryXAxisDescription (Boolean)`** – aktiverar beskrivningen för X-axeln.

Axelns egenskaper

Axelobjekten för ett StarOffice-diagram stöder tjänsten `com.sun.star.chart.ChartAxis`. Utöver egenskaperna för tecken (tjänsten `com.sun.star.style.CharacterProperties`, se kapitel 6) och linjer (tjänsten `com.sun.star.drawing.LineStyle`, se kapitel 8), medföljer de här egenskaperna:

- **`Max (Double)`** – axelns maxvärde.
- **`Min (Double)`** – axelns minvärde.
- **`Originalvärde (Double)`** – skärningspunkten där axlarna korsar varandra.
- **`StepMain (Double)`** – avståndet mellan två primära linjer för axeln.
- **`StepHelp (Double)`** – avståndet mellan två sekundära linjer för axeln.
- **`AutoMax (Boolean)`** – bestämmer automatiskt axelns maxvärde.
- **`AutoMin (Boolean)`** – bestämmer automatiskt axelns minvärde.
- **`AutoOrigin (Boolean)`** – bestämmer automatiskt skärningspunkten där axlarna korsar varandra.
- **`AutoStepMain (Boolean)`** – bestämmer automatiskt avståndet mellan en axels primära linjer.
- **`AutoStepHelp (Boolean)`** – bestämmer automatiskt avståndet mellan en axels sekundära linjer.
- **`Logaritmisk (Boolean)`** – skalar axeln logaritmiskt (snarare än linjärt).
- **`DisplayLabels (Boolean)`** – aktiverar axlarnas textetiketter.
- **`TextRotation (Long)`** – rotationsvinkel för axelns textetikett i hundradelar av en grad.
- **`Marks (Const)`** - konstant som anger om axelns primära linjer ska vara innanför eller utanför diagramområdet (standardvärden i överensstämmelse med `com.sun.star.chart.ChartAxisMarks`)
- **`HelpMarks (Const)`** – konstant som anger om axelns sekundära linjer ska vara innanför och / eller utanför diagramområdet (standardvärden i överensstämmelse med `com.sun.star.chart.ChartAxisMarks`)
- **`Overlap (Long)`** – procenttal som anger hur mycket staplarna i olika datauppsättningar får överlappa varandra (vid 100 % visas staplarna helt överlappade, vid -100 % är avståndet mellan staplarna en stapel bred).
- **`GapWidth (long)`** – procenttal som anger avståndet som får finnas mellan olika stapelgrupper i ett diagram (vid 100 % motsvarar avståndet bredden på en stapel).
- **`ArrangeOrder (enum)`** – detaljer för påskriftens placering. Utöver placeringen på en linje finns även alternativet att dela etiketten över två linjer (standardvärde i överensstämmelse med `com.sun.star.chart.ChartAxisArrangeOrderType`)
- **`TextBreak (Boolean)`** – möjliggör linjebrytningar.

- **`TextCanOverlap (Boolean)`** – möjliggör överlappande text.
- **`NumberFormat (Long)`** – talformat (se kapitel 7, ***Tal-, datum- och textformat***)

Egenskaper för axelgittret

Objektet för axelgittret baseras på tjänsten `com.sun.star.chart.ChartGrid` som i sin tur stöder linjeegenskaperna för stödtjänsten `com.sun.star.drawing.LineStyle` (se kapitel 8).

Egenskaper för axelrubriken

Objekten som används för att formatera axelrubriken baseras på tjänsten `com.sun.star.chart.ChartTitle` som även används för diagramrubriker.

Exempel

I följande exempel skapas ett linjediagram. Färgen för diagrammets bakre sidyta har angetts till vit. Både X- och Y-axeln har grå stödgitter för att underlätta den visuella orienteringen. Minvärdet för X-axeln har angetts till det fasta värdet 0 och maxvärdet till det fasta värdet 100. På detta sätt bibehålls diagrammets upplösning även om värdena ändras.

```
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("MittDiagram", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MittDiagram").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-diagram

De flesta diagram i StarOffice kan även visas med 3D-grafik. Alla diagramtyper som det här alternativet är tillgängligt för stöder tjänsten `com.sun.star.chart.Dim3DDiagram`. Med tjänsten följer endast en egenskap:

- **`Dim3D (Boolean)`** – aktiverar 3D-visning.

Staplade diagram

Staplade diagram är diagram som innehåller flera enskilda värden ordnade ovanpå varandra som ger ett totalt värde. Den här vyn visar inte bara de enskilda värdena utan ger också en översikt över alla värden.

I StarOffice kan olika typer av diagram visas som ett staplat diagram. Följande diagram stöder tjänsten `com.sun.star.chart.StackableDiagram` som i sin tur tillhandahåller de här egenskaperna:

- **`Stacked (Boolean)`** – aktiverar det staplade visningsläget.
- **`Percent (Boolean)`** – visar värdedistributionen i procent i stället för i absoluta värden.

Diagramtyper

Linjediagram

Linjediagram (tjänsten `com.sun.star.chart.LineDiagram`) stöder en X-axel, två Y-axlar och en Z-axel. De kan visas i både 2D- och 3D-grafik (tjänsten `com.sun.star.chart.Dim3Ddiagram`). Linjerna kan staplas (`com.sun.star.chart.StackableDiagram`).

Med linjediagram följer de här egenskaperna:

- **`SymbolType (const)`** – symbol för visning av datapunkterna (konstant i överensstämmelse med `com.sun.star.chart.ChartSymbolType`).
- **`SymbolSize (Long)`** – storleken på symbolen för visning av datapunkterna i hundradelar av en millimeter.
- **`SymbolBitmapURL (String)`** – filnamn på grafik som används för visning av datapunkterna.
- **`Linjer (Boolean)`** – länkar datapunkterna med hjälp av linjer.
- **`SplineType (Long)`** – spline-funktion för utjämning av linjerna (0: ingen spline-funktion, 1: kubiska splines, 2: B-splines).

- **`SplineOrder (Long)`** – polynom vikt för splines (endast för B-splines).
- **`SplineResolution (Long)`** – antal stödpunkter för splineberäkning.

Ytdiagram

Ytdiagram (tjänsten `com.sun.star.chart.AreaDiagram`) stöder en X-axel, två Y-axlar och en Z-axel. De kan visas i både 2D- och 3D-grafik (tjänsten `com.sun.star.chart.Dim3Ddiagram`). Områdena kan staplas (`com.sun.star.chart.StackableDiagram`).

Stapeldiagram

Stapeldiagram (tjänsten `com.sun.star.chart.BarDiagram`) stöder en X-axel, två Y-axlar och en Z-axel. De kan visas i både 2D- och 3D-grafik (tjänsten `com.sun.star.chart.Dim3Ddiagram`). Staplarna kan staplas (`com.sun.star.chart.StackableDiagram`).

De här egenskaperna finns:

- **`Vertikalt (Boolean)`** – visar staplarna vertikalt, annars visas de horisontellt.
- **`Deep (Boolean)`** – i 3D-visningsläge placeras staplarna bakom varandra i stället för bredvid varandra.
- **`StackedBarsConnected (Boolean)`** – länkar de associerade staplarna i ett staplat diagram med hjälp av linjer (endast tillgängligt för horisontella diagram).
- **NumberOfLines (Long)** – antal linjer som ska visas som linjer i ett staplat diagram i stället för som staplar.

Cirkeldiagram

Cirkeldiagram (`com.sun.star.chart.PieDiagram`) innehåller inga axlar och kan inte staplas. De kan visas i både 2D- och 3D-grafik (tjänsten `com.sun.star.chart.Dim3Ddiagram`).

KAPITEL 10

Databasåtkomst

StarOffice har ett integrerat databasgränssnitt (oberoende av system) som kallas Star Database Connectivity (SDBC). Avsikten med utvecklandet av det här gränssnittet har varit att ge åtkomst till så många olika datakällor som möjligt.

Detta har gjorts möjligt genom att åtkomst till datakällorna sker via drivrutiner. Vilka källor som drivrutinerna får sina data ifrån är irrelevant för SDBC-användaren. En del drivrutiner använder filbaserade databaser och får data direkt därifrån. För andra drivrutiner används gränssnitt som t.ex. JDBC eller ODBC. Det finns även speciella drivrutiner som får åtkomst till MAPI-adressboken, LDAP-kataloger och StarOffice-tabeller som datakällor.

Eftersom drivrutinerna baseras på UNO-komponenter går det att utveckla andra drivrutiner som öppnar nya datakällor. Mer information om det finns i StarOffice Developer's Guide.

Som koncept kan SDBC jämföras med ADO- och DAO-biblioteken som är tillgängliga i VBA. Den möjliggör högnivååtkomst till databaser oberoende av de underliggande databaserna.

Databasgränssnittet i StarOffice har blivit större i och med lanseringen av StarOffice 7. Tidigare skedde åtkomsten till databaserna primärt via olika metoder för objektet `Program`, medan gränssnittet i StarOffice 7 delas upp i flera objekt. En `DatabaseContext` används som rotobjekt för databasfunktionerna.

SQL: ett sökningsspråk

I SDBC används sökningsspråket SQL. För att lösa problemet med olika SQL-dialekter har SDBC-komponenterna i StarOffice försetts med en egen SQL-tolk. Den kontrollerar de SQL-kommandon som matats in i sökningsfönstret och korrigerar enkla syntaxfel, t.ex. stora och små bokstäver.

Om en drivrutin ger åtkomst till en datakälla som inte stöder SQL måste de SQL-kommandon som överförs konverteras till den inbyggda åtkomstmetoden.

SQL-implementeringen i SDBC ligger närmast SQL-ANSI-standarden. Tillägg som är specifika för Microsoft, t.ex. `INNER JOIN`, stöds inte. Sådana tillägg bör ersättas med standardkommandon (`INNER JOIN`, bör t.ex. ersättas med ett motsvarande `WHERE`-uttryck).

Typer av databasåtkomst

Databasgränssnittet i StarOffice är tillgängligt i StarOffice Writer och StarOffice Calc liksom i databasformulären.

I StarOffice Writer kan du med hjälp av SDBC-datakällor skapa standardbrev som sedan kan skrivas ut. Det finns också ett alternativ som gör att det går att flytta data från databasfönstret till textdokumentet genom att dra-och-släppa.

Om användaren flyttar en databastabell till ett tabelldokument skapas ett tabellområde som kan uppdateras med en musklickning om ursprungsinformationen har ändrats. I motsatt rikting kan tabelldata flyttas till en databastabell, d.v.s. en databasimport.

Slutligen finns det i StarOffice en mekanism för formulär baserade på databaser. Användaren skapar först ett standardformulär för StarOffice Writer eller StarOffice Calc och länkar sedan fälten till en databas.

Alla alternativ som angetts här baseras på användargränssnittet i StarOffice. Det behövs ingen kunskap om programmering för att använda funktionerna.

I det här kapitlet finns däremot sparsamt med information om funktionerna som angetts. I stället sätta fokus på programmeringsgränssnittet i SDBC som möjliggör automatiserade databassökningar och därför ger utrymme för användandet av ett bredare urval av program.

Grundläggande kunskaper om SQL och hur databaser fungerar behövs om du vill tillgodogöra dig innehållet i följande avsnitt helt och hållet.

Datakällor

Du integrerar en databas i StarOffice genom att skapa något som brukar kallas en ***datakälla***. I användargränssnittet finns ett alternativ i **Extra**-menyn som du kan använda för att skapa datakällor. Du kan även skapa datakällor och arbeta med dem genom att använda StarOffice Basic.

Ett databaskontextobjekt som skapats med funktionen `createUnoService` fungerar som startpunkten för åtkomst till en datakälla. Detta baseras på tjänsten `com.sun.star.sdb.DatabaseContext` och är rotobjektet för alla databasåtgärder.

I följande exempel visas hur en databaskontext kan skapas och användas för att bestämma namnen på alla tillgängliga datakällor. Namnen visas i en meddelanderuta.

```
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
```

Den enskilda datakällorna baseras på tjänsten `com.sun.star.sdb.DataSource` och kan bestämmas från databaskontexten med hjälp av metoden `getByName`:

```
Dim DatabaseContext As Object
Dim DataSource As Object
```

```
DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Kunder")
```

I exemplet skapas ett `DataSource`-objekt för en datakälla som kallas ***Kunder***.

För datakällorna finns det en rad olika egenskaper som i sin tur ger allmän information om dataursprunget och information om åtkomstmetoder. Egenskaperna är:

- **`Namn (String)`** – datakällans namn.
- **`URL (String)`** – datakällans URL i formatet ***jdbc: underprotokoll: delnamn*** eller ***sdbc: underprotokoll: delnamn***.
- **`Info (Array)`** – matris som innehåller `PropertyValue`-par med anslutningsparametrar (oftast åtminstone användarnamn och lösenord).
- **`Användare (String)`** – användarnamn.
- **`Lösenord (String)`** – användarens lösenord (sparas inte).
- **`IsPasswordRequired (Boolean)`** – lösenordet krävs och användaren ombeds interaktivt att ange det.
- **`IsReadOnly (Boolean)`** – möjliggör skrivskyddad åtkomst till databasen.
- **`NumberFormatsSupplier (Object)`** – objekt som innehåller de talformat som är tillgängliga för databasen (stöder gränssnittet `com.sun.star.util.XNumberFormatsSupplier`, se kapitel 7, ***Tal-, datum- och textformat***).
- **`TableFilter (Array)`** – lista över tabellnamn som ska visas.
- **`TableTypeFilter (Array)`** – lista över tabelltyper som ska visas. De värden som är tillgängliga är `TABLE`, `VIEW` och `SYSTEM TABLE`.
- **`SuppressVersionColumns (Boolean)`** – begränsar visningen av kolumner som används för versionshantering.

Datakällorna från StarOffice är inte helt jämförbara med datakällorna i ODBC. Där en ODBC-datakälla bara täcker information om dataursprunget inkluderar StarOffice-datakällan även information om hur data visas i databasfönstret i StarOffice.

Sökningar

Det går att tilldela en datakälla fördefinierade sökningar. SQL-kommandona för sökningarna noteras i StarOffice så att de alltid är tillgängliga. Sökningar förenklar databasarbetet eftersom de kan öppnas med en musklickning och gör att användare utan kunskap om SQL kan använda SQL-kommandon.

Ett objekt som stöder tjänsten `com.sun.star.sdb.QueryDefinition` döljs bakom en sökning. Åtkomst till sökningar sker via metoden `QueryDefinitions` för datakällan.

I följande exempel visas en lista i en meddelanderuta med namnen på de datakällsökningar som kan upprättas:

```
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("Kunder")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
      QueryDefinition = QueryDefinitions(I)
      MsgBox QueryDefinition.Name
Next I
```

Utöver egenskapen `Namn` som används i exemplet ger `com.sun.star.sdb.QueryDefinition` en hel rad andra egenskaper. Dessa är:

- **`Namn (String)`** – sökningens namn.
- **`Kommando (String)`** – SQL-kommandot (oftast ett `SELECT`-kommando).
- **`UpdateTableName (String)`** – för sökningar som använder flera tabeller: namn på tabell där det går att ändra värdena.
- **`UpdateCatalogName (String)`** – namn på kataloger för uppdateringstabeller.
- **`UpdateSchemaName (String)`** – namn på diagram för uppdateringstabeller.

I följande exempel visas hur ett sökningsobjekt kan skapas på ett programstyrt sätt och tilldelas en datakälla.

```
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("Kunder")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELECT * FROM Kunder"

QueryDefinitions.insertByName("NySökning", QueryDefinition)
```

Sökningsobjektet skapas först med anropet `createUnoService`, sedan initieras det och infogas i objektet `QueryDefinitions` med hjälp av `insertByName`.

Länkar med databasformulär

För att underlätta arbetet med datakällor finns det i StarOffice ett alternativ som används för att länka datakällorna med databasformulär. Länkarna görs tillgängliga via metoden `getBookmarks()`. Den returnerar en namngiven container (`com.sun.star.sdb.DefinitionContainer`) som innehåller alla länkar för datakällan. Åtkomst till bokmärkena kan ske via antingen `Namn` eller `Index`.

I följande exempel bestäms URL:en för bokmärket ***MittBokmärke***.

```
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("Kunder")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("MittBokmärke")
MsgBox URL
```

Databasåtkomst

För få åtkomst till en databas krävs en databasanslutning. Det är en överföringskanal som möjliggör direktkommunikation med databasen. Till skillnad från datakällorna som presenterades i det föregående avsnittet måste databasanslutningen upprättas på nytt varje gång programmet startas om.

I StarOffice finns det olika sätt att upprätta databasanslutningar. Här följer en förklaring av metoden som baseras på en befintlig datakälla.

```
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("Kunder")

If Not DataSource.IsPasswordRequired Then
      Connection = DataSource.GetConnection("","")
Else
      InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
      Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If
```

Med koden som används i exemplet kontrolleras först om databasen är lösenordsskyddad. Om den inte är det skapas databasanslutningen med anropet `GetConnection`. De två tomma strängarna på kommandoraden står för användarnamnet och lösenordet.

Om databasen är lösenordsskyddad skapas i exemplet en `InteractionHandler` och databasanslutningen öppnas med metoden `ConnectWithCompletion`. Denna InteractionHandler gör att StarOffice frågar användaren efter de inloggningsdata som krävs.

Tabelluppreppningar

I StarOffice sker oftast åtkomsten till en tabell via objektet `ResultSet`. En `ResultSet` är en slags tecken som indikerar att en aktuell uppsättning data i en resultatvolym erhållits med kommandot `SELECT`.

I exemplet visas hur `ResultSet` kan användas för sökningar efter värden i en databastabell.

```
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("Kunder")

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 Kundnummer FROM Kunder")

If Not IsNull(ResultSet) Then
      While ResultSet.next
            MsgBox ResultSet.getString(1)
      Wend
End If
```

När databasanslutningen har upprättats används med koden i exemplet först anropet `Connection.createObject` för att skapa ett `Uttryck`-objekt. Detta `Uttryck`-objekt använder sedan anropet `executeQuery` för att anropa den faktiska `ResultSet`. Programmet kontrollerar sedan om `ResultSet` faktiskt finns och traverserar dataposterna genom att använda en loop. De värden som krävs (i exemplet är det de från fältet `Kundnummer`) returnerar `ResultSet` genom att använda metoden `getString`, där parametern 1 bestämmer att anropet gäller värdena i första kolumnen.

Objektet `ResultSet` i SDBC kan jämföras med objektet `Recordset` i DAO och ADO eftersom dessa också ger upprepad åtkomst till en databas.

Åtkomst till databasen i StarOffice 7.x sker faktiskt via ett `ResultSet`-objekt. Detta återspeglar innehållet i en tabell eller resultatet av ett SQL-SELECT-kommando. Tidigare tillhandahöll objektet `ResultSet` de inbyggda metoderna i objektet `Program` för navigering bland data (d.v.s. `DataNextRecord`).

Typspecifika metoder för hämtning av värden

Så som framgått i exemplet i det föregående avsnittet finns det i StarOffice en `getString`-metod som används för åtkomst till tabellinnehåll. Metoden ger resultatet i form av en sträng. Följande `get`-metoder är tillgängliga:

- `getByte()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getShort()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getInt()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getLong()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getFloat()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getDouble()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getBoolean()` – stöder SQL-datatyperna för tal, tecken och strängar.
- `getString()` – stöder alla SQL-datatyper.
- `getBytes()` – stöder SQL-datatyperna för binära värden.
- `getDate()` – stöder SQL-datatyperna för tal, strängar, datum och tidsstämpel.
- `getTime()` – stöder SQL-datatyperna för tal, strängar, datum och tidsstämpel.
- `getTimestamp()` – stöder SQL-datatyperna för tal, strängar, datum och tidsstämpel.
- `getCharacterStream()` – stöder SQL-datatyperna för tal, strängar och binära värden.
- `getUnicodeStream()` – stöder SQL-datatyperna för tal, strängar och binära värden.
- `getBinaryStream()` – binära värden.
- `getObject()` – stöder alla SQL-datatyper.

För samtliga gäller att antalet kolumner ska listas som en parameter vars värden ska sökas.

ResultSet-varianterna

Det är ofta viktigt att åtkomsten till databasen är snabb. Därför finns det i StarOffice flera sätt att optimera `ResultSets` och därmed kontrollera åtkomsthastigheten. Ju fler funktioner som `ResultSet` tillhandahåller desto mer komplicerad blir implementeringen och funktionerna långsammare.

En enkel `ResultSet`, t.ex. den som presenterades i avsnittet "Tabelluppreppningar" ger det minsta urvalet av funktioner. Den möjliggör endast upprepningar framåt och sökningar efter värden. Navigeringsalternativ som t.ex. möjligheten att ändra värden är därför inte inkluderade.

Objektet `Uttryck` som används för att skapa `ResultSet` tillhandahåller en del egenskaper som gör att funktionerna för `ResultSet` kan påverkas:

- **`ResultSetConcurrency (const)`** – specifikationer som anger om data kan ändras eller inte (i överensstämmelse med `com.sun.star.sdbc.ResultSetConcurrency`).
- **`ResultSetType (const)`** – specifikationer angående typ av `ResultSets` (i överensstämmelse med `com.sun.star.sdbc.ResultSetType`).

Värdena som definierats i `com.sun.star.sdbc.ResultSetConcurrency` är:

- **`UPDATABLE`** – `ResultSet` möjliggör att värden ändras.
- **`READ_ONLY`** – `ResultSet` ändringar är inte möjliga.

Gruppen med konstanter, `com.sun.star.sdbc.ResultSetConcurrency`, ger följande specifikationer:

- **`FORWARD_ONLY`** – `ResultSet` endast framåtnavigering är möjlig.
- **`SCROLL_INSENSITIVE`** – `ResultSet` möjliggör all navigering, ändringar av ursprungsdata noteras däremot inte.
- **`SCROLL_SENSITIVE`** – `ResultSet` möjliggör all navigering, ändringar av ursprungsdata påverkar `ResultSet`.

En `ResultSet` som innehåller egenskaperna `READ_ONLY` och `SCROLL_INSENSITIVE` motsvarar en postuppsättning för `Snapshot`-typen i ADO och DAO.

När du använder `ResultSet-egenskaperna` `UPDATEABLE` och `SCROLL_SENSITIVE` går, urvalet av funktioner för en `ResultSet` att jämföras med `Dynaset`-typen `Recordset` i ADO och DAO.

Navigeringsmetoder i ResultSets

Om en `ResultSet` är av typen `SCROLL_INSENSITIVE` eller `SCROLL_SENSITIVE` stöder den en hel rad metoder för datanavigering. De centrala metoderna är:

- **`next()`** – navigering till nästa datapost.
- **`previous()`** – navigering till föregående datapost.
- **`first()`** – navigering till den första dataposten.
- **`last()`** – navigering till den sista dataposten.
- **`beforeFirst()`** – navigering till före den första dataposten.
- **`afterLast()`** – navigering till efter den sista dataposten.

Alla metoder returnerar en logisk parameter som anger om navigeringen lyckades eller inte.

För att kunna bestämma den aktuella markörpositionen används följande testmetoder och alla returnerar ett logiskt värde:

- **`isBeforeFirst()`** – `ResultSet` är före den första dataposten.
- **`isAfterLast()`** – `ResultSet` är efter den sista dataposten.
- **`isFirst()`** – `ResultSet` är den första dataposten.
- **`isLast()`** – `ResultSet` är den sista dataposten.

Ändra dataposter

Om en `ResultSet` har skapats med värdet `ResultSetConcurrency = UPDATEABLE` kan dess innehåll redigeras. Detta gäller bara så länge som SQL-kommandot tillåter att data skrivs om till databasen (beroende på princip). Detta är inte möjligt med komplicerade SQL-kommandon med länkade kolumner eller ackumulerade värden.

Objektet `ResultSet` tillhandhåller `Update`-metoder för att ändra värden som är strukturerade på samma sätt som `get`-metoder för att hämta värden. Metoden `updateString` medger till exempel att en sträng skrivs.

Efter att de ändrats måste värdena först överföras till databasen med metoden `updateRow()`. Anropet måste ske innan nästa navigeringskommando annars går värdena förlorade.

Om ett fel görs under ändringarna går det att ångra med metoden `cancelRowUpdates()`. Det här anropet är bara tillgängligt om data inte har skrivits om till databasen med `updateRow()`.

Dialogrutor

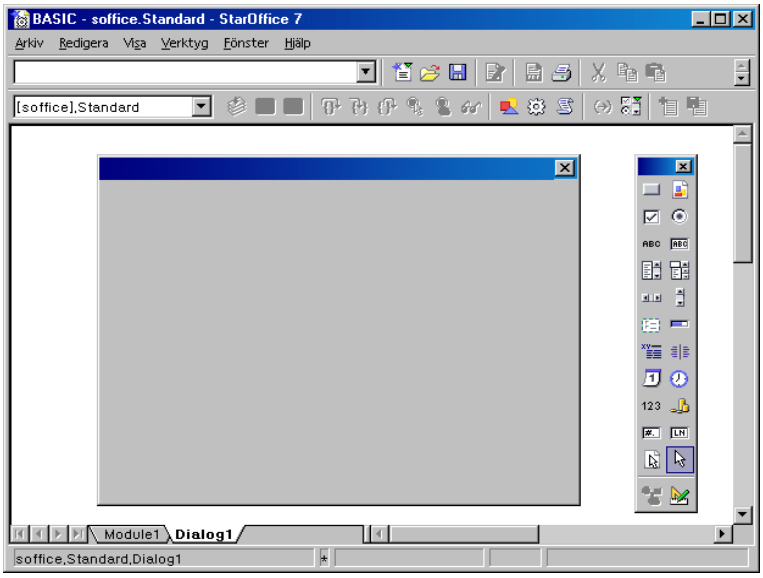
Du kan lägga till användardefinierade dialogrutor och formulär i StarOffice-dokument. De kan i sin tur länkas till StarOffice Basic-makron vilket ökar användningsområdet för StarOffice Basic. Dialogrutor kan t.ex. visa databasinformation eller leda användare genom stegvisa processer som exempelvis att skapa ett nytt dokument med AutoPiloten.

Arbeta med dialogrutor

Dialogrutorna i StarOffice Basic består av en dialogruta som kan innehålla textfält, listrutor, alternativfält och andra kontrollelement.

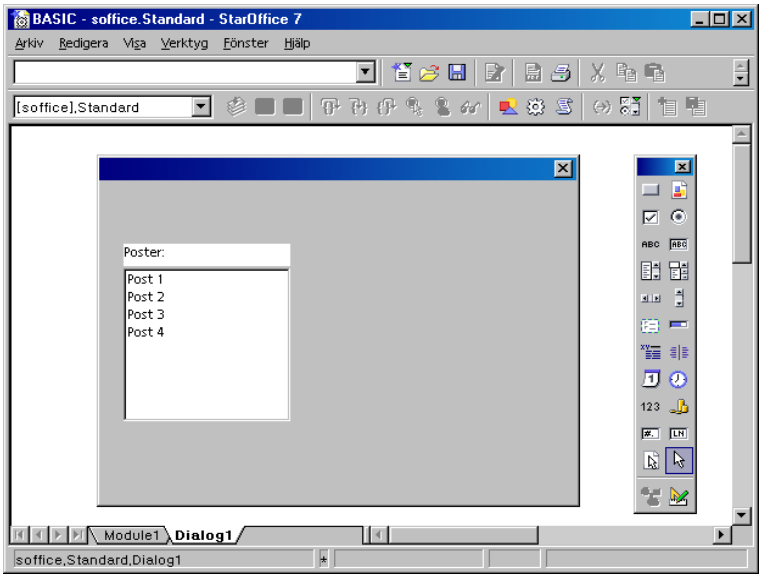
Skapa dialogrutor

Du kan skapa och strukturera dialogrutor genom att använda StarOffice-dialogruteredigeraren, som du kan använda på samma sätt som en StarOffice Draw:



Du drar kontrollelementen som du vill använda från designpaletten (till höger) till dialogområdet där du kan definiera position och storlek.

I exemplet visas en dialogruta som innehåller en etikett och en listruta.



Du kan öppna en dialogruta med den här koden:

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` skapar ett objekt som kallas `Dlg` som refererar till den associerade dialogrutan. Innan du skapar dialogrutan måste du kontrollera att biblioteket som använder den (i det här exemplet biblioteket `Standard`) är laddat. Om det inte är det utför metoden `LoadLibrary` den här uppgiften.

När dialogobjektet `Dlg` har initierats kan du använda metoden `Execute` för att visa dialogrutan. Dialogrutor som den här beskrivs som modala eftersom de inte tillåter andra programåtgärder förrän de stängs. Medan den här dialogrutan är öppen förblir programmet i anropet `Execute`.

Disponeringsmetoden `dispose` i slutet av koden godkänner resurserna som används av dialogrutan när programmet avslutas.

Stänga dialogrutor

Stänga med OK eller Avbryt

Om en dialogruta innehåller knappen **OK** eller **Avbryt** stängs dialogrutan automatiskt när du trycker på en av knapparna. Mer information om hur du arbetar med de här knapparna finns i avsnittet Kontrollelement i dialogrutor i det här kapitlet.

Om du stänger en dialogruta genom att klicka på knappen **OK** returnerar metoden `Utför` värdet `1` i stället för 0.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
      MsgBox "OK trycktes ned"
Case 0
      MsgBox "Avbryt trycktes ned"
End Select
```

Stänga med knappen Stäng på titellisten

Om du vill kan du stänga en dialogruta genom att klicka på knappen Stäng på dialogrutans titellist. I det här fallet returnerar metoden `Utför` för dialogrutan värdet 0, d.v.s. samma värde som när du trycker på knappen Avbryt.

Stänga med ett explicit programanrop

Du kan även stänga en öppen dialogruta med metoden `endExecute`:

```
Dlg.endExecute()
```

Åtkomst till enskilda kontrollelement

En dialogruta kan innehålla ett obegränsat antal kontrollelement. Du får åtkomst till de här elementen via metoden `getControl` som returnerar namnet på kontrollelementet.

```
Dim Ctl As Object

Ctl = Dlg.getControl("MinKnapp")
Ctl.Label = "Ny etikett"
```

Med den här koden bestämsobjekt för kontrollelementet `MinKnapp` och sedan initieras objektvariabeln `Ctl` med en referens till elementet. Slutligen anges egenskapen `Label` för kontrollelementet till värdet `Ny etikett`.

Lägg märke till att StarOffice Basic skiljer på stora och små bokstäver i namnen för kontrollelementen.

Arbeta med *modeller* för dialogrutor och kontrollelement

Uppdelningen mellan synliga programelement (***Visa***) och de data eller dokument som ligger bakom (***Modell***) inträffar på många ställen i StarOffice API. Utöver metoderna och egenskaperna för kontrollelementen, har både dialog- och kontrollelementobjekt underordnade `Modell`-objekt. Det här objektet gör det möjligt för dig att få direkt åtkomst till innehållet i en dialogruta eller ett kontrollelement.

När det gäller dialogrutor är skillnaden mellan data och avbildning inte alltid lika tydlig som för andra API-områden i StarOffice. Element av API:t är tillgängliga via både Visa och Modell.

Egenskapen `Modell` ger programkontrollerad åtkomst till modellen för dialog- och kontrollelementobjekt.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

I det här exemplet inaktiveras knappen `cmdNext` i dialogrutan `Dlg` med hjälp av modellobjektet från `cmdNext`.

Egenskaper

Namn och rubrik

Alla kontrollelement har egna namn som det går att söka efter med följande modellegenskap:

- **`Model.Name (String)`** – kontrollelementnamn

Du kan ange vilken rubrik som visas i dialogrutans titellist med den här modellegenskapen:

- **`Model.Title (String)`** – dialogrutenamn (gäller endast dialogrutor).

Position och storlek

Du kan söka efter storlek och position för ett kontrollelement med följande egenskaper för modellobjektet:

- **`Model.Height (long)`** – kontrollelementets höjd (i ma-enheter)
- **`Model.Width (long)`** – kontrollelementets bredd (i ma-enheter)
- **`Model.PositionX (long)`** – kontrollelementets X-position, mätt från dialogrutans inre vänstra kant dialog (i ma-enheter)
- **`Model.PositionY (long)`** – kontrollelementets Y-position, mätt från dialogrutans övre inre kant dialog (i ma-enheter)

För att se till att dialogrutornas utseende är plattformsoberoende används i StarOffice den interna enheten ***Map AppFont (ma)*** för att ange position och storlek i dialogrutor. En ma-enhet definieras som en åttondel av medelhöjden för det systemteckensnitt som angetts för operativsystemet och en

fjärdedel av dess bredd. Genom att använda ma-enheter ser StarOffice-dialogrutor likadana ut på olika system under olika systeminställningar.

Om du vill ändra storleken eller positionen för kontrollelement i runtime bestämmer du den totala storleken för dialogrutan och justerar värdena för kontrollelementen i förhållande till denna.

Map AppFont (ma) ersätter enheten Twips för att förbättra plattformsoberoendet.

Fokus och tabbsekvens

I alla dialogrutor kan du navigera genom kontrollelementen genom att använda Tab-tangenten. Följande egenskaper är tillgängliga i kontrollelementmodellen i det här sammanhanget:

- **`Model.Enabled (Boolean)`** – aktiverar kontrollelementet
- **`Model.Tabstop (Boolean)`** – gör att kontrollelementet kan nås via Tab-tangenten
- **`Model.TabIndex (Long)`** – kontrollelementets placering i aktiveringsordningen

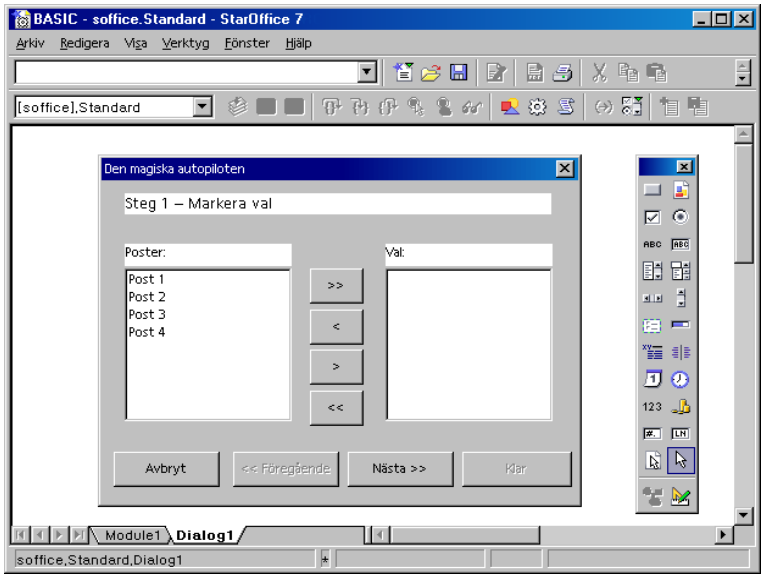
Slutligen tillhandahåller kontrollelementet en `getFocus`-metod som ser till att de underliggande kontrollelementen tar emot fokus:

- **`getFocus`** – kontrollelementet tar emot fokus (gäller endast dialogrutor)

Flersidiga dialogrutor

En dialogruta i StarOffice kan ha en eller flera flikar. Egenskapen `Step` för en dialogruta definierar den aktuella fliken för dialogrutan medan egenskapen `Step` för ett kontrollelement anger vilken flik som kontrollelementet ska visas på.

`Step`-värdet `0` är ett specialfall. Om du anger det här värdet till noll i en dialogruta visas alla kontrollelement oavsett vilket `Step`-värde de har. Om du anger det här värdet till noll för ett kontrollelement visas elementet på alla flikar i en dialogruta.



I det ovanstående exemplet kan du även tilldela `Step`-värdet `0` för delningslinjen liksom för knapparna `Avbryt`, `Föregående`, `Nästa` och `Klar` om du vill visa de här elementen på alla sidor. Du kan även placera elementen på en enskild flik (t.ex. fliken `1`).

Följande programkod visar hur värdet `Step` i händelsehanterare för knapparna `Nästa` och `Föregående` kan ökas eller minskas och hur knapparnas status ändras.

```
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
```

En global `Dlg`-variabel som refererar till en öppen dialogruta måste inkluderas för att exemplet ska bli möjligt. Dialogrutan ändrar i så fall utseende på följande sätt:

Sida 1:

Sida 2:

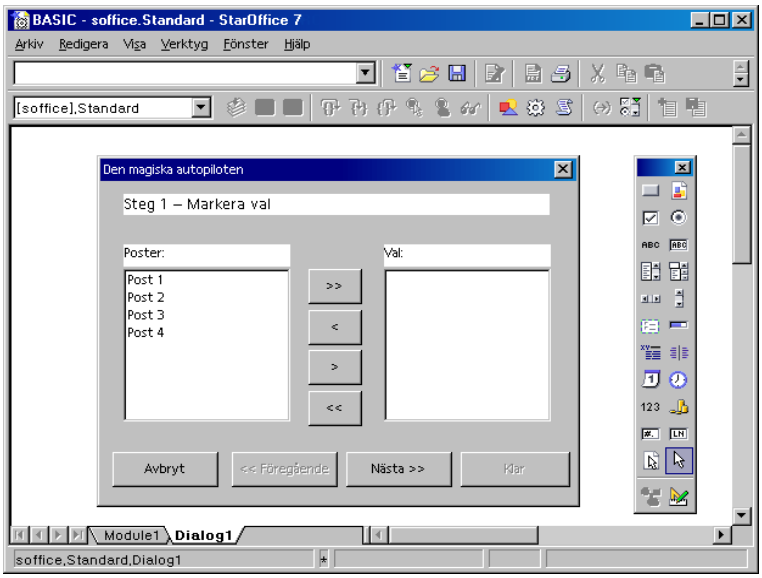
Händelser

StarOffice-dialogrutor och -formulär är baserade på en händelseorienterad programmeringsmodell där du kan tilldela ***händelsehanterare*** för kontrollelementen. En händelsehanterare kör en fördefinierad procedur när en viss åtgärd inträffar, även när åtgärden är en annan händelse. Du kan även redigera dokument eller öppna databaser med händelsehantering liksom få åtkomst till andra kontrollelement.

Kontrollelement i StarOffice känner igen olika typer av händelser som kan utlösas i olika situationer. De här händelsetyperna kan delas in i fyra grupper:

- **Muskontroll**: Händelser som motsvarar musåtgärder (t.ex. enkla musrörelser eller en klickning på en viss plats på skärmbilden)
- **Tangentbordskontroll**: Händelser som utlöses av tryckningar på tangentbordet
- **Fokusändringar**: Händelser som utförs när kontrollelement aktiveras eller inaktiveras i StarOffice
- **Händelser som är specifika för kontrollelement**: Händelser som bara inträffar i relation till vissa kontrollelement

När du arbetar med händelser bör du se till att skapa de associerade dialogrutorna i StarOffice utvecklingsmiljö och att de innehåller de kontrollelement eller dokument som krävs (om du använder händelser för ett formulär).



Ovanstående figur visar utvecklingsmiljön i StarOffice Basic med en dialogruta som innehåller två listrutor. Du kan flytta data från en lista till den andra genom att använda knapparna mellan listrutorna.

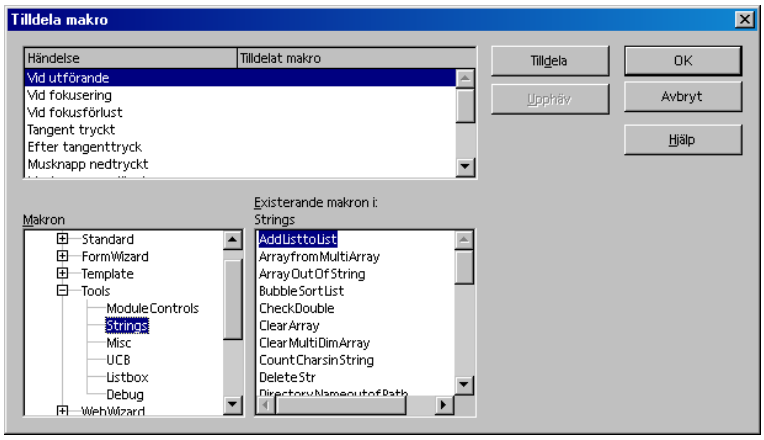
Om du vill visa layouten på skärmen bör du skapa de associerade StarOffice Basic-procedurerna så att de kan anropas av händelsehanterarna. Även om du kan använda de här procedurerna i vilken modul som helst är det ändå bäst att begränsa användningen till två moduler. Om du ger procedurerna namn som säger något om sammanhanget blir koden lättare att läsa. Ett hopp direkt till en allmän programprocedur från ett makro kan göra koden otydlig. I stället bör du skapa en annan procedur som fungerar som en startpunkt för händelsehanteringen, även om den bara kör ett enda anrop till målproceduren. Det underlättar både underhållet och felsökningen av koden.

Med koden i följande exempel flyttas en post från vänster till höger listruta i en dialogruta.

```
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.SelectedItemPos, 1)
      Else
            Beep
      End If
End Sub
```

Om den har proceduren har skapats i StarOffice Basic kan du koppla den till en händelse som krävs när egenskapsfönstret i dialogruteredigeraren används.



I tilldelningsdialogrutan listas alla StarOffice Basic-procedurer. Om du vill koppla en procedur till en händelse markerar du proceduren och klickar sedan på **Tilldela**.

Parametrar

Det är inte alltid tillräckligt med förekomsten av en viss händelse för att korrekt respons ska utlösas. Det kan krävs ytterligare information. Om du t.ex. vill utlösa en musklickning kan du behöva positionen på skärmen där musen ska klickas.

I StarOffice Basic används objektparametrar för att förmedla mer information om en händelse till en procedur, exempelvis:

```
Sub ProcessEvent(Event As Object)

End Sub
```

Med vilken precision som `Händelse`-objektet har strukturerats och vilka egenskaper det har beror på vilken typ av händelse som proceduranropet utlöser. I följande avsnitt beskrivs händelsetyperna i detalj.

Oavsett händelsetyp ger alla objekt åtkomst till det relevanta kontrollelementet och motsvarande modell. Du når kontrollelementet genom att använda

```
Event.Source
```

och modellen med

```
Event.Source.Model
```

Du kan använda de här egenskaperna om du vill utlösa en händelse i en händelsehanterare.

Mushändelser

StarOffice Basic känner igen följande mushändelser:

- **`Mus flyttas`** – användaren flyttar musen
- **`Mus flyttas medan tangent trycks ned`** – användaren drar musen medan en tangent hålls ned
- **`Musknapp trycks ned`** – användaren trycker på en musknapp
- **`Musknapp släpps`** – användaren släpper en musknapp
- **`Mus är utanför`** – användaren flyttar musen utanför det aktuella fönstret

De associerade händelseobjektens struktur definieras i strukturen `com.sun.star.awt.MouseEvent` som ger följande information:

- **`Knappar (short)`** – knapp trycks ned (en eller fler konstanter i överensstämmelse med `com.sun.star.awt.MouseButton`).
- **`X (long)`** – Musens X-koordinat mätt i pixlar från det övre vänstra hörnet av kontrollelementet
- **`Y (long)`** – Musens Y-koordinat mätt i pixlar från det övre vänstra hörnet av kontrollelementet
- **`ClickCount (long)`** – antal klickningar som ska associeras med mushändelsen (om StarOffice reagerar tillräckligt snabbt är ClickCount = 1 även för en dubbelklickning eftersom endast en enskild händelse initieras).

Konstanterna som definierats i `com.sun.star.awt.MouseButton` för musknapparna är:

- **`LEFT`** – vänster musknapp
- **`RIGHT`** – höger musknapp
- **`MIDDLE`** – mittenknappen på musen

Följande exempel resulterar i utdata som talar om muspositionen och vilken musknapp som trycktes ned.

```
Sub MouseUp(Event As Object)
      Dim Msg As String
      Msg = "Knappar: "
      If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
            Msg = Msg & "VÄNSTER "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
            Msg = Msg & "HÖGER "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
            Msg = Msg & "MITTEN "
      End If
      Msg = Msg & Chr(13) & "Position: "
      Msg = Msg & Event.X & "/" & Event.Y
      MsgBox Msg
End Sub
```

VBA-händelserna `Click` och `Doubleclick` är inte tillgängliga i StarOffice Basic. Använd i stället StarOffice Basic-händelsen `MouseUp` som `click`-händelsen och imitera `Doubleclick`-händelsen genom att ändra programlogiken.

Tangentbordshändelser

Följande tangentbordshändelser är tillgängliga i StarOffice Basic:

- **`Tangent trycks ned`** – användaren trycker ned en tangent
- **`Tangent släpps`** – användaren släpper en tangent

Båda händelserna relaterar till ***logiska*** tangentåtgärder och inte till ***fysiska*** åtgärder. Om användaren trycker ned flera tangenter för att mata ut ett enda tecken (t.ex. lägga till en accent) skapas bara en händelse i StarOffice Basic.

En enskild tangentåtgärd på en ändringstangent, t.ex. Skift- eller Alt-tangenten skapar inte en oberoende händelse.

Information om en tangent som trycks ned tillhandahålls av händelseobjektet som ges till proceduren för händelsehantering. Det innehåller de här egenskaperna:

- **`KeyCode (short)`** – kod för den nedtryckta tangenten (standardvärden i överensstämmelse med `com.sun.star.awt.Key`)
- **`KeyChar (String)`** – tecken som matas in (tar hänsyn till ändringstangenterna)

I följande exempel används egenskapen KeyCode för att ta reda på tangenterna Retur Tab eller en av de andra ctrl-tangenterna har tryckts ned. Om en av de här tangenterna har tryckts ned returneras namnet på tangenten, annars returneras tecknet som matats in:

```
Sub KeyPressed(Event As Object)
      Dim Msg As String
      Select Case Event.KeyCode
      Case com.sun.star.awt.Key.RETURN
            Msg = "Retur trycktes ned"
      Case com.sun.star.awt.Key.TAB
            Msg = "Tab trycktes ned"
      Case com.sun.star.awt.Key.DELETE
            Msg = "Del trycktes ned"
      Case com.sun.star.awt.Key.ESCAPE
            Msg = "Esc trycktes ned"
      Case com.sun.star.awt.Key.DOWN
            Msg = "Nedåtpil trycktes ned"
      Case com.sun.star.awt.Key.UP
            Msg = "Uppåtpil trycktes ned"
      Case com.sun.star.awt.Key.LEFT
            Msg = "Vänsterpil trycktes ned"
      Case com.sun.star.awt.Key.RIGHT
            Msg = "Högerpil trycktes ned"
      Case Else
            Msg = "Tecknet " & Event.KeyChar & " matades in"
      End Select
      MsgBox Msg
End Sub
```

Information om andra tangentbordskonstanter finns i API-referensen under konstantgruppen com.sun.star.awt.Key.

Fokushändelser

Fokushändelser indikerar om ett kontrollelement tar emot eller förlorar fokus. Du använder de här händelserna om du t.ex. vill bestämma om en användare har slutat processen av ett kontrollelement så att du kan uppdatera andra element för en dialogruta. Du kan använda följande fokushändelser:

- **När något tar emot fokus** – tar elementet emot fokus
- **När något förlorar fokus** – förlorar elementet fokus

Händelseobjekten för fokushändelserna är strukturerade så här:

- **FocusFlags (short) –** orsak till fokusförändringen (standardvärde i överensstämmelse med com.sun.star.awt.FocusChangeReason).
- **NextFocus (Object)** – objekt som tar emot fokus (gäller endast händelsen När något förlorar fokus)
- **Temporary (Boolean)** – fokus förloras tillfälligt

Händelser som är specifika för kontrollelement

Utöver de ovanstående händelserna som stöds av alla kontrollelement finns det även några händelser som är specifika för vissa definierade kontrollelement. De viktigaste av dessa är:

- **`När objekt ändras`** – värdet för ett kontrollelement ändras
- **`Objektstatus ändras`** – status för ett kontrollelement ändras
- **`Text ändras`** – texten för ett kontrollelement ändras
- **`Vid initiering`** – en åtgärd som kan utföras när kontrollelementet löses ut (t.ex. en knapp trycks ned)

När du arbetar med händelser bör du tänka på att en del händelser, som t.ex. `Vid initiering` kan initieras varje gång du klickar med musen på något kontrollelement (t.ex. alternativknappar). Ingen åtgärd utförs för att kontrollera om status för kontrollelementet verkligen har ändrats. Du undviker den här typen av "blinda händelser" genom att spara det gamla kontrollelementvärdet i en global variabel, sedan kontrollerar du om värdet har ändrats när en händelse körs.

Egenskaperna för händelsen `Objektstatus ändras` är:

- **`Selected (long)`** – den för tillfället markerade posten
- **`Highlighted (long)`** – den för tillfället visade posten
- **`ItemId (long)`** – postens ID

Kontrollelement i dialogrutor

StarOffice Basic känner igen en serie kontrollelement som kan delas in i följande gruper:

Inmatningsfält:

- Textfält
- Datumfält
- Tidsfält
- Numeriska fält
- Valutafält
- Fält som anpassas till alla format

Knappar:

- Standardknappar
- Kryssrutor
- Alternativfält

Urvalslistor:

- Listrutor
- Kombinationsfält

Andra kontrollelement:

- Rullningslister (horisontella och vertikala)
- Fält för grupper
- Förloppsindikatorer
- Delningslinjer (horisontella och vertikala)
- Grafik
- Filurvalsfält

De viktigaste av de här kontrollelementen beskrivs nedan.

Knappar

En knapp utför en åtgärd när du klickar på den.

I denenklaste formen utlöser knappen en `Vid initiering`-händelse när användaren klickar på den. Du kan även länka en annan åtgärd till knappen, t.ex. att öppna en dialogruta med egenskapen `PushButtonType`. När du klickar på en knapp som har egenskapen satt till värdet 0 påverkas inte dialogrutan. Om du klickar på en knapp som har egenskapen satt till värdet 1 stängs dialogrutan och `Utför`-metoden för dialogrutan returnerar värdet 1 (dialogrutesekvensen har avslutats korrekt). Om `PushButtonType` har värdet 2 stängs dialogrutan och `Utför`-metoden för dialogrutan returnerar värdet 0 (dialogrutan stängs).

Nedanför följer alla egenskaper som är tillgängliga via knappmodellen:

- **`Model.BackgroundColor (long)`** – bakgrundens färg
- **`Model.DefaultButton (Boolean)`** – knappen används som standardvärde och reagerar på Retur-tangenten om fokus saknas.
- **`Model.FontDescriptor (struct)`** – struktursom anger detaljerna för teckensnittet som ska användas (i överensstämmelse med strukturen `com.sun.star.awt.FontDescriptor`)
- **`Model.Label (String)`** – etikett som visas på knappen
- **`Model.Printable (Boolean)`** – kontrollelementet kan skrivas ut
- **`Model.TextColor (Long)`** – kontrollelementets textfärg
- **`Model.HelpText (String)`** – hjälptext som visas när du flyttar muspekaren över kontrollelementet
- **`Model.HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet
- **`PushButtonType (short)`** – åtgärd som är länkad till knappen (0: ingen åtgärd, 1: OK, 2: Avbryt)

Alternativfält

De här fälten används oftast i grupper så att du kan välja bland ett eller flera alternativ. När du markerar ett alternativ inaktiveras alla andra alternativ i gruppen. Detta gör att endast ett alternativfält i taget kan vara markerat.

För kontrollelementet alternativfält finns två egenskaper:

- **`State (Boolean)`** – aktiverar fältet
- **`Label (String)`** – etikett som visas på fältet

Du kan även använda följande egenskaper från modellen för alternativfält:

- **`Model.FontDescriptor (struct)`** – struktur med detaljer om teckensnittet som ska användas (i överensstämmelse med `com.sun.star.awt.FontDescriptor`)
- **`Model.Label (String)`** – etikett som visas på kontrollelementet
- **`Model.Printable (Boolean)`** – kontrollelementet kan skrivas ut
- **`Model.State (Short)`** – om den här egenskapen har värdet 1 är alternativet aktiverat, annars är det inaktiverat
- **`Model.TextColor (Long)`** – kontrollelementets textfärg
- **`Model.HelpText (String)`** – hjälptext som visas när du vilar muspekaren över kontrollelementet
- **`Model.HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet

Om du vill kombinera flera alternativfält i en grupp måste du placera dem efter varandra i en aktiveringssekvens utan mellanrum (egenskapen `Model.TabIndex`, beskrivs som Order i dialogruteredigeraren). Om aktiveringssekvensen avbryts av ett annat kontrollelement startar StarOffice automatiskt med ett nytt kontrollelement som kan aktiveras oberoende av den första gruppen kontrollelement.

> Till skillnad från i VBA kan du inte infoga alternativfält i en grupp av kontrollelement i StarOffice Basic. Grupperingen av kontrollelement i StarOffice Basic används bara som en visuell delning, d.v.s. en ram ritas runt kontrollelementen.

Kryssrutor

Kryssrutor används för inmatning av Ja- eller Nej-värden och beroende på läge kan de anta två eller tre tillstånd. Utöver tillståndet Ja respektive Nej kan en kryssruta ha ett ***mellantillstånd*** om den Ja- eller Nej-status som angetts har mer än en betydelse eller är otydlig.

De här egenskaperna finns för kryssrutor:

- **`State (Short)`** – kryssrutans status (0: nej, 1: ja, 2: mellan)
- **`Label (String)`** – etikett för kontrollelementet
- **`enableTriState (Boolean)`** – utöver aktiverad och inaktiverad kan du även använda mellantillståndet.

Modellobjektet för en kryssruta ger tillgång till följande egenskaper:

- **Model.FontDescriptor (struct)** – strukturmed detaljer om teckensnittet som används (i överensstämmelse med strukturen `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)** – etikett för kontrollelementet
- **Model.Printable (Boolean)** – kontrollelementet kan skrivas ut
- **Model.State (Short)** – kryssrutans status (0: nej, 1: ja, 2: mellan)
- **Model.Tabstop (Boolean)** – kontrollelementet kan nås med Tab-tangenten
- **Model.TextColor (Long)** – kontrollelementets textfärg
- **Model.HelpText (String)** – hjälptext som visas när du vilar muspekaren över kontrollelementet
- **Model.HelpURL (String)** – URL till online-hjälpen för det motsvarande kontrollelementet

Textfält

I textfält kan användare mata in nummer och text. Tjänsten `com.sun.star.awt.UnoControlEdit` utgör basen för textfälten.

Ett textfält kan innehålla en eller flera rader och användarnas inmatningar kan redigeras eller blockeras. Textfält kan också användas som speciella valutafält eller numeriska fält och som skärmbilder för speciella åtgärder. Eftersom de här kontrollelementen baseras på `tjänsten UnoControlEdit` Uno, är den programkontrollerade hanteringen liknande.

De här egenskaperna finns för textfält:

- **Text (String)** – aktuell text
- **SelectedText (String)** – den för tillfället visade texten
- **Selection (Struct)** – skrivskyddad visning av detaljer (struktur i överensstämmelse med `com.sun.star.awt.Selection`, med egenskaperna `Min` och `Max` för att ange start- och slutpunkt för den aktuella visningen)
- **MaxTextLen (short)** – maximalt antal tecken som du kan mata in i fältet
- **Editable (Boolean)** – `True` aktiverar alternativet för textinmatning, `False` blockerar inmatningsalternativ (det går inte att anropa egenskapen direkt utan endast via `IsEditable`)
- **IsEditable (Boolean)** – kontrollelementets innehåll kan ändras, skrivskyddat.

Dessutom ger det associerade modellobjektet tillgång till de här egenskaperna:

- **Model.Align (short)** – textorientering (0: vänsterjusterad, 1: centrerad, 2: högerjusterad)
- **Model.BackgroundColor (long)** – färg på kontrollelementets bakgrund
- **Model.Border (short)** – inramningstyp (0: ingen inramning, 1: 3D-inramning, 2: enkel inramning)
- **Model.EchoChar (String)** – echo-tecken för lösenordsfält

- **Model.FontDescriptor (struct)** – struktur med detaljer om teckensnittet som används (i överensstämmelse med strukturen `com.sun.star.awt.FontDescriptor`)
- **Model.HardLineBreaks (Boolean)** – automatiska radbrytningar infogas permanent i kontrollelementets text
- **Model.HScroll (Boolean)** –
- **Model.MaxTextLen (Short)** – maximal längd för texten, där 0 motsvarar ingen begränsning
- **Model.MultiLine (Boolean)** – tillåter att flera rader med text matas in
- **Model.Printable (Boolean)** – kontrollelementet kan skrivas ut
- **Model.ReadOnly (Boolean)** – kontrollelementets innehåll är skrivskyddat
- **Model.Tabstop (Boolean)** – kontrollelementet kan nås med Tab-tangenten
- **Model.Text (String)** – text associerad med kontrollelementet
- **Model.TextColor (Long)** – kontrollelementets textfärg
- **Model.VScroll (Boolean)** – texten har en vertikal rullningslist
- **Model.HelpText (String)** – hjälptext som visas när du vilar muspekaren över kontrollelementet
- **Model.HelpURL (String)** – URL till online-hjälpen för det motsvarande kontrollelementet

Listrutor

Listrutor (tjänsten `com.sun.star.awt.UnoControlListBox`) stöder de här egenskaperna:

- **ItemCount (Short)** – antal element, skrivskyddad
- **SelectedItem (String)** – text för den visade posten, skrivskyddad
- **SelectedItems (Array Of Strings)** – datafält med visade poster, skrivskyddad
- **SelectedItemPos (Short)** – den för tillfället visade postens nummer, skrivskyddad
- **SelectedItemsPos (Array of Short)** – datafält med de visade posternas nummer (gäller listor som stöder flera val), skrivskyddad
- **MultipleMode (Boolean)** – `True` aktiverar alternativet för val av flera poster, `False` blockerar flera val (det går inte att anropa egenskapen direkt utan endast via `IsMultipleMode`)
- **IsMultipleMode (Boolean)** – tillåter flera val i listor, skrivskyddad

För listrutor finns följande metoder:

- **addItem (Item, Pos)** – anger strängen som angetts i Item i listan vid positionen Pos position
- **addItems (ItemArray, Pos)** – anger posterna som listats i strängens ItemArray-datafält i listan vid positionen Pos position
- **removeItems (Pos, Count)** – tar bort Count-posterna vid positionen Pos position
- **selectItem (Item, SelectMode)** – aktiverar eller inaktiverar visning för de element som angetts i strängen Item beroende på den logiska variabeln SelectMode
- **makeVisible (Pos)** – rullar genom listfältet så att posten som angetts med Pos syns

Modellobjektet för listrutorna ger tillgång till följande egenskaper:

- **Model.BackgroundColor (long)** – kontrollelementets bakgrundsfärg
- **Model.Border (short)** – inramningstyp (0: ingen inramning, 1: 3D-inramning, 2: enkel inramning)
- **Model.FontDescriptor (struct)** – strustruktur med detaljer om teckensnittet som används (i överensstämmelse med strukturen com.sun.star.awt.FontDescriptor)
- **Model.LineCount (Short)** – antal rader i ett kontrollelement
- **Model.MultiSelection (Boolean)** – tillåter att flera poster markeras
- **Model.SelectedItems (Array of Strings)** – lista över markerade poster
- **Model.StringItemList (Array of Strings)** – lista över alla poster
- **Model.Printable (Boolean)** – kontrollelementet kan skrivas ut
- **Model.ReadOnly (Boolean)** – kontrollelementets innehåll är skrivskyddat
- **Model.Tabstop (Boolean)** – kontrollelementet kan nås med Tab-tangenten.
- **Model.TextColor (Long)** – kontrollelementets textfärg
- **Model.HelpText (String)** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet
- **Model.HelpURL (String)** – URL till online-hjälpen för det motsvarande kontrollelementet

VBA-alternativet för utfärdande av listposter med ett numeriskt extravärde (ItemData) finns inte i StarOffice Basic. Om du vill använda numeriska värden (t.ex. databas-ID) utöver den språkliga texten, måste du skapa ett tilläggsdatafält som hanteras parallellt med listrutan.

KAPITEL **12**

Formulär

I mångt och mycket motsvarar strukturen för StarOffice-formulär dialogrutorna som beskrevs i föregående kapitel. Det finns dock några viktiga skillnader:

- Dialogrutorupptäder i form av en enskild dialogruta som visas över dokumentet och som inte tillåter några andra åtgärder än dialogruteprocesser förrän dialogrutesessionen avslutats. Formulär å andra sidan visas direkt i dokumentet precis som teckningselement.
- Dialogrutor skapas i en dialogruteredigerare som finns i StarOffice Basic utvecklingsmiljö. Formulär skapas med **symbollisten Formulärfunktioner** direkt i dokumentet.
- Medan dialogrutefunktionerna finns i alla StarOffice-dokument är alla formulärfunktioner bara tillgängliga i text- och tabelldokument.
- Kontrollelementen i ett formulär kan länkas med en extern databastabell. Den här funktionen är inte tillgänglig för dialogrutor.
- Kontrollelementen för dialogrutor och formulär skiljer sig ur flera aspekter.

Användare som vill använda formulär med egna metoder för händelsehantering hänvisas till kapitel 11 (***Dialogrutor***). De mekanismer som beskrivs där är identiska med de som gäller för formulär.

Arbeta med formulär

StarOffice-formulär kan innehålla textfält, listrutor, alternativfält och många andra kontrollelement som infogas direkt i ett text- eller tabelldokument. Du använder symbollisten **Formulärfunktioner** när du redigerar formulär.

För StarOffice-formulär finns två lägen: utkastläge och visningsläge. I utkastläge kan kontrollelementens placering ändras och egenskaperna kan redigeras i ett egenskapsfönster.

Symbollisten **Formulärfunktioner** använder du även när du växlar mellan lägena.

Bestämma objektsformulär

I StarOffice placeras kontrollelementen för ett formulär på ritobjektsnivå. Det egentliga objektsformuläret kommer du åt via listan `Formulär` på teckningsnivån. I textdokument får du åtkomst till objekten så här:

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Metoden `GetByIndex` returnerar formen med indexnumret 0.

När du arbetar med tabelldokument används ett mellanläge via listan `Tabeller` eftersom teckningsnivåerna inte finns direkt i dokumentet utan i enskilda tabeller:

```
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)
```

Vilket metodnamnet `GetByIndex` redan avslöjat kan ett dokument innehålla flera formulär. Detta är till exempel användbart om innehållet i olika databaser visas i samma dokument eller om en 1:n-databasrelation visas i ett formulär. Alternativet för att skapa underformulär tillhandahålls också för detta ändamål.

Tre aspekter av kontrollelementformulär

Ett kontrollelement i ett formulär har tre aspekter:

- Först har vi kontrollelementets ***Modell***. Detta är nyckelobjektet för en StarOffice Basic-programmerare som arbetar med kontrollelementformulär.
- Motsvarigheten till denna är kontrollelementets ***Vy*** som visningsinformationen hanteras med.
- Eftersom kontrollelementformulär i dokument hanteras som ett speciellt teckningselement finns det även ett ***Figurobjekt*** som återspeglar de teckningselementspecifika egenskaperna för kontrollelementet (särskilt dess position och storlek).

Åtkomst till kontrollelementformulärets modell

Det går att komma åt modellerna för kontrollelementen i ett formulär via `GetByName`-metoden för `formuläret Objekt`:

```
Dim Doc As Object
Dim Form As Object
Dim Ctl As Object

Doc = StarDesktop.CurrentComponent
Form = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Form.getByName("MinListruta")
```

I exempletbestäms modellen för kontrollelementet `MinListruta` som finns i det första formuläret i det öppna textdokumentet.

Om du är osäker på vilket formulär som kontrollelementets finns i kan du söka i alla formulär efter det kontrollelement det gäller:

```
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("MinListruta") Then
            Ctl = Form.GetbyName("MinListruta")
            Exit Function
      End If
Next I
```

I exemplet kontrolleras alla formulär i ett textdokument med metoden `HasByName` för att se om de innehåller en kontrollelementmodell som heter `MinListruta`. Om en motsvarande modell hittas sparas en referens till denna i variabeln `Ctl` och sökningen avslutas.

Åtkomst till kontrollelementformulärets vy

För att komma åt kontrollelementformulärets vy behövs den associerade modellen. Vyn kan sedan bestämmas med hjälp av modellen och dokumentstyrenheten.

```
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("MinListruta") Then
            Ctl = Form.GetbyName("MinListruta")
            CtlView = DocCrl.GetControl(Ctl)
            Exit Function
      End If
Next I
```

Koden i exemplet är mycket lik koden som visades i det föregående exemplet där kontrollelementets modell bestämdes. Det är inte bara dokumentobjektet `Doc` som används utan även dokumentstyrenhetsobjektet `DocCrl` som refererar till det aktuella dokumentfönstret. Med hjälp av detta styrenhetsobjekt och modellen för kontrollelementet används sedan metoden `GetControl` för att bestämma vyn (variabeln `CtlView`) för kontrollelementsformuläret.

Åtkomst till kontrollelementformulärets figurobjekt

Den metod som används för att få åtkomst till kontrollelementets figurobjekt använder också dokumentets teckningsnivå. För att bestämma ett visst kontrollelement måste alla teckningselement på teckningsnivån sökas igenom.

```
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 = "MinListruta" Then
                        Exit Function
            End If
      End If
Next
```

I exemplet kontrolleras alla teckningselement för att bestämma om de stöder gränssnittet `com.sun.star.drawing.XControlShape` som krävs för kontrollelementformulär. Om så är fallet kontrolleras det med egenskapen `Control.Name` om namnet på kontrollelementet är `MinListruta`. Om det är sant avslutar funktionen sökningen.

Bestämma storlek och position för kontrollelement

Storlek och position för kontrollelment bestäms som tidigare nämnts genom att använda det associerade `figurobjektet`. Kontrollelementets figur tillhandahåller, liksom för andra `figurobjekt`, egenskaperna `Storlek` och `Position` för det här ändamålet:

- **`Size (struct)`** – kontrollelementets storlek (`datastrukturen com.sun.star.awt.Size`).
- **`Position (struct)`** – kontrollelementets position (`datastrukturen com.sun.star.awt.Point`).

I följande exempel visas hur position och storlek för ett kontrollelement kan anges med det associerad`figurobjektet`:

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
```

```
Shape.Position = Point
```

`Figurobjektet` för kontrollelementet måste vara känt för att koden ska fungera. Om så inte är fallet måste det bestämmas med ovanstående kod.

Kontrollelementformulär i detalj

De kontrollelement som är tillgängliga för formulär liknar de för dialogrutor. Urvalet varierar från enkla textfält via list- och kombinationsrutor vidare till olika knappar.

Nedanför finns en lista över det viktigaste egenskaperna för kontrollelementformulär. Alla egenskaper utgör en del av de associerade modellobjekten.

Utöver standardkontrollelementen finns det även ett tabellkontrollelement för formulär med vilket hela databastabeller kan integreras. Detta beskrivs i avsnittet ***Databasformulär*** i kapitel 11.

Knappar

Modellobjektet för en formulärknapp ger följande egenskaper:

- **`BackgroundColor (long)`** – bakgrundsfärg.
- **`DefaultButton (Boolean)`** – knappen fungerar som ett standardvärde. I det här fallet reagerar den också på inmatningsknappen om fokus saknas.
- **`Enabled (Boolean)`** – kontrollelementet kan aktiveras.
- **`Tabstop (Boolean)`** – kontrollelementet kan nås via Tab-tangenten.
- **`TabIndex (Long)`** – kontrollelementets positition i aktiveringssekvensen.
- **`FontName (String)`** – teckensnittstypens namn.
- **`FontHeight (Single)`** – tecknets höjd i punkter (pt).
- **`Tag (String)`** – sträng som innehåller ytterligare information som kan sparas i knappen för programkontrollerad åtkomst.
- **`TargetURL (String)`** – mål-URL för knappar av URL-typen.
- **`TargetFrame (String)`** – namnet på fönstret (eller ramen) som `TargetURL` öppnas i när knappen aktiveras (gäller knappar av `URL` -typen).
- **`Label (String)`** – knappetikett.
- **`TextColor (Long)`** – kontrollelementets textfärg.
- **`HelpText (String)`** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet.
- **`HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet.
- **`ButtonType (Enum)`** – åtgärd som är länkad med knappen (standardvärde från `com.sun.star.form.FormButtonType`).

Via egenskapen `ButtonType` kan du definiera en åtgärd som utförs automatiskt när knappen trycks ned. Den associerade konstantgruppen `com.sun.star.form.FormButtonType` ger följande värden:

- **`PUSH`** – standardknapp.
- **`SUBMIT`** – slut på formulärinmatning (gäller framför allt HTML-formulär).
- **`RESET`** – återställer alla värden i formuläret till de ursprungliga värdena.
- **`URL`** – anrop av den URL som definierats i `TargetURL` (öppnas i det fönster som angetts via TargetFrame).

Knapptyperna **OK** och **Avbryt** som finns för dialogrutor stöds inte i formulär.

Alternativfält

Följande egenskaper för alternativfält är tillgängliga via deras modellobjekt:

- **`Enabled (Boolean)`** – kontrollelementet kan aktiveras.
- **`Tabstop (Boolean)`** – kontrollelementet kan nås med Tab-tangenten.
- **`TabIndex (Long)`** – kontrollelementets position i aktiveringssekvensen.
- **`FontName (String)`** – teckensnittstypens namn.
- **`FontHeight (Single)`** – tecknets höjd i punkter (pt).
- **`Tag (String)`** – sträng som innehåller ytterligare information som kan sparas i knappen för programkontrollerad åtkomst.
- **`Label (String)`** – skrift på en knapp.
- **`Printable (Boolean)`** – kontrollelementet kan skrivas ut.
- **`State (Short)`** – om värdet är 1 är alternativet aktiverat annars är det inaktiverat.
- **`RefValue (String)`** – sträng som används om ytterligare information ska sparas (t.ex. ID-nummer för dataposter).
- **`TextColor (Long)`** – kontrollelementets textfärg.
- **`HelpText (String)`** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet.
- **`HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet.

Mekanismen för gruppering av alternativfält skiljer mellan kontrollelement för dialogrutor och formulär. Medan kontrollelement i dialogrutor som förekommer efter varandra automatiskt kombineras till en grupp, utförs grupperingen i formulär på grundval av namn. Alla alternativfält i en grupp måste ha samma namn. I StarOffice kombineras de grupperade kontrollelementen i en matris så att de enskilda knapparna i ett StarOffice Basic-program kan nås på samma sätt som tidigare.

I följande exempel visas hur modellen för en kontrollelementgrupp kan bestämmas:

```
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("MinaAlternativ") Then
            Ctl = Form. GetGroupbyName("MinaAlternativ")
            Exit Function
      End If
Next I
```

Koden motsvarar den i tidigare exempel där en enkel kontrollelementmodell bestämdes. Alla formulär i det aktuella textdokumentet söks igenom i en loop och metoden `HasByName` används för att kontrollera om det motsvarande formuläret innehåller ett element med det eftersökta namnet `MinaAlternativ`. Om så är fallet sker åtkomsten till modellmatrisen med metoden `GetGroupByName` (i stället för metoden `GetByName` som används för enkla modeller).

Kryssrutor

Med modellobjektet för ett kryssruteformulärkan de här egenskaperna användas:

- **`Enabled (Boolean)`** – kontrollelementet kan aktiveras.
- **`Tabstop (Boolean)`** – kontrollelementet kan nås med Tab-tangenten.
- **`TabIndex (Long)`** – kontrollelementets position i aktiveringssekvensen.
- **`FontName (String)`** – teckensnittstypens namn.
- **`FontHeight (Single)`** – tecknets höjd i punkter (pt).
- **`Tag (String)`** – sträng som innehåller ytterligare information som kan sparas i knappen för programkontrollerad åtkomst.
- **`Label (String)`** – knappetikett.
- **`Printable (Boolean)`** – kontrollelementet kan skrivas ut.
- **`State (Short)`** – om värdet är 1 är alternativet aktiverat annars är det inaktiverat.
- **`RefValue (String)`** – sträng som används om ytterligare information ska sparas (t.ex. ID-nummer för dataposter).
- **`TextColor (Long)`** – kontrollelementets textfärg.
- **`HelpText (String)`** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet.

- **`HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet.

Textfält

För modellobjekten för textfältformulär finns följande egenskaper:

- **`Align (short)`** – textorientering (0: vänsterjusterad, 1: centrerad, 2: högerjusterad).
- **`BackgroundColor (long)`** – kontrollelementets bakgrundsfärg.
- **`Border (short)`** – inramningstyp (0: ingen inramning, 1: 3D-inramning, 2: enkel inramning).
- **`EchoChar (String)`** – echo-tecken för lösenordsfält.
- **`FontName (String)`** – teckensnittstypens namn.
- **`FontHeight (Single)`** – tecknets höjd i punkter (pt).
- **`HardLineBreaks (Boolean)`** – de automatiska radbrytningarna infogas permanent i kontrollelementets text.
- **`HScroll (Boolean)`** – texten har en horisontell rullningslist.
- **`MaxTextLen (Short)`** – maximal längd för texten, där 0 motsvarar ingen begränsning.
- **`MultiLine (Boolean)`** – inmatningar över flera rader tillåts.
- **`Printable (Boolean)`** – kontrollelementet kan skrivas ut.
- **`ReadOnly (Boolean)`** – kontrollelementets innehåll är skrivskyddat.
- **`Enabled (Boolean)`** – kontrollelementet kan aktiveras.
- **`Tabstop (Boolean)`** – kontrollelementet kan nås med Tab-tangenten.
- **`TabIndex (Long)`** – kontrollelementets position i aktiveringssekvensen.
- **`FontName (String)`** – teckensnittstypens namn.
- **`FontHeight (Single)`** – tecknets höjd i punkter (pt).
- **`Text (String)`** – kontrollelementets text.
- **`TextColor (Long)`** – kontrollelementets textfärg.
- **`VScroll (Boolean)`** – texten har en vertikal rullningslist.
- **`HelpText (String)`** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet.
- **`HelpURL (String)`** – URL till online-hjälpen för det motsvarande kontrollelementet.

Listrutor

Modellobjektet för listruteformulär ger tillgång till följande egenskaper:

- **BackgroundColor (long)** – kontrollelementets bakgrundsfärg.
- **Border (short)** – inramningstyp (0: ingen inramning, 1: 3D-ram, 2: enkel ram).
- **FontDescriptor (struct)** – struktur med detaljer om teckensnittet som används (i överensstämmelse med strukturen `com.sun.star.awt.FontDescriptor`).
- **LineCount (Short)** – antal rader i ett kontrollelement.
- **MultiSelection (Boolean)** – tillåter att flera poster markeras.
- **SelectedItems (Array of Strings)** – lista över markerade poster.
- **StringItemList (Array of Strings)** – lista över alla poster.
- **ValueItemList (Array of Variant)** – lista som innehåller ytterligare information för varje post (t.ex. ID-nummer för dataposter).
- **Printable (Boolean)** – kontrollelementet kan skrivas ut.
- **ReadOnly (Boolean)** – kontrollelementets innehåll är skrivskyddat.
- **Enabled (Boolean)** – kontrollelementet kan aktiveras.
- **Tabstop (Boolean)** – kontrollelementet kan nås med Tab-tangenten.
- **TabIndex (Long)** – kontrollelementets position i aktiveringssekvensen.
- **FontName (String)** – teckensnittstypens namn.
- **FontHeight (Single)** – tecknets höjd i punkter (pt).
- **Tag (String)** – sträng som innehåller ytterligare information som kan sparas i knappen för programkontrollerad åtkomst.
- **TextColor (Long)** – kontrollelementets textfärg.
- **HelpText (String)** – visar automatiskt hjälptexten som visas om muspekaren befinner sig ovanför kontrollelementet.
- **HelpURL (String)** – URL till online-hjälpen för det motsvarande kontrollelementet.

> Via egenskapen `ValueItemList` ger listruteformulären en motsvarighet till VBA-egenskapen `ItemData` som gör att du kan administrera ytterligare information för enskilda listposter.

Dessutom är följande metoder tillgängliga via listrutans visningsobjekt.

- **addItem (Item, Pos)** – infogar strängen som angetts i `Item` vid positionen `Pos` i listan.
- **addItems (ItemArray, Pos)** – infogar posterna som listats i strängens `ItemArray` -datafält i listan vid positionen `Pos` position
- **removeItems (Pos, Count)** – tar bort `Count`-posterna vid positionen `Pos`.
- **selectItem (Item, SelectMode)** – aktiverar eller inaktiverar visning för de element som angetts i strängen `Item` beroende på variabeln `SelectMode`.
- **makeVisible (Pos)** – rullar genom listfältet så att posten som angetts med `Pos` syns.

Databasformulär

StarOffice-formulär kan länkas direkt till en databas. De formulär som skapas på det här sättet ger tillgång till alla funktioner som finns i ett databasgränssnitt utan att något oberoende programmeringsarbete krävs.

Användaren kan bläddra och söka i utvalda tabeller och sökningar, ändra dataposter och infoga nya. Alla relevanta data hämtas automatiskt från databasen och alla ändringar som görs skrivs tillbaka till databasen.

Ett databasformulär motsvarar i stort sett ett StarOffice-standardformulär. Utöver standardegenskaperna måste följande databasspecifika egenskaper anges i formuläret:

- **`DataSourceName (String)`** – namn på datakällan (se kapitel 10, ***Databasåtkomst***; Databasåtkomst, datakällan måste vara globalt skapad i StarOffice).
- **`Command (String)`** – namnet på tabellen, sökningen eller det SQL-SELECT-kommando som en länk ska göras till.
- **`CommandType (Const)`** – anger om `Kommando` är en tabell, en sökning eller ett SQL-komando (värde från beräkningen `com.sun.star.sdb.CommandType`).

Beräkningen `com.sun.star.sdb.CommandType` täcker följande värden:

- **`TABLE`** – Tabell
- **`QUERY`** – Sökning
- **`COMMAND`** – SQL-kommando

Databasfälten kopplas till enskilda kontrollelement via den här egenskapen:

- **`DataField (String)`** – namn på det länkade databasfältet.

Tabeller

Det finns ett annat kontrollelement som används för arbete med databaser: tabellkontrollelementet. Det representerar innehållet i en hel databastabell eller en hel sökning. Det enklaste sättet är att länka ett tabellkontrollelement till en databas genom att använda AutoPilot-formuläret, som länkar alla kolumner med de relevanta databasfälten i enlighet med användarens specifikationer. Eftersom den associerade API:n är ganska komplicerad finns det för tillfället ingen anledning att ge en fullständig beskrivning av den.

Bilaga

Konverteringstips för VBA

Konverteringstips för StarOffice 5.x

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

T

U

V

W

X

Y