Guida alla programmazione in BASIC in StarOffice 8

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

N. di parte: 819–1329–05
2005

Adobe PostScript

050323@11223

Indice

CAPITOLO 1

Introduzione

Questo manuale contiene un'introduzione alla programmazione con StarOffice™ 8 Basic e indica le possibili applicazioni consentite dall'uso di StarOffice Basic in StarOffice. Per trarre il massimo dalla consultazione di questo manuale, è consigliabile avere dimestichezza con gli altri linguaggi di programmazione.

I numerosi esempi forniti rendono più veloce lo sviluppo di propri programmi StarOffice Basic.

Organizzazione del manuale

I primi tre capitoli costituiscono un'introduzione a StarOffice Basic:

- Capitolo 2, Il linguaggio StarOffice Basic
- Capitolo 3, La libreria runtime di StarOffice Basic
- Capitolo 4, Introduzione alla API StarOffice

Questi capitoli forniscono una panoramica di StarOffice Basic e se ne consiglia la lettura a chiunque intenda scrivere programmi in StarOffice Basic.

I restanti capitoli descrivono i singoli componenti della API StarOffice in maggiore dettaglio e possono essere consultati a seconda delle singole esigenze:

- Capitolo 5, Lavorare con i documenti di StarOffice
- Capitolo 6, Documenti di testo
- Capitolo 7, Fogli elettronici
- Capitolo 8, Disegni e presentazioni
- Capitolo 9, Diagrammi
- Capitolo 10, Accesso ai database
- Capitolo 11, Finestre di dialogo
- Capitolo 12, Formulari

Informazioni su StarOffice Basic

Il linguaggio di programmazione StarOffice Basic è stato sviluppato specificamente per StarOffice ed è strettamente integrato nel pacchetto Office.

Come suggerisce il nome, StarOffice Basic è un linguaggio di programmazione della famiglia Basic. Chiunque abbia una precedente esperienza di programmazione con altri linguaggi Basic – in particolare con Visual Basic o Visual Basic for Applications (VBA) di Microsoft – apprenderà rapidamente l'uso di StarOffice Basic. Ampie sezioni dei costrutti di base di StarOffice Basic sono infatti compatibili con Visual Basic.

Il linguaggio di programmazione StarOffice Basic può essere suddiviso in quattro componenti:

- **Il linguaggio StarOffice Basic**: definisce i costrutti elementari del linguaggio, ad esempio, per le dichiarazioni delle variabili, operazioni cicliche e funzioni.
- **La libreria runtime**: fornisce le funzioni standard che non hanno riferimento diretto con StarOffice, ad esempio, le funzioni per la modifica di numeri, stringhe, valori di data e file.
- **La API (Application Programming Interface) di StarOffice**: consente di accedere ai documenti di StarOffice e di crearli, salvarli, modificarli e stamparli.
- **Il Dialog Editor**: crea finestre di dialogo personali e permette di aggiungere elementi di controllo e gestori di eventi.

Nota – La compatibilità tra StarOffice Basic e VBA riguarda sia il linguaggio StarOffice Basic che la libreria runtime. La API StarOffice e il Dialog Editor *non* sono invece compatibili con VBA (la standardizzazione di queste interfacce avrebbe reso impossibili molti dei concetti forniti in StarOffice).

Utenti di StarOffice Basic

Il campo di applicazione di StarOffice Basic inizia dove terminano le funzioni standard di StarOffice. Con StarOffice Basic è possibile automatizzare operazioni di routine, creare collegamenti con altri programmi – ad esempio a un server di database – ed eseguire attività complesse mediante la semplice pressione di un pulsante, il tutto grazie all'uso di script predefiniti.

StarOffice Basic offre un accesso completo a tutte le funzioni di StarOffice, il supporto di tutte le funzionalità, permette di modificare i tipi di documenti e fornisce opzioni per creare finestre di dialogo personali.

Uso di StarOffice Basic

StarOffice Basic può essere utilizzato da qualsiasi utente di StarOffice e non richiede l'impiego di programmi o ausili addizionali. Anche nella sua installazione standard, StarOffice Basic dispone di tutti i componenti necessari per creare macro personalizzate in Basic, tra cui:

- **L'ambiente di sviluppo integrato** (IDE, Integrated Development Environment) che fornisce un editor per la creazione e la prova delle macro.
- **L'interprete** necessario per eseguire le macro di StarOffice Basic.
- **Le interfacce** per le varie applicazioni di StarOffice che garantiscono l'accesso diretto ai documenti di Office.

Ulteriori informazioni

I componenti della API StarOffice presentati nel manuale sono stati selezionati in base ai loro vantaggi pratici per i programmatori in StarOffice Basic. In generale, sono trattate solo parti delle interfacce. Per un quadro in maggiore dettaglio, consultare i riferimenti sulla API disponibili su Internet all'indirizzo:

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

Il documento Developer's Guide descrive la API StarOffice in maggiore dettaglio, ma è destinato principalmente ai programmatori in Java e C++. Chiunque abbia già sufficiente dimestichezza con la programmazione in StarOffice Basic potrà reperire informazioni nella Developer's Guide on StarOffice Basic and StarOffice programming. Il documento può essere scaricato via Internet all'indirizzo:

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

Ai programmatori che desiderino lavorare direttamente in Java o C++ invece che in StarOffice Basic si consiglia la consultazione della StarOffice Developer's Guide e non del presente manuale. La programmazione di StarOffice con Java o C++ costituisce un processo considerevolmente più complesso della programmazione con StarOffice Basic.

CAPITOLO 2

Il linguaggio StarOffice Basic

StarOffice Basic appartiene alla famiglia dei linguaggi Basic. Molte parti di StarOffice Basic sono identiche a Microsoft Visual Basic for Applications e Microsoft Visual Basic. Tutti coloro che abbiano già utilizzato questi linguaggi acquisiranno rapidamente familiarità con StarOffice Basic.

Anche i programmatori specializzati in altri linguaggi – quali Java, C++, o Delphi – apprenderanno velocemente come utilizzare al meglio StarOffice Basic. StarOffice Basic è un linguaggio di programmazione procedurale pienamente sviluppato e non fa più uso di strutture di comando rudimentali, come `GoTo` e `GoSub`.

Permette inoltre di usufruire dei vantaggi della programmazione orientata agli oggetti, dato che un'interfaccia in StarOffice Basic ammette l'uso di librerie di oggetti esterni. Tutta la API StarOffice si basa su queste interfacce, descritte in maggior dettaglio nei capitoli successivi.

Questo capitolo offre una presentazione generale degli elementi chiave e dei costrutti del linguaggio StarOffice Basic nonché della struttura in cui applicazioni e librerie sono orientate a StarOffice Basic.

Presentazione generale dei programmi in StarOffice Basic

StarOffice Basic è un linguaggio interprete. A differenza di C++ o Turbo Pascal, cioè, il compilatore di StarOffice non crea file eseguibili o auto-estraenti, passibili di esecuzione automatica. Al contrario, per eseguire un programma in StarOffice Basic dovrete premere un pulsante: a questo punto il codice viene prima controllato per verificare la presenza di errori e quindi eseguito riga per riga.

Righe di programma

L'esecuzione "orientata alla riga" dell'interprete Basic produce una delle differenze chiave tra Basic e gli altri linguaggi di programmazione. Mentre la posizione delle interruzioni di riga forzate nel codice sorgente dei programmi in Java, C++ o Delphi è irrilevante, ogni riga di un programma in Basic forma un'unità conclusa in se stessa. Le chiamate di funzione, le espressioni matematiche ed altri elementi come le intestazioni di funzioni e operazioni cicliche devono essere completate sulla stessa riga in cui hanno avuto inizio.

Qualora non vi si abbastanza spazio o se ciò producesse righe molto lunghe, è possibile collegare tra loro diverse righe mediante l'aggiunta di trattini di sottolineatura (_). L'esempio seguente mostra come collegare quattro righe di un'espressione matematica:

```
EspressioneLunga = (Espressione1 * Espressione2) + _
   (Espressione3 * Espressione4) + _
   (Espressione5 * Espressione6) + _
   (Espressione7 * Espressione8)
```

Nota – Il trattino di sottolineatura deve essere sempre l'ultimo carattere della riga collegata e non può essere seguito da uno spazio o da una tabulazione, altrimenti il codice genera un errore.

Oltre a collegare le singole righe, StarOffice Basic permette di utilizzare i due punti (:) per dividere una riga in più sezioni in modo che vi sia spazio sufficiente per diverse espressioni. Le assegnazioni

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

possono essere scritte come segue:

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

Commenti

Oltre al codice eseguibile, un programma StarOffice Basic può contenere anche commenti che spieghino le singole parti del programma e forniscano informazioni importanti e utili per le fasi successive.

StarOffice Basic prevede due metodi di inserimento dei commenti nel codice del programma:

- Tutti i caratteri che seguono un apostrofo vengono trattati come commenti:

```
Dim A    ' Questo è un commento per la variabile A
```

- La parola chiave `Rem`, seguita dal commento:

```
Rem Questo commento è introdotto dalla parola chiave Rem.
```

Un commento generalmente include tutti i caratteri fino al termine della riga. StarOffice Basic interpreta quindi la riga seguente come se si trattasse di nuovo di una normale istruzione. Se i commenti coprono diverse righe, ogni riga deve essere identificata come un commento:

```
Dim B     ' Questo commento per la variabile B è relativamente lungo
          ' e si estende su diverse righe. Il
          ' carattere di commento deve pertanto essere ripetuto
          ' in ciascuna riga.
```

Marcatori

Un programma StarOffice Basic può contenere decine, centinaia e persino migliaia di *marcatori*, che possono consistere in nomi di variabili, costanti, funzioni e così via. Per la scelta dei nomi dei marcatori occorre osservare le seguenti regole:

- I marcatori possono contenere solo caratteri latini, numeri e trattini di sottolineatura (_).
- Il primo carattere di un marcatore deve essere una lettera o un trattino di sottolineatura.
- I marcatori non possono contenere caratteri speciali come ä â î ß.
- La lunghezza massima di un marcatore è 255 caratteri.
- Non viene operata alcuna distinzione tra caratteri maiuscoli e minuscoli. Il marcatore `UnaVariabileDiProva`, per esempio, definisce la stessa variabile di `unavariabilediProva` e `UNAVARIABILEDIPROVA`.

  Esiste tuttavia un'eccezione a questa regola: si effettua la distinzione tra maiuscole e minuscole per le costanti UNO-API. Per maggiori informazioni su UNO, vedere il Capitolo 4.

Nota – Le regole di costruzione dei marcatori sono diverse in StarOffice Basic rispetto a VBA. Ad esempio, StarOffice Basic permette di utilizzare i caratteri speciali nei marcatori solo se si utilizza Option Compatible, poiché questi caratteri possono causare problemi in progetti internazionali.

Di seguito sono riportati alcuni esempi di marcatori corretti ed errati:

```
Cognome      ' Corretto
Cognome5     ' Corretto (il numero 5 non è la prima cifra)
Nome Cognome ' Errato (gli spazi non sono consentiti)
DéjàVu       ' Errato (le lettere quali é, à non sono consentite)
5Cognomi     ' Errato (il primo carattere non deve essere un numero)
```

```
Nome,Cognome ' Errato (virgole e punti non sono consentiti)
```

Utilizzo delle variabili

Dichiarazione implicita delle variabili

I linguaggi Basic sono progettati per la massima facilità d'uso. Di conseguenza, StarOffice Basic permette di creare una variabile tramite il semplice uso e senza una dichiarazione esplicita. In altre parole, una variabile esiste a partire dal momento in cui la includete nel codice. A seconda delle variabili già presenti, l'esempio seguente dichiara fino a tre variabili:

```
a = b + c
```

La dichiarazione implicita delle variabili non è una buona prassi di programmazione poiché può causare l'introduzione involontaria di una nuova variabile, ad esempio attraverso un errore di digitazione. Invece di produrre un messaggio di errore, l'interprete inizializza l'errore di digitazione sotto forma di una nuova variabile con valore 0. Pertanto, può risultare arduo individuare gli errori di questo tipo all'interno del codice

Dichiarazione esplicita delle variabili

Al fine di evitare gli errori causati da una dichiarazione implicita delle variabili, StarOffice Basic fornisce uno switch denominato:

```
Option Explicit
```

che deve essere elencato nella prima riga di programma di ciascun modulo e garantisce che venga emesso un messaggio di errore se una delle variabili utilizzate non è dichiarata. Lo switch `Option Explicit` dovrebbe essere incluso in tutti i moduli in Basic.

Nella sua forma più semplice, il comando per la dichiarazione esplicita di una variabile è il seguente:

```
    Dim MyVar
```

Questo esempio dichiara una variabile denominata `Var` e il tipo variante. La variante è una variabile universale che può registrare tutti i valori concepibili, comprese stringhe, numeri interi, numerici decimali e valori logici. Di seguito vengono riportati alcuni esempi di variabile di tipo variante:

```
Var = "Hello World"        ' Assegnazione di una stringa
Var = 1                    ' Assegnazione di un numero intero
Var = 1.0                  ' Assegnazione di un numero decimale
Var = True                 ' Assegnazione di un valore logico
```

Le variabili dichiarate nell'esempio precedente possono essere utilizzate anche per tipi di variabili diverse nello stesso programma. Sebbene ciò garantisca una considerevole flessibilità, è meglio limitare la variabile a un unico tipo di variabile. Quando StarOffice Basic trova un tipo di variabile non definito correttamente in un particolare contesto, viene generato un messaggio di errore.

Per eseguire una dichiarazione di variabile vincolata al tipo, utilizzate lo stile seguente:

```
Dim Var As Integer   ' Dichiarazione di una variabile intera
```

La variabile è dichiarata come intera e può registrare valori numerici interi. Per dichiarare una variabile intera potete avvalervi anche dello stile seguente:

```
Dim Var%          ' Dichiarazione di una variabile intera
```

L'istruzione Dim può registrare diverse dichiarazioni di variabili:

```
Dim Var1, Var2
```

Per assegnare in modo permanente le variabili a un tipo, dovete eseguire assegnazioni separate per ciascun valore:

```
Dim Var1 As Integer, Var2 As Integer
```

Se non si dichiara il tipo per una variabile, StarOffice Basic assegna la variabile di tipo variante. Ad esempio, nella dichiarazione seguente, `Var1` diventa una variante e `Var2` un numero intero:

```
Dim Var1, Var2 As Integer
```

Le sezioni seguenti presentano i tipi di variabili disponibili in StarOffice Basic e come utilizzarli e dichiararli.

Stringhe

Le stringhe, assieme ai valori numerici, formano i tipi di base più importanti di StarOffice Basic. Una stringa è formata da una sequenza di singoli caratteri consecutivi. Il computer salva le stringhe internamente sotto forma di sequenze a numeriche in cui ciascun numero rappresenta un carattere specifico.

Da un set di caratteri ASCII a Unicode

I set di caratteri abbinano i caratteri di una stringa al codice corrispondente (numeri e caratteri) in una tabella che descrive come il computer dovrà visualizzare la stringa.

Il set di caratteri ASCII

Il set di caratteri ASCII è un set di codici che rappresenta numeri, caratteri e simboli speciali con un byte. I codici ASCII da 0 a 127 corrispondono all'alfabeto e ai simboli più comuni (punti, parentesi e virgole), nonché alcuni speciali codici di controllo di schermo e stampante. Il set di caratteri ASCII è generalmente utilizzato come formato standard per il trasferimento dei dati di testo tra computer.

Questo set di caratteri non include tuttavia una serie di caratteri speciali utilizzati in Europa, quali â, ä e î, nonché caratteri di altri formati come ad esempio l'alfabeto cirillico.

Il set di caratteri ANSI

Microsoft ha basato Windows sul set di caratteri ANSI (American National Standards Institute), che si è gradualmente ampliato fino ad includere i caratteri mancanti dal set ASCII.

Tabelle codici

Il set di caratteri ISO 8859 rappresenta uno standard internazionale. I primi 128 caratteri del set di caratteri ISO corrispondono al set di caratteri ASCII. Lo standard ISO introduce però nuovi set di caratteri (*tabelle codici*) così da poter visualizzare correttamente un numero maggiore di lingue. In questo modo, però, lo stesso valore di carattere può rappresentare caratteri diversi nelle diverse lingue.

Unicode

Unicode utilizza quattro byte per ogni carattere e combina set di caratteri diversi per creare uno standard in grado di rappresentare il più ampio numero di lingue del mondo. La versione 2.0 di Unicode è ora supportata da molti programmi, compresi StarOffice e StarOffice Basic.

Variabili stringa

StarOffice Basic salva le stringhe come variabili a stringa in Unicode. Una variabile a stringa può memorizzare fino a 65535 caratteri. Al suo interno, StarOffice Basic salva il valore Unicode associato per ogni carattere. La memoria operativa necessaria per una variabile a stringa dipende dalla lunghezza della stringa stessa.

Esempio di dichiarazione di una variabile a stringa:

```
Dim Variabile As String
```

Ma potete scrivere la stessa dichiarazione anche come:

```
Dim Variabile$
```

Nota – Quando eseguite il porting delle applicazioni in VBA, accertatevi che venga osservata la lunghezza massima consentita della stringa in StarOffice Basic (pari a 65535 caratteri).

Specifica delle stringhe esplicite

Per assegnare una stringa esplicita a una variabile a stringa, racchiudete la stringa tra virgolette (").

```
Dim Stringa As String
MyString = " Questa è una prova"
```

Per dividere una stringa su due righe, aggiungete un segno più (+) alla fine della prima riga:

```
Dim Stringa As String
Stringa =    "Questa stringa è così lunga che" + _
             "è stata divisa in due righe."
```

Per includere le virgolette (") in una stringa, inseritele due volte nel punto pertinente:

```
Dim Stringa As String
Stringa = "una virgoletta ""."    ' produce una virgoletta "
```

Numeri

StarOffice Basic supporta cinque tipi base per l'elaborazione dei numeri:

- Integer
- Long Integer
- Float
- Double
- Currency

Variabili intere (`Integer`)

Le variabili intere possono memorizzare un numero intero compreso tra -32768 e 32767. Una variabile intera può occupare fino a due byte di memoria. Il simbolo di dichiarazione del tipo è "%". I calcoli che utilizzano le variabili intere sono molto rapidi e particolarmente utili per i contatori utilizzati nelle operazioni cicliche. Se si assegna un numero decimale a una variabile intera, il numero viene arrotondato per eccesso o per difetto al numero intero successivo.

Esempi di dichiarazioni di variabili intere:

```
Dim Variabile As Integer
Dim Variabile%
```

Variabili intere lunghe (`Long`)

Le variabili intere lunghe possono memorizzare valori interi compresi tra -2147483648 e 2147483647. Una variabile intera lunga può occupare fino a quattro byte di memoria. Il simbolo di dichiarazione del tipo è "&". I calcoli che utilizzano le variabili intere lunghe sono molto rapidi e particolarmente utili per i contatori utilizzati nelle operazioni cicliche. Se si assegna un numero decimale a una variabile intera lunga, il numero viene arrotondato per eccesso o per difetto al numero intero successivo.

Esempi di dichiarazioni di variabili intere lunghe:

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

Variabili singole (`Single`)

Le variabili singole possono memorizzare qualunque numero decimale positivo o negativo compreso tra $3{,}402823 \times 10^{38}$ e $1{,}401298 \times 10^{-45}$. Una variabile singola può occupare fino a quattro byte di memoria. Il simbolo di dichiarazione del tipo è "!".

In origine, le variabili singole venivano utilizzate per ridurre i tempi di elaborazione richiesti dalle variabili doppie, più precise. Tuttavia, queste considerazioni sulla velocità non sono oggi più applicabili, riducendo così la necessità di utilizzare le variabili singole.

Esempi di dichiarazioni di variabili singole:

```
Dim Variabile as Single
Dim Variabile!
```

Variabili doppie (`Double`)

Le variabili doppie possono memorizzare qualunque numero decimale positivo o negativo compreso tra 1,79769313486232 x 10^{308} e 4,94065645841247 x 10^{-324}. Una variabile doppia può occupare fino a otto byte di memoria. Le variabili doppie sono adatte per i calcoli precisi. Il simbolo di dichiarazione del tipo è "#".

Esempi di dichiarazioni di variabili doppie:

```
Dim Variabile As Double
Dim Variabile#
```

Variabili di tipo valuta (`Currency`)

Le variabili per valuta differiscono dagli altri tipi di variabili in quanto gestiscono valori. Il punto decimale è fisso e seguito da quattro posizioni decimali. La variabile può contenere fino a 15 numeri prima del punto decimale. Una variabile di tipo valuta può memorizzare qualsiasi valore compreso tra –922337203685477,5808 e +922337203685477,5807 e occupa otto byte di memoria. Il simbolo di dichiarazione del tipo è "@".

Le variabili per valuta sono destinate principalmente ai calcoli aziendali che producono errori di arrotondamento non prevedibili a causa dell'uso di numeri decimali.

Esempi di dichiarazioni di variabili per valuta:

```
Dim Variabile As Currency
Dim Variabile@
```

Specifica di numeri espliciti

I numeri possono essere presentati in modi diversi, ad esempio in formato decimale o in notazione scientifica, o persino con una base diversa dal sistema decimale. Ai caratteri numerici in StarOffice Basic si applicano le seguenti regole:

Numeri interi

Il metodo più semplice è quello di utilizzare gli interi, che vengono elencati nel testo sorgente senza la virgola (o il punto) di separazione delle migliaia:

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

I numeri possono essere preceduti da un segno più (+) o da un segno meno (-) (con o senza spazio tra essi):

```
Dim A As Integer
Dim B As Float

A = +121
B = -243
```

Numeri decimali

Per l'inserimento dei numeri decimali, utilizzate il punto (.) come separatore (punto decimale). Questa regola garantisce che i testi sorgente possano essere trasferiti da un paese all'altro senza conversione.

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

A = 1223.53      ' è arrotondato
B = - 23446.46   ' è arrotondato
C = + 3532.76323
```

Potete utilizzare il segno più (+) e il segno meno (-) come prefissi per i numeri decimali (anche in questo caso, con o senza spazio).

Se a una variabile intera viene assegnato un numero decimale, StarOffice Basic arrotonda il valore per eccesso o per difetto.

Stile di scrittura esponenziale

StarOffice Basic permette di specificare i numeri nello stile di scrittura esponenziale, ad esempio, potete scrivere 1.5e-10 per il numero 1,5 x 10^{-10} (0,00000000015). La lettera "e" può essere minuscola o maiuscola, con o senza il segno più (+) come prefisso.

Di seguito sono riportati alcuni esempi di numeri in formato esponenziale corretti ed errati:

```
Dim A as Double

A = 1.43E2       ' Corretto
A = + 1.43E2     ' Corretto (spazio tra segno più e numero base)
A = - 1.43E2     ' Corretto (spazio tra segno meno e numero base)
A = 1.43E-2      ' Corretto (esponente negativo)

A = 1.43E -2     ' Errato (non sono consentiti spazi nel numero)
A = 1,43E-2      ' Errato (virgola non ammessa come separatore decimale)
A = 1.43E2.2     ' Errato (l'esponente deve essere un numero intero)
```

Si noti che nel primo e nel terzo degli esempi errati non viene generato alcun messaggio di errore anche se le variabili restituiscono valori errati. L'espressione

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

è interpretata come 1.43 meno 2, che corrisponde al valore –0.57. Tuttavia, il valore previsto era 1.43 * 10^2 (corrispondente a 0.0143). Con il valore

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

StarOffice Basic ignora la parte dell'esponente dopo il punto decimale e interpreta l'espressione come

```
A = 1.43E2
```

Valori esadecimali

Nel sistema esadecimale (sistema a base 16), un numero a 2 cifre corrisponde precisamente a un byte. Ciò consente di gestire i valori in modo da riflettere più da vicino l'architettura della macchina. Nel sistema esadecimale, i numeri da 0 a 9 e le lettere da A ad F sono utilizzati come valori numerici. "A" corrisponde al valore decimale 10, mentre la lettera F rappresenta il numero decimale 15. StarOffice Basic permette di utilizzare valori esadecimali interi, purché siano preceduti da "&H".

```
Dim A As Long

A = &HFF   ' Valore esadecimale FF, corrisponde al valore decimale 255
A = &H10   ' Valore esadecimale 10, corrisponde al valore decimale 16
```

Valori in ottali

StarOffice Basic comprende anche il sistema ottale (sistema a base 8) che utilizza i numeri da 0 a 7. È necessario utilizzare numeri interi preceduti da "&O".

```
Dim A As Long

A = &O77   ' Valore ottale 77, corrisponde al valore decimale 63
A = &O10   ' Valore ottale 10, corrisponde al valore decimale 8
```

Vero e falso

Variabili booleane

Le variabili booleane possono memorizzare solo due valori: `True` o `False`. Questi valori sono idonei per le specifiche binarie che possono adottare solo uno degli stati nominati. Un valore logico (o booleano) è salvato internamente come numero intero a

due byte, dove 0 corrisponde a `False` e qualsiasi altro valore a `True`. Per le variabili booleane non esiste un simbolo di dichiarazione del tipo. La dichiarazione può essere eseguita unicamente utilizzando il supplemento *As Boolean*.

Esempio di dichiarazione di una variabile booleana:

```
Dim Variabile As Boolean
```

Data e ora

Variabili data

Le variabili data possono contenere valori di data e ora. Al salvataggio dei valori di data, StarOffice Basic utilizza un formato interno che consente confronti e operazioni matematiche sui valori di data e ora. Per le variabili per data non esiste un simbolo di dichiarazione del tipo. La dichiarazione può essere eseguita unicamente utilizzando il supplemento *As Date*.

Esempio di dichiarazione di una variabile data:

```
Dim Variabile As Date
```

Campi di dati

Oltre alle variabili semplici (*scalari*), StarOffice Basic supporta anche i campi di dati (*matrici* o *array*). Un campo di dati contiene diverse variabili identificate con un indice.

Matrici semplici

Una dichiarazione di matrice è simile a quella di una variabile semplice ma, a differenza di questa, il nome della matrice è seguito da parentesi che contengono le specifiche del numero di elementi. L'espressione

```
Dim Matrice(3)
```

dichiara una matrice con quattro variabili del tipo variante, `Matrice(0)`, `Matrice(1)`,`Matrice(2)` e `Matrice(3)`.

In una matrice potete dichiarare anche variabili di tipo specifico. La riga seguente, ad esempio, dichiara una matrice con quattro variabili intere:

```
Dim Intero(3) As Integer
```

In questi esempi, l'indice per la matrice inizia sempre con valore iniziale standard di zero. In alternativa, per la dichiarazione del campo di dati potete specificare un intervallo di validità con valori iniziali e finali. L'esempio seguente dichiara un campo di dati con sei valori interi e indirizzabile utilizzando gli indici da 5 a 10:

```
Dim Intero(5 To 10)
```

Gli indici non devono necessariamente essere valori positivi. L'esempio seguente mostra una dichiarazione corretta, ma con limiti dei campi di dati negativi:

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

L'esempio dichiara un campo di dati intero con 6 valori che possono essere identificati utilizzando gli indici da -10 a -5.

Per la definizione degli indici dei campi di dati dovete osservare tre vincoli:

- L'indice più piccolo possibile è -32768.
- L'indice più grande possibile è 32767.
- Il numero massimo di elementi (all'interno di una dimensione del campo di dati) è 16368.

Nota – Per gli indici dei campi di dati in VBA si applicano talvolta limiti diversi. Lo stesso si applica anche al numero massimo di elementi possibile per ogni dimensione. I valori validi possono essere reperiti nella documentazione VBA pertinente.

Valore specificato per l'indice iniziale

L'indice iniziale di un campo di dati inizia generalmente con il valore 0. In alternativa, è possibile modificare l'indice iniziale per tutte le dichiarazioni dei campi di dati sul valore 1 utilizzando la chiamata:

```
Option Base 1
```

La chiamata deve essere inclusa nell'intestazione di un modulo se si desidera applicarla a tutte le dichiarazioni di matrice del modulo. Tuttavia, questa chiamata non incide sulle sequenze UNO definite tramite la API StarOffice il cui indice inizia *sempre* con 0. Per maggiore chiarezza, evitare l'uso di Option Base 1.

Il numero di elementi della matrice non è influenzato dall'uso di Option Base 1; viene modificato solo l'indice iniziale. La dichiarazione

```
Option Base 1
' ...
```

```
Dim Intero(3)
```

crea 4 variabili intere che possono essere descritte con le espressioni `Intero(1)`, `Intero(2)`, `Intero(3)` e `Intero(4)`.

Nota – In StarOffice Basic, l'espressione Option Base 1 non incide sul numero di elementi dell'array come in VBA. In StarOffice Basic è invece l'indice iniziale a spostarsi. Mentre la dichiarazione `Intero(3)` in VBA crea tre valori interi con gli indici da 1 a 3, la stessa dichiarazione in StarOffice Basic crea quattro valori interi con gli indici da 1 a 4. Usando Option Compatible, StarOffice Basic si comporta come VBA.

Campi di dati a più dimensioni

Oltre ai campi di dati a una dimensione, StarOffice Basic supporta anche i campi di dati a più dimensioni. Le dimensioni corrispondenti sono separate per mezzo di virgole. L'esempio

```
Dim MatriceInt(5, 5)
```

definisce una matrice intera con due dimensioni, ciascuna con 6 indici (possono essere identificati con gli indici da 0 a 5). L'intera matrice può registrare un totale di 6 x 6 = 36 valori interi.

Sebbene sia possibile definire centinaia di dimensioni nelle matrici di StarOffice Basic, la quantità di memoria disponibile limita il numero di dimensioni possibili.

Modifiche dinamiche nelle dimensioni dei campi di dati

Gli esempi precedenti sono basati sui campi di dati con una dimensione specificata. Potete però definire anche matrici in cui la dimensione dei campi di dati varia dinamicamente. Ad esempio, potete definire una matrice per contenere tutte le parole di un testo che iniziano con la lettera A. Poiché il numero di queste parole è inizialmente sconosciuto, dovete essere in grado di modificare successivamente i limiti del campo. Per procedere in tal senso in StarOffice Basic, utilizzate la chiamata seguente:

```
ReDim  Matrice(10)
```

Nota – A differenza di VBA, in cui potete utilizzare solo matrici dinamiche con l'istruzione `Dim Array()`, StarOffice Basic permette di modificare gli array sia statici che dinamici con l'istruzione `ReDim`.

L'esempio seguente modifica la dimensione della matrice iniziale in modo che possa registrare 11 o 21 valori:

```
Dim Array(4) As Integer         ' Dichiarazione con cinque elementi

' ...

ReDim Array(10) As Integer      ' Aumento a 11 elementi

' ...

ReDim Array(20) As Integer      ' Aumento a 21 elementi
```

Quando ripristinate le dimensioni di un array, potete utilizzare una qualsiasi delle opzioni descritte nelle sezioni precedenti. Questo include la dichiarazione di campi di dati a più dimensioni e la specifica di valori iniziali e finali espliciti. Modificando le dimensioni del campo di dati, tutti i contenuti vanno persi. Per conservare i valori originali, avvaletevi del comando `Preserve`:

```
Dim Array(10) As Integer              ' Definizione delle dimensioni
                                      ' iniziali

' ...

ReDim Preserve Matrice(20)   As Integer      ' Aumento nel
                                             ' campo di dati, mentre
                                             ' si conserva il contenuto
```

Quando usate `Preserve`, accertatevi che il numero di dimensioni e il tipo di variabili rimanga lo stesso.

Nota – A differenza di VBA, in cui l'uso di `Preserve` vi permette di modificare solo il limite superiore dell'ultima dimensione del campo di dati, StarOffice Basic vi permette di modificare anche le altre dimensioni.

Se utilizzate `ReDim` con `Preserve`, dovete utilizzare lo stesso tipo di dati specificato nella dichiarazione del campo di dati originale.

Campo di applicazione e vita utile delle variabili

Una variabile in StarOffice Basic ha una vita utile limitata e un campo di applicazione limitato in cui può essere letta e utilizzata in altri frammenti di programma. Il tempo di conservazione di una variabile e le posizioni da cui è accessibile dipendono dalla posizione specificata e dal tipo.

Variabili locali

Le variabili dichiarate in una funzione o procedura sono denominate variabili locali:

```
Sub Test
   Dim Intero As Integer

   ' ...
End Sub
```

Le variabili locali rimangono valide solo finché la funzione o la procedura sono in esecuzione e quindi sono riportate a zero. Ogni volta che la funzione viene chiamata, i valori generati in precedenza non sono disponibili.

Per conservare i valori precedenti, dovrete definire la variabile come `Static`:

```
Sub Test
   Static Intero As Integer

   ' ...

End Sub
```

Nota – A differenza di VBA, StarOffice Basic assicura che il nome di una variabile locale non sia utilizzato contemporaneamente come variabile globale e privata nell'intestazione del modulo. Quando si trasferisce un'applicazione VBA in StarOffice Basic, occorre modificare tutti i nomi di variabili duplicati.

Variabili pubbliche

Le variabili pubbliche sono definite nella sezione dell'intestazione di un modulo con la parola chiave Dim. Queste variabili sono disponibili per tutti i moduli nella loro libreria:

Modulo A:

```
Dim A As Integer

Sub Test
   Flip
   Flop
End Sub

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

Modulo B:

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

Il valore della variabile `A` non è modificato dalla funzione `Test`, ma è aumentato di uno nella funzione `Flip` e ridotto di uno nella funzione `Flop`. Entrambe le modifiche alla variabile sono globali.

Per dichiarare un variabile pubblica potete utilizzare anche la parola chiave `Public` invece di `Dim`:

```
Public A As Integer
```

Una variabile pubblica è disponibile solo finché la macro associata è in esecuzione, poi la variabile viene ripristinata.

Variabili globali

Dal punto di visto della funzione, le variabili globali sono simili alle variabili pubbliche, eccetto che i loro valori vengono conservati anche dopo l'esecuzione della macro associata. Le variabili globali sono dichiarate nella sezione dell'intestazione di un modulo utilizzando la parola chiave `Global`:

```
Global A As Integer
```

Variabili private

Le variabili `private` sono disponibili solo nel modulo in cui vengono definite. Utilizzate la parola chiave `Private` per definire la variabile:

```
Private Intero As Integer
```

Se diversi moduli contengono una variabile `Private` con lo stesso nome, StarOffice Basic crea una variabile diversa per ogni ricorrenza del nome. Nell'esempio seguente, i due moduli `A` e `B` presentano una variabile `Private` di nome `C`. La funzione `Test` imposta prima la variabile `Private` nel modulo `A` e quindi la variabile `Private` nel modulo `B`.

Modulo A:

```
Private C As Integer

Sub Test
   SetModuleA      ' Imposta la variabile C del modulo A
   SetModuleB      ' Imposta la variabile C del modulo B

   ShowVarA        ' Mostra la variabile C del modulo A (= 10)
   ShowVarB        ' Mostra la variabile C del modulo B (= 20)
End Sub

Sub SetmoduleeA
   A = 10
End Sub

Sub ShowVarA
   MsgBox C        ' Mostra la variabile C del modulo A.
End Sub
```

Modulo B:

```
Private C As Integer

Sub SetModuleB
   A = 20
End Sub

Sub ShowVarB
   MsgBox C        ' Mostra la variabile C del modulo B.
End Sub
```

Costanti

In StarOffice Basic, utilizzate la parola chiave `Const` per dichiarare una costante.

```
Const A = 10
```

Volendo, nella dichiarazione potete anche specificare il tipo di costante:

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

Operatori

StarOffice Basic comprende gli operatori matematici, logici e di confronto più diffusi.

Operatori matematici

Potete applicare gli operatori matematici a tutti i tipi di numeri, mentre l'operatore + può essere utilizzato anche per collegare le stringhe.

+	Aggiunta di numeri e valori di date, collegamento di stringhe
-	Sottrazione di numeri e valori di date
*	Moltiplicazione di numeri
/	Divisione di numeri
\	Divisione di numeri con un risultato numerico intero (arrotondato)
^	Elevazione a potenza dei numeri
MOD	operazione modulo (calcolo del resto di una divisione)

Operatori logici

Gli operatori logici vi permettono di collegare gli elementi in base alle regole dell'algebra booleana. Se gli operatori sono applicati ai valori logici, il collegamento fornisce direttamente il risultato richiesto. Se utilizzato assieme a valori interi e valori interi lunghi, il collegamento è effettuato a livello di bit.

AND	Collegamento And
OR	Collegamento Or
XOR	Collegamento Exclusive or
NOT	Negazione
EQV	Equivalenza (parti `True` o `False`)
IMP	Implicazione (se la prima espressione è vera, deve essere vera anche la seconda)

Operatori di confronto

Gli operatori di confronto si possono applicare a tutti i tipi di variabili elementari (numeri, date, stringhe e valori logici).

=	Uguaglianza di numeri, date e stringhe
<>	Disuguaglianza di numeri, date e stringhe
>	Controllo "Maggiore di" per numeri, date e stringhe
>=	Controllo "Maggiore di o uguale" per numeri, date e stringhe
<	Controllo "Minore di" per numeri, date e stringhe
<=	Controllo "Minore di o uguale" per numeri, date e stringhe

---

Nota – StarOffice Basic non supporta l'operatore di confronto `Like` di VBA.

Istruzioni condizionali

Utilizzate le istruzioni condizionali per limitare l'esecuzione di un blocco di codice finché non è soddisfatta una particolare condizione.

If...Then...Else

L'istruzione condizionale più comune è l'istruzione `If` come illustrato nell'esempio seguente:

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

L'assegnazione `B = 2` si verifica solo quando il valore della variabile `A` è maggiore di tre. Una variazione dell'istruzione `If` è la clausola `If/Else`:

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

In questo esempio, alla variabile `B` è assegnato il valore di 2 quando `A` è maggiore di 3, altrimenti a `B` è assegnato il valore 0.

Per dichiarazioni più complesse, è possibile inserire a cascata l'istruzione `If`, ad esempio:

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

Se il valore della variabile `A` è uguale a zero, a `B` è assegnato il valore 0. Se `A` è minore di 3 (ma non uguale a zero), allora a `B` è assegnato il valore 1. In tutti gli altri casi (ovvero, se `A` è maggiore o uguale a 3), a `B` viene assegnato il valore 2.

Select...Case

L'istruzione `Select...Case` è un'alternativa all'utilizzo di una serie di istruzioni `If` ed è utilizzata quando occorre controllare un valore a fronte di diverse condizioni:

```
Select Case GiornoSettimana

Case 1:
   NomeGiorno = "Domenica"
Case 2:
   NomeGiorno = "Lunedì"
Case 3:
   NomeGiorno = "Martedì"
Case 4:
   NomeGiorno = "Mercoledì"
Case 5:
   NomeGiorno = "Giovedì"
Case 6:
   NomeGiorno = "Venerdì"
Case 7:
   NomeGiorno = "Sabato"
End Select
```

In questo esempio, il nome del giorno della settimana corrisponde a un numero, così che alla variabile `GiornoSettimana` è assegnato il valore di 1 per `Domenica`, 2 per `Lunedì` e così via.

Il comando `Select` non è limitato alle semplici assegnazioni 1:1 – potete specificare anche operatori di confronto o elenchi di espressioni in una clausola `Case`. L'esempio seguente elenca la varianti di sintassi di maggiore importanza:

```
Select Case Var

Case 1 To 5

   ' ... Var è compresa tra i numeri 1 e 5
```

```
Case 6, 7, 8

   ' ... Var è 6, 7 o 8

Case Var > 8 And Var < 11

   ' ... Var è maggiore di 8 e minore di 11

Case Else

   ' ... tutti gli altri casi


End Select
```

Cicli

Un ciclo è un'operazione ciclica che esegue un blocco di codice per il numero di passaggi specificati. Potete impostare cicli con un numero indefinito di passaggi.

For...Next

Il ciclo `For...Next` ha un numero fisso di passaggi. Il contatore di cicli definisce il numero di esecuzioni del ciclo. Nell'esempio seguente,

```
Dim I
For I = 1 To 10
   ' ...  Parte interna del ciclo
Next I
```

la variabile `I` è il contatore di cicli, con un valore iniziale di 1. Il contatore è incrementato di 1 al termine di ogni passaggio. Quando la variabile I è uguale a 10, il ciclo di arresta.

Per incrementare il contatore di cicli di un valore diverso da 1 al termine di ogni passaggio, usate la funzione `Step`:

```
Dim I

For I = 1 To 10 Step 0.5

   ' ... Parte interna del ciclo

Next I
```

Nell'esempio, il contatore è aumentato di 0,5 al termine di ogni passaggio e il ciclo viene pertanto eseguito 19 volte.

Si possono utilizzare anche valori di incremento negativi:

```
Dim I

For I = 10 To 1 Step -1

   ' ... Parte interna del ciclo

Next I
```

In questo esempio, il contatore inizia a 10 ed è ridotto di 1 al termine di ogni passaggio, finché il contatore non è 1.

L'istruzione `Exit For` permette di uscire da un ciclo `For` anticipatamente. Nell'esempio seguente, il ciclo viene terminato durante il quinto passaggio:

```
Dim I

For I = 1 To 10

   If I = 5 Then
      Exit For
   End If

   ' ... Parte interna del ciclo

Next I
```

Nota – La variante di ciclo `For Each...Next` usata in VBA non è supportata in StarOffice Basic.

Do...Loop

Il processo `Do...Loop` non è collegato a un numero fisso di passaggi. `Do...Loop` viene eseguito finché non è soddisfatta una determinata condizione. Sono disponibili quattro varianti del processo `Do...Loop` (negli esempi seguenti, `A > 10` rappresenta qualsiasi condizione):

1. La variante `Do While...Loop`

   ```
   Do While A > 10
      ' ... corpo del ciclo
   Loop
   ```

   controlla se la condizione è ancora soddisfatta prima di ogni passaggio e solo allora esegue il ciclo.

2. La variante `Do Until...Loop`

```
Do Until A > 10
   ' ... corpo del ciclo
Loop
```

   esegue il ciclo finché la condizione *non è più* soddisfatta.

3. La variante `Do...Loop While`

```
Do
   ' ... corpo del ciclo
Loop While A > 10
```

   controlla la condizione solo dopo il primo passaggio del ciclo e termina se questa è *soddisfatta*.

4. La variante `Do...Loop Until`

```
Do
   ' ... corpo del ciclo
Loop Until A > 10
```

   controlla inoltre la sua condizione dopo il primo passaggio, ma intraprende il ciclo finché la condizione *non è più* soddisfatta.

Come nel ciclo `For...Next`, il processo `Do...Loop` fornisce anche un comando di arresto. Il comando `Exit Do` permette infatti di uscire da un ciclo indipendentemente dal punto in cui si trova.

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

   ' ... corpo del ciclo
While A > 10
```

Esempio di programmazione: ordinamento con cicli incorporati

Esistono molti modi di utilizzare i cicli, ad esempio, per eseguire ricerche in elenchi, restituire valori o eseguire operazioni matematiche complesse. L'esempio seguente è un algoritmo che utilizza i cicli per ordinare un elenco per nome.

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

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

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

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

I valori vengono interscambiati come coppie finché non sono ordinati in ordine crescente. Le variabili migrano gradualmente verso la posizione destra. Questo algoritmo è noto anche come *Bubble Sort*.

Procedure e funzioni

Le procedure e le funzioni costituiscono i perni della struttura di un programma e forniscono la struttura per dividere un problema complesso in diverse sub-operazioni.

Procedure

Una *procedura* esegue un'azione senza fornire un valore esplicito. La sintassi è:

```
Sub Test

   ' ... qui va inserito il codice effettivo della procedura

End Sub
```

L'esempio definisce una procedura denominata `Test` che contiene codice accessibile da qualsiasi punto del programma. La chiamata viene effettuata inserendo il nome della procedura nel punto pertinente del programma:

```
Test
```

Funzioni

Una *funzione*, proprio come una procedura, combina un blocco di programmi per eseguirli in un'unica unità logica. Tuttavia, a differenza della procedura, la funzione restituisce un valore.

```
Function Test

   ' ... qui va inserito il codice effettivo della funzione

   Test = 123
End Function
```

Il valore restituito viene attribuito utilizzando una semplice assegnazione. L'assegnazione non va necessariamente collocata al termine della funzione, ma può essere inserita anche in qualsiasi punto della funzione.

La funzione precedente può essere richiamata all'interno di un programma nel modo seguente:

```
Dim A

A = Test
```

Il codice definisce una variabile `A` e vi assegna il risultato della funzione `Test`.

Il valore restituito può essere sovrascritto diverse volte all'interno della funzione. Come nel caso dell'assegnazione classica delle variabili, la funzione di questo esempio restituisce il valore che le era stato assegnato per ultimo.

```
Function Test

   Test = 12

   ' ...

   Test = 123

End Function
```

In questo esempio, il valore restituito dalla funzione è 123.

Se un'assegnazione viene interrotta, la funzione restituisce un valore `zero` (numero 0 per i valori numerici e una stringa vuota per le stringhe).

Il valore restituito da una funzione può essere di qualsiasi tipo. Il tipo è dichiarato nello stesso modo di una dichiarazione di variabile:

```
Function Test As Integer

   ' ... qui va inserito il codice effettivo della funzione

End Function
```

Se la specifica di un valore esplicito viene interrotta, il tipo del valore restituito è assegnato come variante.

Termine anticipato di procedure e funzioni

In StarOffice Basic, potete utilizzare i comandi `Exit Sub` e `Exit Function` per terminare anticipatamente una procedura o una funzione, ad esempio per gestire un errore. Questi comandi interrompono la procedura o la funzione e riportano il programma al punto in cui la procedura o la funzione erano state richiamate.

L'esempio seguente mostra una procedura che termina l'implementazione quando la variabile `ErrorOccured` ha il valore `True`.

```
Sub Test
   Dim ErrorOccured As Boolean

   ' ...

   If ErrorOccured Then
      Exit Sub
   End If

   ' ...

End Sub
```

Passaggio dei parametri

Le funzioni e le procedure possono accettare uno o più parametri. I parametri essenziali devono essere racchiusi tra parentesi dopo il nome della funzione o della procedura. L'esempio

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

definisce una procedura che prevede un valore intero `A` e una stringa `B` come parametri.

In StarOffice Basic, i parametri sono normalmente passati come *riferimento*. Le modifiche apportate alle variabili vengono conservate anche dopo l'uscita dalla procedura o dalla funzione:

```
Sub Test
   Dim A As Integer
   A = 10
   ChangeValue(A)
   ' Il parametro A ha ora il valore 20
End Sub
Sub ChangeValue(TheValue As Integer)
    TheValue = 20
End Sub
```

In questo esempio, il valore `A` definito nella funzione `Test` viene passato come parametro alla funzione `ChangeValue`. Il valore viene quindi modificato in 20 e trasmesso a `TheValue`, e conservato all'uscita dalla funzione.

Potete passare un parametro anche come *valore* se non si desidera che successive modifiche al parametro incidano sul valore trasmesso in origine. Per specificare che un parametro deve essere passato come valore, assicurarsi che la parola chiave `ByVal` preceda la dichiarazione della variabile nell'intestazione della funzione.

Nell'esempio precedente, se sostituiamo la funzione `ChangeValue` con la funzione

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

la variabile superiore A non viene influenzata dalla modifica. Dopo la chiamata della funzione `ChangeValue`, la variabile `A` conserva il valore 10.

Nota – Il metodo per passare i parametri alle procedure e alle funzioni in StarOffice Basic è virtualmente identico a quello di VBA. Come standard, i parametri vengono trasmessi come riferimento. Per passare i parametri come valori, utilizzate la parola chiave `ByVal`. In VBA, potete utilizzare la parola chiave `ByRef` per forzare il passaggio di un parametro come riferimento. StarOffice Basic non supporta questa parola chiave perché costituisce già la procedura standard di StarOffice Basic.

Parametri opzionali

Le funzioni e le procedure possono essere chiamate solo se durante la chiamata vengono passati tutti i parametri necessari.

StarOffice Basic consente di definire i parametri come *opzionali*, ovvero, se i valori corrispondenti non sono inclusi in una chiamata, StarOffice Basic passa un parametro vuoto. Nell'esempio seguente,

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

End Sub
```

il parametro `A` è obbligatorio, mentre il parametro `B` è opzionale.

La funzione `IsMissing` controlla se un parametro è stato passato o meno.

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

   ' Controlla se il parametro B è effettivamente presente
   If Not IsMissing (B) Then
      B_Local = B      ' parametro B presente
   Else
      B_Local = 0      ' parametro B mancante -> valore standard 0
   End If

   ' ... Avvia la funzione effettiva

End Sub
```

L'esempio prima prova se il parametro `B` è stato passato e, se necessario, passa lo stesso parametro alla variabile interna `B_Local`. Se il parametro corrispondente non è presente, viene passato a `B_ Local` un valore standard (in questo caso, il valore 0) invece del parametro passato.

Nota – La parola chiave `ParamArray` presente in VBA non è supportata in StarOffice Basic.

Ricorsività

In StarOffice Basic è ora possibile la ricorsività. Una procedura o una funzione ricorsiva ha la capacità di richiamare se stessa finché non rileva che qualche condizione base è stata soddisfatta. Quando la funzione è richiamata con la condizione base, viene restituito un risultato.

L'esempio seguente utilizza una funzione ricorsiva per calcolare il fattoriale dei numeri `42`, `-42` e `3.14`:

```
Sub Main
    Msgbox CalculateFactorial(  42 )    ' Visualizza 1,40500611775288E+51
    Msgbox CalculateFactorial( -42 )    ' Visualizza "Numero non valido
                                        ' per fattoriale"
    Msgbox CalculateFactorial( 3.14 )   ' Visualizza "Numero non valido
                                        ' per fattoriale"
End Sub
```

```
Function CalculateFactorial( Numero )
    If Numero < 0 Or Numero <> Int( Numero ) Then
        CalculateFactorial = "Numero non valido per fattoriale"
    ElseIf Numero = 0 Then
        CalculateFactorial = 1
    Else
        ' Questa è la chiamata ricorsiva:
        CalculateFactorial = Numero * CalculateFactorial( Numero - 1 )
    Endif
End Function
```

L'esempio restituisce il fattoriale del numero `42` mediante chiamate ricorsive della funzione `CalculateFactorial` finché non raggiunge la condizione base di `0! = 1`.

Nota – I livelli di ricorsività vengono impostati in modo diverso a seconda della piattaforma software. Per Windows, il livello di ricorsività è 5800. Per Solaris e Linux, viene eseguita una valutazione della dimensione dello stack e il livello di ricorsività viene calcolato di conseguenza.

Gestione degli errori

La corretta gestione delle situazioni di errore è una delle attività di programmazione che richiede più tempo. StarOffice Basic fornisce un'ampia gamma di strumenti per semplificare la gestione degli errori.

L'istruzione On Error

L'istruzione `On Error` è fondamentale per la gestione degli errori:

```
Sub Test
   On Error Goto ErrorHandler

   ' ... esegue l'attività durante la quale può verificarsi un errore

Exit Sub

ErrorHandler:

   ' ... codice per la gestione dell'errore

End Sub
```

La riga `On Error Goto ErrorHandler` definisce il modo in cui StarOffice Basic procede in caso di errori. L'istruzione `Goto ErrorHandler` verifica che StarOffice Basic esca dalla riga di programma corrente, quindi esegue il codice `ErrorHandler` .

Il comando Resume

Il comando `Resume Next` fa proseguire il programma dalla riga seguente al punto in cui si è verificato l'errore nel programma, dopo l'esecuzione del codice nel gestore di errori:

```
ErrorHandler:

   ' ... codice per la gestione dell'errore

   Resume Next
```

Usate il comando `Resume Proceed` per specificare il punto in cui far proseguire il programma dopo la gestione dell'errore:

```
ErrorHandler:

   ' ... codice per la gestione dell'errore

   Resume Proceed

Proceed:

   ' ... il programma continua da qui dopo l'errore
```

Per far proseguire un programma senza un messaggio di errore quando si verifica un errore, utilizzate il formato seguente:

```
Sub Test
   On Error Resume Next

   ' ... esegue l'attività durante la quale può verificarsi un errore

End Sub
```

Utilizzate il comando `On Error Resume Next` con cautela, perché ha un effetto globale. Per ulteriori informazioni, consultare i "Suggerimenti per la gestione strutturata degli errori" a pagina 46.

Ricerche riguardanti informazioni sugli errori

Nella gestione degli errori, è utile disporre di una descrizione dell'errore e sapere dove e perché si è verificato:

- La variabile `Err` contiene il numero di errori verificatisi.
- La variabile `Error$` contiene una descrizione dell'errore.
- La variabile `Erl` contiene il numero della riga in cui si è verificato l'errore.

La chiamata

```
MsgBox "Errore " & Err & ": " & Error$ & " (riga : " & Erl & ")"
```

mostra come l'informazione sull'errore può essere visualizzata in una finestra di messaggi.

Nota – Mentre VBA riassume i messaggi di errore in un oggetto statistico denominato `Err`, StarOffice Basic fornisce le variabili `Err`, `Error$` ed `Erl`.

Le informazioni di stato rimangono valide finché il programma non incontra un comando `Resume` o `On Error`, quindi l'informazione viene ripristinata.

Nota – In VBA, il metodo `Err.Clear` dell'oggetto `Err` ripristina lo stato di errore dopo che si è verificato un errore. In StarOffice Basic, questa operazione viene espletata con i comandi `On Error` o `Resume`.

Suggerimenti per la gestione strutturata degli errori

Sia il comando di definizione, `On Error`, che il comando `Resume` sono varianti del costrutto `Goto`.

Per strutturare chiaramente il codice in modo da impedire la generazione di errori quando si utilizza questo costrutto, evitate l'uso di comandi di salto senza monitorarli.

Utilizzate il comando `On Error Resume Next` con cautela, perché chiude tutti i messaggi di errore aperti.

La soluzione migliore è adottare un unico approccio di gestione degli errori all'interno del programma – tenere la gestione degli errori separata dal codice effettivo del programma e non tornare al codice originale dopo che si è verificato l'errore.

Il seguente è un esempio di un errore della procedura di gestione:

```
Sub Example

   ' Definire il gestore di errori all'inizio della funzione
   On Error Goto ErrorHandler

   ' ... Qui va inserito il codice effettivo del programma

   ' Disattiva la gestione degli errori
   On Error Goto 0

   ' Fine della regolare esecuzione del programma
Exit Sub

' Inizio della gestione degli errori
ErrorHandler:

   ' Controlla se l'errore era previsto
   If Err = ExpectedErrorNo Then
      ' ... Errore di processo
   Else
      ' ... Avvertenza per errore inatteso
   End If

   On Error Goto 0            ' Disattiva la gestione degli errori
End Sub
```

Questa procedura inizia con la definizione di un gestore di errori, seguito dal codice effettivo del programma. Alla fine del codice del programma, la gestione degli errori è disattivata dalla chiamata `On Error Goto 0` e l'implementazione della procedura è terminata dal comando `Exit Sub` (da non confondersi con `End Sub`).

L'esempio controlla prima se il numero di errore corrisponde al numero previsto (memorizzato nella costante immaginaria `ExpectedErrorNo`) e quindi gestisce l'errore di conseguenza. Se si verifica un altro errore, il sistema emette un'avvertenza. È importante controllare il numero di errore in modo da poter rilevare gli errori imprevisti.

La chiamata `On Error Goto 0` alla fine del codice ripristina le informazioni di stato dell'errore (il codice di errore nelle variabili di sistema `Err`) in modo che possiate riconoscere con chiarezza un errore verificatosi in data successiva.

CAPITOLO 3

La libreria runtime di StarOffice Basic

Le sezioni seguenti presentano le funzioni principali della libreria runtime.

Funzioni di conversione

In molte situazioni, si verificano circostanze in cui una variabile di un tipo deve essere modificata in una variabile di un altro tipo.

Conversioni di tipo implicite ed esplicite

Il modo più semplice di convertire una variabile da un tipo ad un altro è quello di utilizzare un'assegnazione.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

In questo esempio, la variabile `A` è una stringa, mentre la variabile `B` è un numero intero. StarOffice Basic garantisce che la variabile `B` sia convertita in una stringa durante l'assegnazione alla variabile `A`. Questa conversione è molto più elaborata di quello che sembra: il numero intero `B` rimane nella memoria operativa sotto forma di un numero di due byte di lunghezza. `A`, d'altro canto, è una stringa e il computer salva un valore di uno o due byte di lunghezza per ogni carattere (ogni numero). Pertanto, prima di copiare il contenuto da `B` ad `A`, `B` deve essere convertito nel formato interno di `A`.

A differenza della maggior parte dei linguaggi di programmazione, Basic esegue la conversione del tipo automaticamente. Ciò può tuttavia avere conseguenze fatali. A un esame ravvicinato, la sequenza di codice seguente

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

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

che a prima vista sembra corretta, si può trasformare in una trappola. L'interprete Basic calcola prima il risultato del processo di addizione, quindi lo converte in una stringa che, come risultato, produce la stringa 2.

Se, d'altro canto, l'interprete Basic convertisse per prima cosa i valori iniziali `B` e `C` in una stringa e applicasse l'operatore più al risultato, questo produrrebbe la stringa `11`.

Lo stesso si applica quando si utilizzano le variabili di tipo variante:

```
Dim A
Dim B
Dim C

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

Poiché le variabili di tipo variante possono contenere sia numeri che stringhe, non è chiaro se la variabile `A` è assegnata al numero 2 o alla stringa 11.

Le occasioni di errore qui indicate per le conversioni di tipo implicite possono essere evitate solo programmando con attenzione, ad esempio non utilizzando il tipo di dati variante.

Per evitare altri errori risultanti dalle conversioni di tipo implicite, StarOffice Basic offre una serie di funzioni di conversione, utilizzabili per definire quando il tipo di dati di un'operazione dovrebbe essere convertito:

- **CStr(Var)** – converte qualsiasi tipo di dati in una stringa.
- **CInt(Var)** – converte qualsiasi tipo di dati in valore intero.
- **CLng(Var)** – converte qualsiasi tipo di dati in un valore intero lungo.
- **CSng(Var)** – converte qualsiasi tipo di dati in un valore singolo.
- **CDbl(Var)** – converte qualsiasi tipo di dati in un valore doppio.
- **CBool(Var)** – converte qualsiasi tipo di dati in un valore booleano.
- **CDate(Var)** – converte qualsiasi tipo di dati in un valore di data.

Potete utilizzare queste funzioni di conversione per definire il modo in cui StarOffice Basic dovrebbe eseguire le operazioni di conversione di tipo:

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

B = 1
C = 1
```

```
A = CStr(B + C)       ' B e C sono prima aggiunti, poi convertiti
                      '  (produce il numero 2)
A = CStr(B) + CStr(C)   ' vengono convertiti in una stringa, quindi
                        ' combinati (produce la stringa "11")
```

Durante la prima addizione dell'esempio, StarOffice Basic aggiunge prima le variabili intere e quindi converte il risultato in una catena di caratteri. Ad `A` è assegnata la stringa `2`. Nel secondo esempio, le variabili intere vengono prima convertite in due stringhe e quindi collegate reciprocamente tramite l'assegnazione. Ad `A` è pertanto assegnata la stringa `11`.

Le funzioni di conversione numerica `CSng` e `CDbl` accettano inoltre i numeri decimali. Il simbolo definito nelle impostazioni specifiche del paese corrispondente deve essere utilizzato come simbolo del punto decimale. Al contrario, i metodi `CStr` utilizzano le impostazioni specifiche del paese attualmente selezionate per la formattazioni di numeri, date e ora.

La funzione `Val` è diversa dai metodi `Csng`, `Cdbl` e `Cstr`. Essa converte una stringa in un numero, ma si aspetta sempre che sia utilizzato un punto come simbolo del punto decimale.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)      ' È convertito correttamente indipendentemente
                ' dalle impostazioni specifiche del paese
```

Controllo del contenuto delle variabili

In alcuni casi, non è possibile convertire la data:

```
Dim A As String
Dim B As Date

A = "test"
B = A            ' Crea un messaggio di errore
```

Nell'esempio illustrato, l'assegnazione della stringa `test` a una variabile data non ha senso, l'interprete Basic restituisce un errore. Lo stesso si verifica quando si cerca di assegnare una stringa a una variabile booleana:

```
Dim A As String
Dim B As Boolean

A = "test"
B = A            ' Crea un messaggio di errore
```

Anche in questo caso, l'interprete Basic restituisce un errore.

Questi messaggi di errore possono essere evitati controllando il programma prima di un'assegnazione, al fine di stabilire se il contenuto della variabile da assegnare corrisponde al tipo della variabile di destinazione. StarOffice Basic fornisce le seguenti funzioni di prova a tal fine:

- **IsNumeric(Valore)** – controlla se un valore è un numero.
- **IsDate(Valore)** – controlla se un valore è una data.
- **IsArray(Valore)** – controlla se un valore è un array.

Queste funzioni sono particolarmente utili quando si richiede l'immissione di valori da parte dell'utente. Ad esempio, è possibile controllare se un utente ha immesso un numero o una data validi.

```
If IsNumeric(UserInput) Then
   ValidInput = UserInput
Else
   ValidInput = 0
   MsgBox "Messaggio di errore."
End If
```

Nell'esempio precedente, se la variabile `InputUtente` contiene un valore numerico valido, quest'ultimo viene assegnato alla variabile `InputValido`. Se `InputUtente` non contiene un numero valido, a `InputValido` è assegnato il valore `0` e viene restituito un messaggio di errore.

Sebbene in StarOffice Basic esistano funzioni di prova per il controllo dei numeri, delle date e degli array, non esiste una funzione corrispondente per il controllo dei valori logici. La funzionalità può tuttavia essere simulata utilizzando la funzione `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
```

La funzione `IsBoolean` definisce una variabile interna `Dummy` di tipo booleano e cerca di assegnarla al valore trasferito. Se l'assegnazione è riuscita, la funzione restituisce `True`. Se invece non riesce, viene prodotto un errore runtime, che intercetta la funzione di prova per restituire un errore.

Nota – Se una stringa in StarOffice Basic contiene un valore non numerico e se questo viene assegnato a un numero, StarOffice Basic non produce un messaggio di errore, ma trasferisce il valore 0 alla variabile. Questa procedura è differente da quanto accade in VBA. In VBA, se viene eseguita l'assegnazione descritta, viene prodotto un errore e il programma termina.

Stringhe

Utilizzo dei set di caratteri

Per la gestione delle stringhe, StarOffice Basic utilizza il set di caratteri Unicode. Le funzioni `Asc` e `Chr` consentono di determinare il valore Unicode appartenente a un carattere e/o di individuare il carattere corrispondente a un dato valore Unicode. Le espressioni seguenti assegnano i vari valori Unicode alla variabile code:

```
Code = Asc("A")         ' carattere latino A (valore Unicode 65)
Code = Asc("—")         ' carattere dell'euro (valore Unicode 8364)
Code = Asc("??")         ' carattere cirillico ?? (valore Unicode 1083)
```

Per contro, l'espressione

```
Stringa = Chr(13)
```

assicura che la stringa `Stringa` sia inizializzata con il valore del numero `13`, che corrisponde a un'interruzione di riga forzata.

Il comando `Chr` è spesso utilizzato nei linguaggi Basic per inserire caratteri di controllo in una stringa. L'assegnazione

```
Stringa = Chr(9) + "Questa è una prova" + Chr(13)
```

assicura pertanto che il testo sia preceduto da un carattere di tabulazione (valore Unicode 9) e che venga aggiunta un'interruzione di riga forzata (valore Unicode `13`) dopo il testo.

Accesso a parti di una stringa

StarOffice Basic comprende quattro funzioni che restituiscono stringhe parziali:

- **Left(Stringa, Lunghezza)** – restituisce i primi caratteri di `Stringa` per la lunghezza determinata da `Lunghezza`.

- **Right(Stringa, Lunghezza)** – restituisce gli ultimi caratteri di `Stringa` per la lunghezza determinata da `Lunghezza`.
- **Mid(Stringa, Inizio, Lunghezza)** – restituisce i primi caratteri di `Stringa` per la lunghezza determinata da `Lunghezza` a partire dalla posizione `Inizio`.
- **Len(Stringa)** – restituisce il numero di caratteri di `Stringa`.

Di seguito sono riportati alcuni esempi di chiamate per le funzioni qui descritte:

```
Dim Stringa As String
Dim Risultato As String
Dim Lunghezza As Integer

Stringa = "Questa è solo una prova"

Risultato = Left(Stringa,5)      ' Restituisce la stringa "Questa"
Risultato = Right(Stringa,5)     ' Restituisce la stringa "prova"
Risultato = Mid(Stringa, 8, 5)   ' Restituisce la stringa " solo"
Lunghezza = Len(Stringa)         ' Restituisce il valore 23
```

Ricerca e sostituzione

La funzione `InStr` di StarOffice Basic permette di ricercare una stringa parziale all'interno di un'altra stringa:

```
Risultato = InStr (Ricerca, Stringa)
```

Il parametro `Ricerca` specifica la stringa da ricercare in `Stringa`. La funzione restituisce un numero contenente la posizione in cui `Ricerca` compare per la prima volta all'interno di `Stringa`. Per individuare altre corrispondenze della stringa, la funzione offre inoltre l'opportunità di specificare una posizione di partenza iniziale da cui StarOffice Basic inizia la ricerca. In questo caso, la sintassi della funzione è:

```
Risultato = InStr(Inizio, Ricerca, Stringa)
```

Negli esempi precedenti, `InStr` ignora i caratteri in maiuscolo e in minuscolo. Per modificare la ricerca in modo che `InStr` distingua tra maiuscole e minuscole, aggiungere il parametro `0`, come illustrato nell'esempio seguente:

```
Risultato = InStr(Stringa, Ricerca, 0)
```

Utilizzando le funzioni presentate in precedenza per la modifica delle stringhe, i programmatori potranno eseguire ricerche e sostituzioni di una stringa all'interno di un'altra:

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

 Result = ""
 StartPos = 1
```

```
 CurrentPos = 1

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

La funzione esegue un ciclo di ricerca nella stringa `Ricerca` tramite `InStr` nel termine originale `Origine`. Se individua il termine, prende la parte prima dell'espressione e la scrive nel buffer `Risultato`. Aggiunge la sezione `Nuovo` al punto del termine di ricerca `Ricerca`. Se non vengono reperite altre corrispondenze per il termine, la funzione determina la parte della stringa rimasta e la aggiunge al buffer. Restituisce la stringa prodotta in questo modo come risultato del processo di sostituzione.

Poiché la sostituzione di parti di sequenze di caratteri è una delle funzioni utilizzate più frequentemente, la funzione `Mid` in StarOffice Basic è stata ampliata in modo che questa attività sia eseguita automaticamente. L'esempio

```
Dim Stringa As String

Stringa  = "Questo era il mio testo"
Mid(Stringa, 8, 3, "è")
```

sostituisce tre caratteri con la stringa è dall'ottava posizione della stringa `Stringa`.

Formattazione delle stringhe

La funzione `Format` formatta i numeri sotto forma di stringa. Per procedere, la funzione si aspetta che sia specificata un'espressione `Format`, che viene quindi utilizzata come modello per la formattazione dei numeri. Ogni segnaposto all'interno del modello assicura che questa voce sia formattata in modo corrispondente nel valore di output. I cinque segnaposto più importanti all'interno di un modello sono i caratteri *zero* (0), *cancelletto* (#), *punto* (.), *virgola* (,) e *dollaro* ($).

Il carattere *zero* all'interno del modello garantisce che un numero sia sempre situato nel punto corrispondente. Se non è fornito un numero, al suo posto viene visualizzato 0.

Un *punto* corrisponde al simbolo del punto decimale definito dal sistema operativo nelle impostazioni specifiche del paese.

L'esempio seguente mostra come i caratteri `zero` e *punto* possono definire le cifre dopo il punto decimale in un'espressione:

```
Formato = "0.00"

Stringa = Format(-1579.8, Formato)       ' Restituisce "-1579,80"
Stringa = Format(1579.8, Formato)        ' Restituisce "1579,80"
Stringa = Format(0.4, Formato)           ' Restituisce "0,40"
Stringa = Format(0.434, Formato)         ' Restituisce "0,43"
```

Analogamente, si possono aggiungere gli zeri davanti a un numero per ottenere la lunghezza desiderata:

```
Formato = "0000.00"

Stringa = Format(-1579.8, Formato)       ' Restituisce "-1579,80"
Stringa = Format(1579.8, Formato)        ' Restituisce "1579,80"
Stringa = Format(0.4, Formato)           ' Restituisce "0000,40"
Stringa = Format(0.434, Formato)         ' Restituisce "0000,43"
```

La *virgola* rappresenta il carattere utilizzato dal sistema operativo per il separatore delle migliaia e il *cancelletto* rappresenta una cifra o posizione visualizzata solo se richiesto dalla stringa di input.

```
Formato = "#,##0.00"

Stringa = Format(-1579.8, Formato)     ' Restituisce "-1.579,80"
Stringa = Format(1579.8, Formato)        ' Restituisce "1.579,80"
Stringa = Format(0.4, Formato)        ' Restituisce "0,40"
Stringa = Format(0.434, Formato)         ' Restituisce "0,43"
```

Al posto del segnaposto *dollaro,* la funzione `Format` visualizza il simbolo di valuta pertinente definito dal sistema:

```
Formato = "#,##0.00 $"

Stringa = Format(-1579.8, Formato)     ' Restituisce "-1.579,80 –"
Stringa = Format(1579.8, Formato)        ' Restituisce "1.579,80 –"
Stringa = Format(0.4, Formato)        ' Restituisce "0,40 –"
Stringa = Format(0.434, Formato)         ' Restituisce "0,43 –"
```

Nota – Le istruzioni utilizzate in VBA per la formattazione della data e dell'ora non sono supportate in StarOffice Basic.

Data e ora

StarOffice Basic fornisce il tipo di dati `Date`, che salva le informazioni sulla data e l'ora in formato binario.

Specifica delle informazioni di data e ora nel codice del programma

Potete assegnare una data a una variabile data tramite l'assegnazione di una stringa semplice:

```
Dim Data As Date

Data = "1.1.2002"
```

Questa assegnazione opera correttamente perché StarOffice Basic converte automaticamente il valore di data definito come stringa in una variabile data. Questo tipo di assegnazione, tuttavia, può causare errori, i valori di data e ora sono definiti e visualizzati in modo diverso nei diversi paesi.

Poiché StarOffice Basic utilizza le impostazioni specifiche del paese del sistema operativo per la conversione di una stringa in un valore di data, l'espressione illustrata in precedenza funziona correttamente solo se le impostazioni specifiche del paese corrispondono alla stringa.

Per evitare questo problema, utilizzate la funzione `DateSerial` per assegnare un valore fisso a una variabile data:

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

Il parametro della funzione deve utilizzare la sequenza: anno, mese, giorno. La funzione garantisce che alla variabile sia effettivamente assegnato il valore corretto, a prescindere dalle impostazioni specifiche del paese.

La funzione `TimeSerial` formatta le informazioni sull'ora nello stesso modo in cui la funzione `DateSerial` formatta le date:

```
Dim Var As Date

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

I parametri devono essere specificati nella sequenza: ore, minuti, secondi.

Estrazione delle informazioni su data e ora

Le funzioni seguenti costituiscono la controparte delle funzioni `DateSerial` e `TimeSerial`:

- **Day(Data)** – restituisce il giorno del mese di `Data`
- **Month(Data)** – restituisce il mese di `Data`
- **Year(Data)** – restituisce l'anno di `Data`
- **Weekday(Data)** – restituisce il giorno della settimana di `Data`
- **Hour(Ora)** – restituisce le ore di `Ora`
- **Minute(Ora)** – restituisce i minuti di `Ora`
- **Second(Ora)** – restituisce i secondi di `Ora`

Queste funzioni estraggono le sezioni di data o ora da una variabile `Data` specificata. L'esempio

```
Dim Data As Date

' ... Inizializzazione di Data

If Year(Data) = 2003 Then

   ' ... La data specificata è nell'anno 2003

End If
```

controlla se la data salvata in `Data` è nell'anno 2003. Analogamente, l'esempio

```
Dim Ora As Date

' ... Inizializzazione di Ora

If Hour(Ora) >= 12 And Hour(Ora) < 14 Then

   ' ... L'ora specificata è tra le 12 e le 14.

End If
```

controlla se `Ora` è tra le 12 e le 14.

La funzione `Weekday` restituisce il numero del giorno della settimana per la data trasferita:

```
Dim Data As Date
Dim GiornoSettimana As String

' ... inizializzazione di Data

Select Case GiornoSettimana(Data)
case 1
   GiornoSettimana = "Domenica"
case 2
   GiornoSettimana = "Lunedì"
case 3
```

```
   GiornoSettimana = "Martedì"
case 4
   GiornoSettimana = "Mercoledì"
case 5
   GiornoSettimana = "Giovedì"
case 6
   GiornoSettimana = "Venerdì"
case 7
   GiornoSettimana = "Sabato"
End Select
```

Nota – Domenica è considerato il primo giorno della settimana.

Richiamo della data e dell'ora di sistema

Le seguenti funzioni di StarOffice Basic permettono di richiamare la data e l'ora di sistema:

- **Date** – restituisce la data attuale
- **Time** – restituisce l'ora attuale
- **Now** – restituisce la data e l'ora attuali come valore combinato

File e directory

L'utilizzo dei file è una delle attività di base di un'applicazione. La API StarOffice fornisce un'intera gamma di oggetti con cui potete creare, aprire e modificare i documenti Office, Questi oggetti sono presentati nel Capitolo 4. In ogni caso, talvolta occorre accedere direttamente al file system, eseguire ricerche nelle directory o modificare i file di testo. La libreria runtime di StarOffice Basic mette a vostra disposizione numerose funzioni fondamentali per queste attività.

Nota – Alcune funzioni di file e directory specifiche DOS non sono più incluse in StarOffice 8, o la loro funzionalità è limitata. Ad esempio, non è contemplato il supporto delle funzioni `ChDir`, `ChDrive` e `CurDir`. Alcune proprietà specifiche del DOS non sono più utilizzate nelle funzioni che prevedono proprietà dei file come parametri (ad esempio, per differenziare tra file nascosti e file di sistema). Questa modifica si è resa necessaria per garantire il massimo livello possibile di indipendenza dalla piattaforma per StarOffice.

Amministrazione dei file

Ricerca nelle directory

La funzione `Dir` in StarOffice Basic esegue la ricerca di file e sottodirectory nelle directory. Alla prima richiesta, deve essere assegnata a `Dir` come suo primo parametro una stringa contenente il percorso delle directory in cui eseguire la ricerca. Il secondo parametro di `Dir` specifica il file o la directory da cercare. StarOffice Basic restituisce il nome della prima voce di directory individuata. Per richiamare la voce successiva, la funzione `Dir` dovrebbe essere richiesta senza parametri. Se la funzione `Dir` non individua più voci, restituisce una stringa vuota.

L'esempio seguente mostra come utilizzare la funzione `Dir` per richiedere tutti i file situati in una directory. La procedura salva i singoli nomi di file nella variabile `TuttiFile` e quindi la visualizza in una finestra di messaggi.

```
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
```

Lo `0` utilizzato come secondo parametro nella funzione `Dir` assicura che `Dir` restituisca solo i nomi dei file e che le directory siano ignorate. I parametri seguenti possono essere specificati qui:

- `0` : restituisce i file normali
- `16` : sottodirectory

L'esempio seguente è praticamente identico a quello precedente, ma la funzione `Dir` trasferisce il valore 16 come parametro, che restituisce le sottodirectory di una cartella invece dei nomi dei file.

```
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
```

Nota – Quando richiesto in StarOffice Basic, a differenza di VBA, la funzione `Dir`, utilizzata con il parametro 16, restituisce solo le sottodirectory di una cartella. In VBA, la funzione restituisce anche i nomi dei file standard in modo che sia necessario un ulteriore controllo per richiamare le sole directory. Se si utilizza la funzione `CompatibilityMode ( true )`, StarOffice Basic si comporta come VBA e la funzione Dir, usando il parametro 16, restituisce le sottodirectory e i file standard.

Nota – Le opzioni fornite in VBA per la ricerca specifica nelle directory dei file con le proprietà *concealed*, *system file*, *archived* e *volume name* non esistono in StarOffice Basic perché le funzioni di file system corrispondenti non sono disponibili su tutti i sistemi operativi.

Nota – Le specifiche di percorso elencate in `Dir` possono utilizzare i segnaposto * e ? sia in VBA che in StarOffice Basic. In StarOffice Basic, a differenza di VBA, il segnaposto * può tuttavia essere solo l'ultimo carattere di un nome e/o estensione di file.

Creazione ed eliminazione di directory

StarOffice Basic dispone della funzione `MkDir` per la creazione di directory.

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

Questa funzione permette di creare directory e sottodirectory. Se richiesto, si possono creare anche tutte le directory necessarie all'interno di una gerarchia. Ad esempio, se esiste solo la directory `C:\SubDir1`, la chiamata

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

crea sia la directory `C:\SubDir1\SubDir2` che la directory `C:\SubDir1\SubDir2\SubDir3`.

La funzione `RmDir` permette di eliminare le directory.

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

Qualora la directory contenga sottodirectory o file, verranno *eliminati anch'essi*. Si consiglia pertanto di utilizzare `RmDir` con cautela.

Nota – In VBA, le funzioni `MkDir` e `RmDir` sono relative alla sola directory corrente. In StarOffice Basic `MkDir` e `RmDir` si possono invece utilizzare per creare o eliminare livelli di directory.

Nota – In VBA, `RmDir` produce un messaggio di errore se una directory contiene un file. In StarOffice Basic, vengono eliminati la directory *e tutti i suoi file*. Se si utilizza la funzione `CompatibilityMode ( true )`, StarOffice Basic si comporta come VBA.

Copia, rinomina, eliminazione e controllo dell'esistenza dei file

La chiamata

```
FileCopy(Origine, Destinazione)
```

crea una copia del file `Source` sotto il nome di `Destinazione`.

Con l'ausilio della funzione

```
Name VecchiNome As NuovoNome
```

è possibile rinominare il file `VecchioNome` in `NuovoNome`. La sintassi della parola chiave `As` e il fatto che non è utilizzata una virgola risalgono alle radici del linguaggio Basic.

La chiamata

```
Kill(Nomefile)
```

elimina il file `Nomefile`. Per eliminare le directory (e i relativi file) avvalersi della funzione `RmDir`.

La funzione `FileExists` può essere utilizzata per controllare se esiste un file:

```
If FileExists(Nomefile) Then
   MsgBox "il file esiste."
End If
```

Lettura e modifica delle proprietà dei file

Quando si lavora con i file, è talvolta importante essere in grado di stabilire le proprietà dei file, l'ora dell'ultima modifica del file e la sua lunghezza.

La chiamata

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

restituisce alcune proprietà su un file. Il valore restituito è fornito come maschera di bit in cui sono possibili i valori seguenti:

- 1 : file di sola lettura
- 16 : nome di una directory

L'esempio

```
Dim FileMask As Integer
Dim FileDescription As String

FileMask = GetAttr("test.txt")
If (FileMask AND 1) > 0 Then
   FileDescription = FileDescription & " sola lettura "
End IF

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

If FileDescription = "" Then
   FileDescription = " normale "
End IF

MsgBox DescrizioneFile
```

determina la maschera di bit del file `test.txt` e controlla se è di sola lettura e se si tratta di una directory. Se nessuna delle due è applicabile, a `DescrizioneFile` è assegnata la stringa "normale".

Nota – I flag utilizzati in VBA per ricercare le proprietà *concealed*, *system file*, *archived* e *volume name* dei file non sono supportati in StarOffice Basic in quanto specifici di Windows e non disponibili (o solo in parte) in altri ambienti operativi.

La funzione `SetAttr` permette di modificare le proprietà di un file. La chiamata

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

può pertanto essere utilizzata per fornire un file con stato di sola lettura. Per eliminare uno stato di sola lettura preesistente, avvalersi della chiamata seguente:

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

La data e l'ora dell'ultima modifica operata su un file sono fornite dalla funzione `FileDateTime`. Qui la data è formattata in conformità alle impostazioni specifiche del paese utilizzate sul sistema.

```
FileDateTime("test.txt")   ' Restituisce la data e l'ora dell'ultima
                                        ' modifica del file.
```

La funzione FileLen determina la lunghezza di un file in byte (come numero intero lungo).

```
FileLen("test.txt")      ' Restituisce la lunghezza del file in byte
```

Scrittura e lettura di file di testo

StarOffice Basic mette a disposizione una gamma completa di metodi di lettura e scrittura dei file. Le informazioni riportate di seguito sono relative alle operazioni con i file di testo (*non* i documenti di testo).

Scrittura di file di testo

Prima di poter accedere a un file di testo occorre aprirlo. Per eseguire questa operazione, è necessario un *descrittore di file* libero, che identifichi chiaramente il file per il successivo accesso.

Per creare un descrittore di file libero, avvalersi della funzione FreeFile. Il descrittore di file è utilizzato come parametro per l'istruzione Open, che apre il file. Per aprire un file in modo da specificarlo come file di testo, usare la chiamata Open seguente:

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

Filename è una stringa contenente il nome del file. FileNo è il descrittore creato dalla funzione FreeFile.

Una volta aperto il file, l'istruzione Print può essere descritta riga per riga:

```
Print #FileNo, "Questa è una riga di prova."
```

FileNo rappresenta anche qui il descrittore di file. Il secondo parametro specifica il testo da salvare come riga del file di testo.

Una volta completato ili processo di scrittura, il file deve essere chiuso utilizzando una chiamata Close:

```
    Close #FileNo
```

Anche qui dovrete specificare il gestore di file.

L'esempio seguente mostra come aprire, descrivere e chiudere un file di testo:

```
Dim NumFile As Integer
Dim Riga As String
Dim Nomefile As String

Nomefile = "c:\data.txt"            ' Definisce il nome del file
NumFile = Freefile              ' Determina il descrittore di file libero
```

```
Open Filename For Output As #FileNo         ' Apre il file (in scrittura)
Print #NumFile, "Questa è una riga di testo" ' Salva la riga
Print #NumFile, "Questa è un'altra riga di testo"    ' Salva la riga
Close #NumFile                 ' Chiude il file
```

Lettura di file di testo

La lettura dei file di testo avviene secondo le stesse modalità della scrittura. L'istruzione `Open` utilizzata per aprire il file contiene l'espressione `For Input` al posto dell'espressione `For Output` e, utilizza l'istruzione Line Input per leggere i dati al posto del comando Print per scriverli.

Infine, quando si richiama un file di testo, viene impiegata l'istruzione

```
eof(FileNo)
```

per controllare se è stata raggiunta la fine del file.

L'esempio seguente illustra come eseguire la lettura di un file di testo:

```
Dim NumFile As Integer
Dim Riga As String
Dim File As String
Dim Msg as String

' Definisce il nome del file
Filename = "c:\data.txt"

' Determina il descrittore di file libero
NumFile = Freefile

' Apre il file (modalità di lettura)
Open Filename For Input As NumFile

' Controlla se è stata raggiunta la fine del file

Do While not eof(NumFile)
   ' Legge la riga
   Line Input #NumFile, Riga
   If Riga <>"" then
      Msg = Msg & Riga & Chr(13)
   end if
Loop

' Chiude il file
Close #NumFile

Msgbox Msg
```

Le singole righe sono richiamate in un ciclo `Do While`, salvato nella variabile `Msg`, e visualizzate alla fine in una finestra di messaggi.

Caselle di messaggi e digitazione

StarOffice Basic fornisce le funzioni `MsgBox` e `InputBox` per le comunicazioni base degli utenti.

Visualizzazione dei messaggi

`MsgBox` visualizza una semplice casella di informazioni, che può avere uno o più pulsanti. Nella sua variante più semplice

```
MsgBox "Questa è un'informazione!"
```

`MsgBox` contiene solo il testo e un pulsante OK.

L'aspetto della casella di informazioni può essere modificato utilizzando un parametro. Questo parametro consente di aggiungere ulteriori pulsanti, definire il pulsante pre-assegnato e inserire un simbolo di informazione. I valori per la selezione dei pulsanti sono i seguenti:

- 0 – Pulsante OK
- 1 – Pulsanti OK e Annulla
- 2 – Pulsanti Annulla e Riprova
- 3 – Pulsanti Sì, No e Annulla
- 4 – Pulsanti Sì e No
- 5 – Pulsanti Riprova e Annulla

Per impostare un pulsante come standard, aggiungere uno dei valori seguenti al valore del parametro delle selezioni dell'elenco di pulsanti. Ad esempio, per creare i pulsanti Sì, No e Annulla (valore 3) in cui Annulla è l'impostazione predefinita (valore 512), il valore del parametro sarà 3 + 512 = 515.

- 0 – Il primo pulsante è il valore standard
- 256 – Il secondo pulsante è il valore standard
- 512 – Il terzo pulsante è il valore standard

Infine, sono disponibili i seguenti simboli di informazione, visualizzabili mediante l'inserimento dei valori di parametro pertinenti:

- 16 – Simbolo di stop
- 32 – Punto interrogativo
- 48 – Punto esclamativo
- 64 – Simbolo di suggerimento

La chiamata

```
MsgBox "Continuare?",  292
```

visualizza una casella di informazioni con i pulsanti Sì e No (valore 4), in cui il secondo pulsante (No) è impostato come valore standard (valore 256) e inserisce anche un punto interrogativo (valore 32), 4+256+32=292

Se una casella di informazioni contiene diversi pulsanti, dovrete ricercare un valore restituito per determinare quale pulsante è stato premuto. In questo caso sono disponibili i valori seguenti:

- 1 – Ok
- 2 – Annulla
- 4 – Riprova
- 5 – Ignora
- 6 - Sì
- 7 - No

Nell'esempio precedente, il controllo dei valori restituiti potrebbe avere la forma seguente:

```
If MsgBox ("Continuare?",  292) = 6 Then
   ' Viene premuto il pulsante Sì
Else
   ' Viene premuto il pulsante No
End IF
```

Oltre al testo informativo e al parametro per organizzare la casella informativa, `MsgBox` ammette anche un terzo parametro, che definisce il testo per il titolo della casella:

```
MsgBox "Continuare?",  292, "Titolo casella"
```

Qualora non venga specificato un titolo, l'opzione standard è "soffice".

Casella di digitazione delle ricerche di stringhe semplici

La funzione `InputBox` consente all'utente di eseguire ricerche nelle stringhe semplici. Costituisce pertanto una semplice alternativa alla configurazione di finestre di dialogo. `InputBox` accetta tre parametri standard:

- un testo informativo
- un titolo della casella
- un valore standard da inserire nell'area di digitazione

```
InputVal = InputBox("Inserire il valore:", "Test", "valore standard")
```

Come valore restituito, `InputBox` fornisce la stringa immessa dall'utente.

Altre funzioni

Beep

La funzione `Beep` fa sì che il sistema emetta un segnale acustico utilizzabile per avvisare l'utente dell'esecuzione di un'azione errata. `Beep` non ha parametri:

```
Beep   ' crea un segnale acustico informativo
```

Shell

La funzione Shell permette di avviare programmi esterni.

```
Shell(Percorso, Stile, Param)
```

`Percorso` definisce il percorso del programma da eseguire. `Stile` definisce la finestra in cui viene lanciato il programma. Sono ammessi i seguenti valori:

- 0 – Il programma viene attivato e avviato in una finestra nascosta.
- 1 – Il programma viene attivato e avviato in una finestra di dimensioni normali.
- 2 – Il programma viene attivato e avviato in una finestra ridotta a simbolo.
- 3 – Il programma viene attivato e avviato in una finestra a schermo intero.
- 4 – Il programma viene avviato in una finestra di dimensioni normali, senza venire attivato.
- 6 – Il programma viene avviato in una finestra ridotta a simbolo e rimane attivata la finestra corrente.
- 10 – Il programma viene avviato in modalità a tutto schermo.

Il terzo parametro, `Param`, permette di trasferire i parametri della riga di comando al programma da avviare.

Wait

La funzione `Wait` arresta l'esecuzione del programma per un periodo di tempo specificato. Il periodo di attesa è specificato in millisecondi. Il comando

```
Wait 2000
```

specifica un'interruzione di 2 secondi (2000 millisecondi).

Environ

La funzione `Environ` restituisce le variabili ambientali del sistema operativo. A seconda della configurazione e del sistema, è possibile salvare diversi tipi di dati. La chiamata

```
Dim TempDir

TempDir=Environ ("TEMP")
```

determina le variabili ambientali della directory temp del sistema operativo.

CAPITOLO 4

Introduzione alla API StarOffice

La API di StarOffice rappresenta un'interfaccia di programmazione universale per l'accesso a StarOffice. Potete utilizzarla per creare, aprire, modificare e stampare i documenti StarOffice. Vi consente inoltre di estendere le funzionalità di StarOffice con macro personali e di scrivere finestre di dialogo specifiche per le vostre esigenze.

La API di StarOffice può essere utilizzata non solo con StarOffice Basic, ma anche con altri linguaggi di programmazione quali Java e C++. Ciò è possibile grazie alla tecnica UNO (*Universal Network Objects*) che fornisce un'interfaccia verso diversi linguaggi di programmazione.

Questo capitolo spiega come utilizzare StarOffice in StarOffice Basic con l'ausilio di UNO. Descrive inoltre i concetti principali di UNO utili per chi programma in StarOffice Basic. I dettagli su come utilizzare le diverse parti della API StarOffice vengono invece forniti nei capitoli successivi.

UNO (Universal Network Objects)

StarOffice fornisce un'interfaccia di programmazione UNO (Universal Network Objects). Si tratta di un'interfaccia di programmazione orientata agli oggetti che StarOffice suddivide in diversi oggetti che garantiscono un accesso controllato dal programma al pacchetto Office.

Poiché StarOffice Basic è un linguaggio di programmazione procedurale, sono stati aggiungi diversi costrutti che consentono l'uso di UNO.

Per utilizzare un Universal Network Object in StarOffice Basic, sarà necessaria una dichiarazione di variabile per l'oggetto associato. Questa dichiarazione viene effettuata utilizzando l'istruzione `Dim` (consultate il Capitolo 2). Per dichiarare una variabile oggetto dovete usare una designazione di tipo `Object`:

```
Dim Ogg As Object
```

La chiamata dichiara una variabile oggetto denominata `Ogg`.

La variabile oggetto creata deve ora essere inizializzata in modo da poterla utilizzare. Questa operazione viene effettuata impiegando la funzione `createUnoService`:

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

Questa chiamata assegna alla variabile `Ogg` un riferimento al nuovo oggetto creato. `com.sun.star.frame.Desktop` somiglia a un tipo di oggetto che, tuttavia, nella terminologia UNO viene denominato *"servizio"* invece di tipo. Conformemente alla filosofia UNO, Ogg sarebbe descritto come un riferimento a un oggetto che supporta il servizio `com.sun.star.frame.Desktop`. Il termine "servizio" utilizzato in StarOffice Basic corrisponde pertanto ai termini *tipo* e *classe* utilizzati in altri linguaggi di programmazione.

Esiste tuttavia una differenza principale: un Universal Network Object può supportare diversi servizi contemporaneamente. Alcuni servizi UNO a loro volta supportano altri servizi in modo da fornire, attraverso un oggetto, un'intera gamma di servizi. È possibile, ad esempio, che l'oggetto summenzionato, basato sul servizio `com.sun.star.frame.Desktop`, includa anche altri servizi per il caricamento dei documenti e per terminare il programma.

Nota – Mentre la struttura di un oggetto in VBA è definita dalla sua classe di appartenenza, in StarOffice Basic la struttura viene definita tramite i servizi che supporta. Un oggetto VBA è sempre assegnato con precisione a una singola classe. Un oggetto di StarOffice Basic può invece supportare diversi servizi.

Proprietà e metodi

In StarOffice Basic, un oggetto fornisce una gamma di proprietà e metodi richiamabili tramite l'oggetto.

Proprietà

Le *proprietà* sono come le proprietà di un oggetto: ad esempio `Filename` e `Title` per un oggetto `Document`.

Le proprietà sono impostate per mezzo di una semplice assegnazione:

```
Document.Title = "Guida alla programmazione in Basic di StarOffice 8"
Document.Filename = "progman.sxv"
```

Una proprietà, proprio come una normale variabile, dispone di un tipo che definisce quali valori può registrare. Le precedenti proprietà `Filename` e `Title` sono del tipo stringa.

Proprietà reali e proprietà imitate

La maggior parte delle proprietà di un oggetto in StarOffice Basic sono definite come tali nella descrizione UNO del servizio. Oltre a queste proprietà "reali", in StarOffice Basic sono disponibili anche altre proprietà, formate da due metodi al livello di UNO. Uno di essi è utilizzato per ricercare il valore della proprietà e l'altro per impostarla (metodi `get` e `set`). La proprietà è stata virtualmente imitata dai due metodi. Gli oggetti carattere in UNO, ad esempio, forniscono i metodi `getPosition` e `setPosition` attraverso i quali è possibile richiamare e modificare il punto chiave associato. Chi programma in StarOffice Basic può accedere ai valori tramite la proprietà `Position`. Indipendentemente da ciò, sono comunque disponibili anche i metodi originali (nel nostro esempio, `getPosition` e `setPosition`).

Metodi

I metodi possono essere considerati come funzioni in relazione diretta con un oggetto e attraverso i quali esso viene richiamato. L'oggetto `Document` precedente potrebbe, ad esempio, fornire un metodo `Save`, richiamabile nel modo seguente:

```
Document.Save()
```

I metodi, come le funzioni, possono contenere parametri e restituire valori. La sintassi delle chiamate di questi metodi è orientata alle funzioni classiche. La chiamata

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

specifica anche il parametro `True` per l'oggetto documento alla richiesta del metodo Save. Una volta completato il metodo, `Save` salva un valore restituito nella variabile `Ok`.

Moduli, servizi e interfacce

StarOffice mette a disposizione centinaia di servizi che, per garantirne una presentazione generale semplificata, sono stati raggruppati in moduli. I moduli non hanno nessun altra importanza funzionale per chi programma in StarOffice Basic. Quando si specifica il nome di un servizio, è solo il nome del modulo ad avere importanza, perché deve essere elencato anch'esso nel nome del servizio. Il nome completo di un servizio è formato dall'espressione `com.sun.star`, che specifica che

si tratta di un servizio StarOffice, seguito dal nome del modulo, come ad esempio `frame` per finire con il nome effettivo del servizio, come ad esempio `Desktop`. Il nome completo in questo esempio sarebbe quindi:

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

Oltre ai termini modulo e servizio, UNO introduce anche il termine *"interfaccia"*. Sebbene quest'ultimo possa essere familiare ai programmatori in Java, non è utilizzato in Basic.

Un'interfaccia combina diversi metodi. Nel senso stretto della parola, un servizio in UNO non supporta metodi, ma piuttosto interfacce, che a loro volta forniscono metodi diversi. In altre parole, i metodi vengono assegnati (come combinazioni) al servizio nelle interfacce. Questo dettaglio può essere di interesse in particolare per i programmatori in Java o C++, dato che in questi linguaggi è necessaria l'interfaccia per richiedere un metodo. In StarOffice Basic, invece, è irrilevante. I metodi infatti sono richiamati direttamente tramite l'oggetto pertinente.

Per una comprensione della API può tuttavia rivelarsi utile avere l'assegnazione dei metodi alle varie interfacce, poiché molte interfacce sono utilizzate nei diversi servizi. Chi abbia dimestichezza con un'interfaccia potrà trasferire le proprie conoscenze da un servizio ad un altro.

Alcune interfacce centrali, utilizzate molto di frequente, vengono illustrate in dettaglio alla fine di questo capitolo, avviate da servizi diversi.

Strumenti per lavorare con UNO

Per quanto riguarda gli oggetti – o i servizi, secondo la terminologia UNO – la questione rimane quali sono le proprietà, i metodi e le interfacce supportate e come determinarli. Oltre al presente manuale, è possibile ricavare ulteriori informazioni sugli oggetti dalle seguenti fonti: il metodo `supportsService`, i metodi di debug nonché la Developer's Guide e il riferimento della API.

Il metodo supportsService

Una serie di oggetti UNO supporta il metodo `supportsService`, con il quale è possibile determinare se un oggetto supporta un particolare servizio o meno. La chiamata

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

ad esempio, determina se l'oggetto `TextElement` supporta il servizio `com.sun.star.text.Paragraph`.

Proprietà di debug

Ogni oggetto UNO in StarOffice Basic sa quali proprietà, metodi e interfacce contiene già. Fornisce infatti delle proprietà che restituiscono queste informazioni sotto forma di elenco. Le proprietà corrispondenti sono le seguenti:

DBG_properties - restituisce una stringa contenente tutte le proprietà di un oggetto

DBG_methods - restituisce una stringa contenente tutti i metodi di un oggetto

DBG_supportetInterfaces - restituisce una stringa contenente tutte le interfacce che supportano un oggetto.

Il seguente codice di programma mostra come utilizzare `DBG_properties` e `DBG_methods` nelle applicazioni. Si procede prima a creare il servizio `com.sun.star.frame.Desktop` e quindi a visualizzare le proprietà e i metodi supportati nelle caselle di messaggi.

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

MsgBox Obj.DBG_Properties
MsgBox Obj.DBG_methods
```

Per l'uso di `DBG_properties`, si noti che la funzione restituisce tutte le proprietà supportate in via teorica da un particolare servizio. Non viene tuttavia fornita alcuna assicurazione che siano utilizzabili con l'oggetto in questione. Prima di richiamare le proprietà, dovrete pertanto usare la funzione `IsEmpty` per controllare che siano effettivamente disponibili.

API, riferimento

Per ulteriori informazioni sui servizi disponibili e le relative interfacce, metodi e proprietà, consultare il riferimento della API StarOffice, reperibile sul sito `www.openoffice.org`:

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

Presentazione generale di alcune interfacce centrali

Alcune interfacce di StarOffice sono reperibili in molte parti della API StarOffice. Definiscono gruppi di metodi per le operazioni astratte, applicabili a diversi problemi. Di seguito è fornita una presentazione generale delle più diffuse tra queste interfacce.

L'origine degli oggetti viene invece spiegata in maggiore dettaglio più avanti nel corso del manuale. In questa fase, vengono presentati solo alcuni aspetti astratti degli oggetti per i quali la API StarOffice fornisce delle interfacce centrali.

Creazione di oggetti dipendenti dal contesto

La API StarOffice fornisce due opzioni per la creazione di oggetti. Una è reperibile nella funzione `createUnoService` menzionata all'inizio di questo capitolo. `createUnoService` crea un oggetto di uso universale. Questi oggetti e servizi sono noti anche come *servizi indipendenti dal contesto*.

Oltre ai servizi indipendenti dal contesto, esistono anche i *servizi dipendenti dal contesto*, i cui oggetti sono utili solo quando utilizzati assieme ad un altro oggetto. Un oggetto disegno per un foglio elettronico, ad esempio, può esistere quindi solo insieme a questo documento.

Interfaccia `com.sun.star.lang.XMultiServiceFactory`

Gli oggetti dipendenti dal contesto sono generalmente creati per mezzo di un metodo dell'oggetto, dal quale dipende l'oggetto. Il metodo `createInstance`, definito nell'interfaccia `XMultiServiceFactory`, è utilizzato in particolare negli oggetti documento.

L'oggetto disegno di cui sopra, ad esempio, può essere creato come segue utilizzando un oggetto foglio elettronico:

```
Dim RectangleShape As Object

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

Analogamente, potete creare un modello di paragrafo in un documento di testo:

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

Accesso con nome ad oggetti subordinati

Le interfacce `XNameAccess` e `XNameContainer` sono utilizzate negli oggetti che contengono oggetti subordinati, che possono essere identificati utilizzando un nome del linguaggio naturale.

Mentre `XNamedAccess` consente di accedere ai singoli oggetti, `XNameContainer` esegue l'inserimento, la modifica e l'eliminazione degli elementi.

Interfaccia com.sun.star.container.XNameAccess

Un esempio di utilizzo di XNameAccess è fornito dall'oggetto foglio di un foglio elettronico, che combina tutte le pagine all'interno del foglio elettronico. Alle singole pagine si accede utilizzando il metodo getByName da XNameAccess:

```
Dim Sheets As Object
Dim Sheet As Object

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

Il metodo getElementNames fornisce una presentazione generale dei nomi di tutti gli elementi e restituisce come risultato un campo di dati contenente i nomi. L'esempio seguente mostra come determinare e visualizzare in un ciclo tutti i nomi degli elementi di un foglio elettronico:

```
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
```

Il metodo hasByName dell'interfaccia XNameAccess rivela se esiste un oggetto subordinato con un particolare nome all'interno dell'oggetto base. L'esempio seguente visualizza quindi un messaggio che informa l'utente se l'oggetto Spreadsheet contiene una pagina di nome Tabella1.

```
Dim Sheets As Object

Sheets = Spreadsheet.Sheets
If Sheets.HasByName("Tabella1") Then
   MsgBox " Tabella1 disponibile"
Else
   MsgBox "Tabella1 non disponibile"
End If
```

Interfaccia com.sun.star.container.XNameContainer

L'interfaccia XNameContainer esegue l'inserimento, l'eliminazione e la modifica degli elementi subordinati in un oggetto base. Le funzioni responsabili sono insertByName, removeByName e replaceByName.

Di seguito viene riportato un esempio pratico di queste operazioni, che richiama un documento di testo contenente un oggetto StyleFamilies e lo utilizza per rendere disponibili i modelli di paragrafo (ParagraphStyles) del documento.

```
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")
```

La riga `insertByName` inserisce lo stile `NewStyle` sotto al nome dello stesso nome nell'oggetto `ParagraphStyles`. La riga `replaceByName` line modifica in `NewStyle` l'oggetto dietro a `ChangingStyle`. Infine, la chiamata `removeByName` rimuove l'oggetto dietro ad `OldStyle` da `ParagraphStyles`.

Accesso basato su indice ad oggetti subordinati

Le interfacce `XIndexAccess` e `XIndexContainer` sono utilizzate negli oggetti che contengono oggetti subordinati e che possono essere identificati utilizzando un indice.

`XIndexAccess` fornisce i metodi per accedere ai singoli oggetti. `XIndexContainer` fornisce i metodi per inserire e rimuovere elementi.

Interfaccia `com.sun.star.container.XIndexAccess`

`XIndexAccess` fornisce i metodi `getByIndex` e `getCount` per richiamare gli oggetti subordinati. `getByIndex` fornisce un oggetto con un indice particolare. `getCount` restituisce il numero di oggetti disponibili.

```
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)
   ' Modifica del foglio
Next I
```

L'esempio mostra un ciclo che viene eseguito attraverso tutti gli elementi del foglio, uno dopo l'altro, e salva un riferimento verso ciascuno nella variabile oggetto `Tabella`. Quando si lavora con gli indici, `getCount` restituisce il numero di elementi. Gli elementi in `getByIndex` sono tuttavia numerati a partire da 0. La variabile conteggio del ciclo pertanto esegue da 0 a `getCount()-1`.

Interfaccia com.sun.star.container.XIndexContainer

L'interfaccia XIndexContainer fornisce le funzioni insertByIndex e removeByIndex. I parametri sono strutturati nello stesso modo delle funzioni corrispondenti in XNameContainer.

Accesso iterativo ad oggetti subordinati

In alcuni casi, un oggetto può contenere un elenco di oggetti subordinati non identificabili tramite un nome o un indice. In queste situazioni, sono idonee le interfacce XEnumeration and XenumerationAccess, che forniscono un meccanismo tramite il quale è possibile passare passo per passo tutti gli elementi subordinati di un oggetto, senza dover utilizzare l'identificazione diretta.

Interfacce com.sun.star.container.XEnumeration e XenumerationAccess

L'oggetto base deve fornire l'interfaccia XEnumerationAccess, che contiene solo un metodo createEnumeration. Questo restituisce un oggetto ausiliario, che a sua volta fornisce l'interfaccia XEnumeration con i metodi hasMoreElements e nextElement. Attraverso di essi è possibile accedere agli oggetti subordinati.

L'esempio seguente attraversa tutti i paragrafi di un testo:

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

L'esempio crea prima un oggetto ausiliario ParagraphEnumeration. Questo gradualmente restituisce i singoli paragrafi del testo in un ciclo. Il ciclo viene terminato non appena il metodo hasMoreElements restituisce il valore False, che segnala il raggiungimento della fine del testo.

CAPITOLO 5

Uso dei documenti di StarOffice

La API StarOffice è stata strutturata in modo da poterne utilizzare universalmente il maggior numero possibile di parti per operazioni diverse. Questo comprende le interfacce e i servizi per creare, aprire, salvare, convertire e stampare i documenti e per la gestione dei modelli. Visto che queste aree funzionali sono disponibili in tutti i tipi di documenti, vengono presentate per prime nel presente capitolo.

Lo StarDesktop

Quando si utilizzano i documenti, due servizi in particolare vengono impiegati molto frequentemente:

- Il servizio `com.sun.star.frame.Desktop`, simile al servizio core di StarOffice, fornisce le funzioni per l'oggetto cornice di StarOffice, sotto al quale sono classificate tutte le finestre dei documenti.Utilizzando questo servizio potrete inoltre creare, aprire e importare i documenti.
- Le funzionalità di base per i singoli oggetti documento sono garantite dal servizio `com.sun.star.document.OfficeDocument`, che fornisce i metodi per salvare, esportare e stampare i documenti.

Il servizio `com.sun.star.frame.Desktop` si apre automaticamente all'avvio di StarOffice. Per eseguire questa attività, StarOffice crea un oggetto raggiungibile tramite il nome globale `StarDesktop`.

L'interfaccia più importante dello `StarDesktop` è `com.sun.star.frame.XcomponentLoader`, che include il metodo `loadComponentFromURL` responsabile della creazione, dell'importazione e dell'apertura dei documenti.

Il nome dell'oggetto `StarDesktop` risale a StarOffice 5, in cui tutte le finestre dei documenti erano integrate in un'unica applicazione comune denominata `StarDesktop`. Nella presente versione di StarOffice, lo `StarDesktop` non è più visibile. Il nome `StarDesktop` è stato tuttavia mantenuto per l'oggetto cornice di StarOffice perché indica chiaramente che si tratta dell'oggetto base dell'intera applicazione.

L'oggetto `StarDesktop` assume sostituisce l'oggetto `Application` di StarOffice 5, che fungeva in precedenza da oggetto radice. A differenza del vecchio oggetto `Application`, questo oggetto è tuttavia principalmente responsabile dell'apertura di nuovi documenti. Le funzioni residenti nel vecchio oggetto `Application` per il controllo della rappresentazione su schermo di StarOffice (ad esempio `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) non sono più utilizzate.

Nota – Mentre i documenti attivi in Word sono accessibili tramite `Application.ActiveDocument` e in Excel tramite `Application.ActiveWorkbook`, in StarOffice è `StarDesktop` responsabile di questa attività. L'accesso all'oggetto documento attivo è eseguito in StarOffice 7 tramite la proprietà `StarDesktop.CurrentComponent`.

Informazioni di base sui documenti in StarOffice

Quando si utilizzano i documenti StarOffice, è utile conoscere alcune delle informazioni di base sulla gestione dei documenti in StarOffice. Tra queste, le modalità di strutturazione dei nomi dei file per i documenti StarOffice, nonché il formato di salvataggio dei file.

Nomi dei file nella notazione URL

Poiché StarOffice è un'applicazione indipendente dalla piattaforma, utilizza la notazione URL (che è indipendente da qualsiasi sistema operativo) come definito nell'Internet Standard RFC 1738 per i nomi di file. I nomi di file standard che utilizzano questo sistema iniziano con il prefisso

```
file:///
```

seguito dal percorso. Se il nome del file contiene sottodirectory, queste sono separate da una singola barra, *non* dalla barra retroversa generalmente utilizzata in Windows. Il seguente percorso fa riferimento al file `test.sxw` nella directory doc sull'unità C:.

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

Per convertire i nomi di file locali in un URL, StarOffice dispone della funzione `ConvertToUrl`. Per convertire un URL in un nome di file locale, StarOffice dispone della funzione `ConvertFromUrl`.

```
MsgBox ConvertToUrl("C:\doc\test.sxw")
      ' fornisce il file:///C:/doc/test.sxw
MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
      '  fornisce (in Windows) c:\doc\test.sxw
```

L'esempio converte un nome di file locale in un URL e lo visualizza in una finestra di messaggio. Quindi converte un URL in un nome di file locale e visualizza anche questo.

Lo standard Internet RFC 1738, su cui si basa, consente l'uso dei caratteri `0-9`, `a-z` e `A-Z`. Tutti gli altri caratteri sono inseriti negli URL come codice di escape. Per eseguire questa operazione, vengono convertiti nei relativi valori esadecimali nel set di caratteri ISO 8859-1 (ISO-Latin) e sono preceduti da un segno di percento. Quindi, ad esempio, uno spazio in un nome di file locale diventa `%20` nell'URL.

Formato XML

Dalla versione 6.0, StarOffice utilizza un formato di file basato su XML. Tramite l'uso di XML, l'utente ha la possibilità di aprire e modificare i file anche in altri programmi.

Compressione dei file

Poiché XML si basa su un normale file di testo, i file risultanti sono generalmente di grandi dimensioni. StarOffice, quindi, li comprime e li salva sotto forma di file ZIP. Per mezzo delle opzioni del metodo `storeAsURL`, l'utente può salvare direttamente i file XML originali. Vedere "Opzioni del metodo `storeAsURL`" a pagina 87.

Creazione, apertura e importazione di documenti

I documenti vengono aperti, importati e creati utilizzando il metodo

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

Il primo parametro di `loadComponentFromURL` specifica l'URL del file associato.

Come secondo parametro, `loadComponentFromURL` prevede un nome per l'oggetto cornice della finestra che StarOffice crea internamente per la sua amministrazione. Il nome predefinito `_blank` è generalmente specificato in questo punto e ciò garantisce che StarOffice crei una nuova finestra. In alternativa, potete specificare anche `_hidden`, che garantisce che il documento corrispondente sia caricato ma rimanga invisibile.

Utilizzando questi parametri, l'utente può aprire un documento di StarOffice, poiché agli ultimi due parametri si possono assegnare dei segnaposti (valori di esempio):

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

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

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

La chiamata precedente apre il file `text.sxw` e lo visualizza in una nuova finestra.

In questo modo potete aprire in StarOffice Basic qualsiasi numero di documenti e quindi modificarli utilizzando gli oggetti documento restituiti.

Nota – `StarDesktop.loadComponentFromURL` sostituisce i metodi `Documents.Add` e `Documents.Open` della precedente API di StarOffice.

Sostituzione del contenuto della finestra del documento

I valori `_blank` e `_hidden` per il parametro `Frame` assicurano che StarOffice crei una nuova finestra per ogni chiamata da `loadComponentFromURL`. In alcune situazioni, è utile sostituire il contenuto di una finestra già esistente. In questo caso, l'oggetto cornice della finestra dovrebbe contenere un nome esplicito. Si noti che questo nome non deve iniziare con trattino di sottolineatura. Inoltre, il parametro `SearchFlags` deve essere impostato in modo che venga creata la struttura corrispondente, qualora non esista già. La costante corrispondente per `SearchFlags` è:

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

L'esempio seguente mostra come sostituire il contenuto di una finestra aperta con l'ausilio del parametro di cornice e di `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, "Cornice", _
      SearchFlags, Dummy)

MsgBox "Premere OK per visualizzare il secondo documento."

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

L'esempio apre prima il file `test.sxw` in una nuova finestra con il nome `Cornice`. Una volta confermata la finestra dei messaggi, sostituisce il contenuto della finestra con il file `test2.sxw`.

Opzioni del metodo `loadComponentFromURL`

Il quarto parametro della funzione `loadComponentFromURL` è un campo di dati `PropertyValue` che offre a StarOffice diverse opzioni per aprire e creare documenti. Il campo di dati deve fornire una struttura `PropertyValue` per ogni opzione in cui è salvato come stringa il nome dell'opzione nonché il valore associato.

`loadComponentFromURL` supporta le opzioni seguenti:

- **AsTemplate (valore logico)** – se il valore è vero, carica un nuovo documento senza titolo dall'URL dato. Se il valore è falso, i file dei modelli vengono caricati per la modifica.
- **CharacterSet (String)** – definisce su quale set di caratteri si basa un documento.
- **FilterName (String)** – specifica un filtro speciale per la funzione `loadComponentFromURL`. I nomi di filtro disponibili sono definiti nel file `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **FilterOptions (String)** – definisce le opzioni addizionali per i filtri.
- **JumpMark (String)** – una volta aperto un documento, passa alla posizione definita in JumpMark.
- **Password (String)** – trasferisce una password per un file protetto.
- **ReadOnly (Boolean)** – carica un documento in sola lettura.

L'esempio seguente mostra come aprire un file di testo separato da virgole in StarOffice Calc utilizzando l'opzione `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())
```

Il campo di dati di `FileProperties` copre precisamente un valore perché registra un'opzione. La proprietà `Filtername` definisce se StarOffice utilizza un filtro di testo di StarOffice Calc per aprire i file.

Creazione di nuovi documenti

StarOffice crea automaticamente un nuovo documento se il documento specificato nell'URL è un modello.

In alternativa, se è necessario solo un documento vuoto senza alcun adattamento, potete specificare un URL `private:factory`:

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

```
Dim Doc As Object

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

La chiamata crea un documento vuoto di StarOffice Writer.

Oggetti documento

La funzione `loadComponentFromURL` presentata nella sezione precedente restituisce un oggetto documento. Supporta il servizio `com.sun.star.document.OfficeDocument`, che a sua volta fornisce due interfacce centrali:

- l'interfaccia `com.sun.star.frame.XStorable`, responsabile del salvataggio dei documenti e
- l'interfaccia `com.sun.star.view.XPrintable`, che contiene i metodi per la stampa dei documenti.

Nota – Al passaggio a StarOffice 8, è possibile riscontrare che la portata funzionale degli oggetti documento è rimasta in gran parte la stessa. Gli oggetti documento, ad esempio, forniscono ancora i metodi per salvare e stampare i documenti. I nomi e i parametri dei metodi sono tuttavia cambiati.

Salvataggio ed esportazione dei documenti

I documenti di StarOffice vengono salvati direttamente tramite l'oggetto documento. Il metodo `store` dell'interfaccia `com.sun.star.frame.XStorable` è disponibile a tal fine:

```
Doc.store()
```

Questa chiamata funziona solo se al documento è già stata assegnata una quantità di memoria. Ma ciò non avviene con i nuovi documenti. In questo esempio, è utilizzato il metodo `storeAsURL`. Questo metodo è definito anche in `com.sun.star.frame.XStorable` e può essere utilizzato per definire la posizione del documento:

```
Dim URL As String
Dim Dummy()

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

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

Oltre ai metodi sopracitati, `com.sun.star.frame.XStorable` fornisce anche alcuni metodi utili per il salvataggio dei documenti:

- **hasLocation()** – specifica se al documento è già stato assegnato un URL.
- **isReadonly()** – specifica se un documento ha la protezione di sola lettura.
- **isModified()** – specifica se un documento è stato modificato dall'ultimo salvataggio.

Il codice per il salvataggio di un documento è stato ampliato da queste opzioni in modo che il documento sia salvato solo se l'oggetto è stato effettivamente modificato e il nome del file viene effettivamente ricercato solo se effettivamente necessario:

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

L'esempio controlla prima se il documento pertinente è stato modificato dall'ultimo salvataggio. Solo in caso di esito positivo il processo di salvataggio prosegue. Se al documento è già stato assegnato un URL e non si tratta di un documento in sola lettura, viene salvato sotto l'URL preesistente. Se invece non ha un URL o se è stato aperto nel suo stato di sola lettura, verrà salvato sotto un nuovo URL.

Opzioni del metodo `storeAsURL`

Come con il metodo `loadComponentFromURL`, potete specificare alcune opzioni anche sotto forma di un campo di dati `PropertyValue` utilizzando il metodo `storeAsURL`. Queste determinano la procedura utilizzata da StarOffice per il salvataggio di un documento. `storeAsURL` dispone delle opzioni seguenti:

- **CharacterSet (String)** – definisce su quale set di caratteri si basa un documento.
- **FilterName (String)** – specifica un filtro speciale per la funzione `loadComponentFromURL`. I nomi di filtro disponibili sono definiti nel file `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **FilterOptions (String)** – definisce le opzioni addizionali per i filtri.
- **Overwrite (Boolean)** – consente la sovrascrittura di un file preesistente senza una ricerca.
- **Password (String)** – trasferisce una password per un file protetto.
- **Unpacked (Boolean)** – salva il documento (non compresso) nelle sottodirectory.

L'esempio seguente mostra come potrete utilizzare l'opzione `Overwrite` insieme a `storeAsURL`:

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

' ... Inizializza Doc
```

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

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

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

L'esempio salva quindi `Doc` con il nome specificato se esiste già un file con quel nome.

Stampa di documenti

Analogamente al salvataggio, è possibile stampare direttamente i documenti tramite l'oggetto documento. Il metodo `Print` dell'interfaccia `com.sun.star.view.Xprintable` è disponibile a tal fine. Nella sua forma più semplice, la chiamata print è:

```
Dim Dummy()

Doc.print(Dummy())
```

Come nel caso del metodo `loadComponentFromURL`, il parametro Dummy è un campo di dati `PropertyValue` attraverso il quale StarOffice può specificare diverse opzioni di stampa.

Le opzioni del metodo print

Il metodo `print` prevede un campo di dati `PropertyValue` come parametro, che riflette le impostazioni della finestra di stampa di StarOffice:

- **CopyCount (Intero)** – specifica il numero di copie da stampare.
- **FileName (String)** – stampa il documento nel file specificato.
- **Collate (Boolean)** – informa la stampante di ricomporre le pagine delle copie.
- **Sort (Boolean)** – ordina le pagine quando si stampano diverse copie(`CopyCount` > 1).
- **Pages (String)** – contiene l'elenco delle pagine da stampare (sintassi come specificata nella finestra di dialogo).

L'esempio seguente mostra come stampare diverse pagine di un documento utilizzando l'opzione `Pages`:

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

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

Doc.print(PrintProperties())
```

Selezione e impostazioni della stampante

L'interfaccia `com.sun.star.view.XPrintable` fornisce la proprietà `Printer`, che seleziona la stampante. Questa proprietà accetta un campo di dati di `PropertyValue` con le impostazioni seguenti:

- **Name (String)** – specifica il nome delle stampante.
- **PaperOrientation (Enum)** – specifica l'orientamento della carta (valore `com.sun.star.view.PaperOrientation.PORTRAIT` per il formato verticale, `com.sun.star.view.PaperOrientation.LANDSCAPE` per il formato orizzontale).
- **PaperFormat (Enum)** – specifica il formato del foglio di carta (ad esempio, `com.sun.star.view.PaperFormat.A4` per DIN A4 o `com.sun.star.view.PaperFormat.Letter` per US letter).
- **PaperSize (Size)** – specifica il formato della carta in centesimi di millimetro.

L'esempio seguente mostra come modificare la stampante e il formato della carta con l'ausilio della proprietà `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   ' corrisponde a 20 cm
PaperSize.Height = 20000   ' corrisponde a 20 cm

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

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

Doc.Printer = PrinterProperties()
```

L'esempio definisce un oggetto denominato `PaperSize` con il tipo `com.sun.star.awt.Size`. Ciò è necessario per specificare il formato della carta. Inoltre, crea un campo di dati per due voci di `PropertyValue` denominate `PrinterProperties`. Questo campo di dati viene quindi inizializzato con i valori da impostare e assegnati alla proprietà `Printer`. Dal punto di vista di UNO, la stampante non è una proprietà reale, ma imitata.

Modelli di documento

I modelli sono elenchi con nome contenenti attributi di formattazione, che coprono tutte le applicazioni di StarOffice e semplificano notevolmente le operazioni di formattazione. Se l'utente modifica uno degli attributi di un modello, StarOffice regola

automaticamente tutte le sezioni dei documenti a seconda dell'attributo. L'utente può quindi, ad esempio, modificare il tipo di carattere di tutte le intestazioni di livello uno con un'unica modifica centrale nel documento. A seconda dei tipi di documento, StarOffice riconosce un'intera gamma di tipi di modelli diversi.

StarOffice Writer supporta

- modelli di carattere,
- modelli di paragrafo,
- modelli di cornice,
- modelli di pagina
- modelli di numerazione

StarOffice Calc supporta

- modelli di cella
- modelli di pagina

StarOffice Impress supporta

- modelli di elementi di caratteri
- modelli di presentazione

Nella terminologia di StarOffice, i diversi tipi di modelli sono denominati `StyleFamilies` in conformità al servizio `com.sun.star.style.StyleFamily` su cui si basano. Si accede alle `StyleFamilies` tramite l'oggetto documento:

```
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")
```

L'esempio fa uso della proprietà `StyleFamilies` di un foglio elettronico per determinare un elenco contenente tutti i modelli di cella disponibili.

È possibile accedere direttamente ai singoli modelli tramite un indice:

```
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
```

Il ciclo aggiunto rispetto all'esempio precedente visualizza i nomi di tutti i modelli di cella uno dopo l'altro in una casella di messaggio.

Informazioni sulle diverse opzioni di formattazione

Ciascun tipo di modello fornisce un'intera gamma di proprietà di formattazione. Di seguito è riportata una panoramica delle più importanti proprietà di formattazione e i punti in cui vengono trattate:

- Proprietà dei caratteri, Capitolo 6, Documenti di testo, servizio `com.sun.star.style.CharacterProperties`
- Proprietà dei paragrafi, Capitolo 6, Documenti di testo, servizio `com.sun.star.text.Paragraph`
- Proprietà delle celle, Capitolo 7, Fogli elettronici, servizio `com.sun.star.table.CellProperties`
- Proprietà delle pagine, Capitolo 7, Fogli elettronici, servizio `com.sun.star.style.PageStyle`
- Proprietà degli elementi dei caratteri, Capitolo 7, Fogli elettronici, servizi vari

Le proprietà di formattazione non si limitano in alcun modo alle applicazioni utilizzate per la loro presentazione in questo manuale e al contrario possono essere applicate universalmente. Ad esempio, la maggior parte delle proprietà delle pagine descritte nel Capitolo 7, si possono utilizzare non solo in StarOffice Calc, ma anche in StarOffice Writer.

Per ulteriori informazioni sull'utilizzo dei modelli, consultare la sezione *Valori predefiniti per le proprietà dei caratteri e dei paragrafi* nel Capitolo 6.

CAPITOLO 6

Documenti di testo

Oltre alle stringhe, i documenti di testo possono contenere anche informazioni di formattazione, che possono comparire in qualsiasi punto del testo. La struttura è ulteriormente complicata dalle tabelle, che comprendono non solo stringhe monodimensionali, ma anche campi bidimensionali. La maggior parte dei programmi di elaborazione di testi fornisce infine ora l'opzione di inserire oggetti disegno, cornici di testo e altri oggetti all'interno di un testo, che possono trovarsi fuori dal flusso del testo ed essere posizionati in qualsiasi punto della pagina.

Questo capitolo presenta le interfacce centrali e i servizi dei documenti di testo. La prima sezione tratta l'anatomia dei documenti di testo e come utilizzare un programma in StarOffice Basic per eseguire passaggi iterativi tramite un documento di StarOffice. Descrive inoltre paragrafi, parti di paragrafi e relativa formattazione.

La seconda sezione spiega come lavorare in modo efficiente con i documenti di testo. A tal fine, StarOffice fornisce diversi oggetti d'ausilio, come l'oggetto `TextCursor`, le cui funzionalità si estendono oltre quelle specificate nella prima sezione.

La terza sezione esula dalle semplici operazioni con i testi e si concentra su tabelle, cornici di testo, campi di testo, segnalibri, directory di contenuti e altro ancora.

Le informazioni su come creare, aprire, salvare e stampare i documenti sono riportate nel Capitolo 5, perché possono essere utilizzate non solo per i documenti di testo, ma anche per altri tipi di documenti.

La struttura dei documenti di testo

Un documento di testo può contenere essenzialmente quattro tipi di informazioni:

- il testo effettivo
- i modelli per la formattazione di caratteri, paragrafi e pagine

- elementi non testuali quali tabelle, grafici e oggetti disegno
- impostazioni globali per il documento di testo

La presente sezione tratta in particolare il testo e le opzioni di formattazione associate.

Nota – La progettazione della API di StarOffice 8 per StarOffice Writer differisce da quella della versione precedente. La vecchia versione della API si concentrava sul lavoro con l'oggetto Selection, fortemente orientato all'idea dell'interfaccia utente per gli utenti finali, che si focalizzava sul lavoro di evidenziazione controllato dal mouse.

La API di StarOffice 8 ha sostituito i collegamenti tra interfaccia utente e interfaccia di programmazione. Il programmatore dispone pertanto di un accesso parallelo a tutte le parti di un'applicazione e può lavorare contemporaneamente con gli oggetti da diverse sotto-sezioni di un documento. Il vecchio oggetto Selection non è più disponibile.

Paragrafi e parti di paragrafi

Il nucleo di un documento di testo consiste di una sequenza di paragrafi. Ad essi non è attribuito un nome né sono indicizzati, pertanto non esiste un metodo per accedere direttamente ai singoli paragrafi. I paragrafi si possono tuttavia vagliare sequenzialmente con l'ausilio dell'oggetto `Enumeration` descritto nel Capitolo 4. In questo modo, i paragrafi possono essere modificati.

Quando si lavora con l'oggetto `Enumeration`, occorre tenere conto del seguente scenario speciale: non restituisce solo paragrafi, ma anche tabelle (in senso stretto, in StarOffice Writer, una tabella è uno speciale tipo di paragrafo). Prima di accedere a un oggetto restituito, dovete controllare se supporta il servizio `com.sun.star.text.Paragraph` per i paragrafi o il servizio `com.sun.star.text.TextTable` per le tabelle.

L'esempio seguente vaglia i contenuti di un documento di testo in un ciclo e utilizza un messaggio in ogni istanza per informare l'utente se l'oggetto in questione è un paragrafo o una tabella.

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

' Crea l'oggetto documento
Doc = StarDesktop.CurrentComponent

' Crea l'oggetto enumerazione
Enum = Doc.Text.createEnumeration

' esegue un ciclo su tutti gli elementi di testo
While Enum.hasMoreElements
```

```
   TextElement = Enum.nextElement

   If TextElement.supportsService("com.sun.star.text.TextTable") Then
      MsgBox "Il blocco contiene una tabella."
   End If

   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      MsgBox "Il blocco contiene un paragrafo."
   End If
Wend
```

L'esempio crea un oggetto documento `Doc` che fa riferimento al documento di StarOffice attivo. Con l'ausilio di `Doc`, l'esempio crea un oggetto `Enumeration` che vaglia le singole parti del testo (paragrafi e tabelle) e assegna l'elemento attivo all'oggetto `TextElement`. In questo esempio viene usato il metodo `supportsService` per determinare se il `TextElement` sia un paragrafo o una tabella.

Paragrafi

Il servizio `com.sun.star.text.Paragraph` garantisce l'accesso al contenuto di un paragrafo. Il testo nel paragrafo può essere recuperato e modificato utilizzando la proprietà String:

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

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

While Enum.hasMoreElements
   TextElement = Enum.nextElement

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

L'esempio apre il documento di testo attivo e lo analizza con l'ausilio dell'oggetto Enumeration. Utilizza la proprietà `TextElement.String` in tutti i paragrafi per accedere ai paragrafi pertinenti e sostituisce le stringhe `uno`, `due` e `per` con i caratteri `1`, `2` e `x`. La funzione `Replace` utilizzata per la sostituzione non rientra nella portata standard del linguaggio StarOffice Basic. Questo caso illustra la funzione di esempio descritta nel Capitolo 3, nella sezione "Ricerca e sostituzione" a pagina 54.

Nota – Il contenuto della procedura qui descritta per accedere ai paragrafi di un testo è comparabile all'elenco Paragraphs utilizzato in VBA, fornito negli oggetti `Range` e `Document` qui disponibili. Mentre in VBA si accede ai paragrafi tramite i loro numeri (ad esempio, mediante la chiamata `Paragraph(1)`), in StarOffice Basic si deve utilizzare l'oggetto `Enumeration` descritto in precedenza.

Non esiste alcuna controparte in StarOffice Basic per gli elenchi `Characters`, `Sentences` e `Words` forniti in VBA. Si ha tuttavia l'opzione di passare a `TextCursor` che permette di spostarsi a livello di caratteri, frasi e parole (vedere la sezione *TextCursor*).

Parti di paragrafi

L'esempio precedente può modificare il testo come richiesto, ma talvolta può distruggere anche la formattazione.

Ciò perché un paragrafo è a sua volta formato da singoli sub-oggetti. Ciascuno di questi sub-oggetti contiene una propria informazione di formattazione. Se il centro di un paragrafo, ad esempio, contiene una parola in grassetto, sarà rappresentata in StarOffice da tra parti di paragrafo: la parte prima dello stile grassetto, la parola in grassetto e infine la parte dopo il grassetto, che è nuovamente raffigurata come normale.

Se il testo del paragrafo viene ora modificato utilizzando la proprietà `String` del paragrafo, StarOffice elimina prima le vecchie parti del paragrafo e quindi inserisce una nuova parte del paragrafo. La formattazione delle sezioni precedenti va quindi persa.

Per impedire questo effetto, l'utente può accedere alle parti di paragrafo associate invece che all'intero paragrafo. I paragrafi forniscono un proprio oggetto `Enumeration` a tal fine. L'esempio seguente mostra un doppio ciclo che vaglia tutti i paragrafi di un documento di testo e le parti di paragrafi in essi contenute, quindi applica i processi di sostituzione dell'esempio precedente:

```
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

' ciclo su tutti i paragrafi
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration
```

```
      ' ciclo su tutti i sottoparagrafi
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         MsgBox "'" & TextPortion.String & "'"
         TextPortion.String = Replace(TextPortion.String, "you", "U")
         TextPortion.String = Replace(TextPortion.String, "too", "2")
         TextPortion.String = Replace(TextPortion.String, "for", "4")

      Wend

   End If

Wend
```

L'esempio viene eseguito su un documento di testo in un doppio ciclo. Il ciclo esterno fa riferimento ai paragrafi del testo. Il ciclo interno elabora invece le parti di paragrafo in questi paragrafi. Il codice di esempio modifica il contenuto in ciascuna di tali parti di paragrafo utilizzando la proprietà `String` della stringa, come nell'esempio precedente per i paragrafi. Tuttavia, dato che le parti di paragrafo sono modificate direttamente, le loro informazioni di formattazione sono conservate quando si sostituisce la stringa.

Formattazione

Esistono diversi modi per formattare un testo. Quello più semplice consiste nell'assegnare le proprietà di formattazione direttamente alla sequenza di testo ed è denominato formattazione diretta. La formattazione diretta è utilizzata in particolare con i documenti brevi perché i formati possono essere assegnati dall'utente con il mouse. È possibile, ad esempio, evidenziare una certa parola all'interno di un testo utilizzando lo stile grassetto oppure centrare una riga.

Oltre alla formattazione diretta, si possono formattare i testi anche utilizzando i modelli. Questa operazione è denominata formattazione indiretta. Con la formattazione indiretta, l'utente assegna un modello predefinito alla parte di testo pertinente. Se il layout del testo viene modificato in un secondo momento, l'utente deve pertanto modificare solo il modello. StarOffice modifica le modalità di raffigurazione di tutte le parti di testo che utilizzano questo modello.

Nota – In VBA, le proprietà di formattazione di un oggetto sono generalmente distribuite su una vasta gamma di sub-oggetti (ad esempio, `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). Si ha accesso alle proprietà tramite espressioni a cascata (ad esempio, `Range.Font.AllCaps`). In StarOffice Basic, le proprietà di formattazione sono d'altro canto disponibili direttamente, utilizzando gli oggetti pertinenti (`TextCursor, Paragraph` e così via). Nelle due sezioni seguenti viene descritta una presentazione generale delle proprietà dei caratteri e dei paragrafi disponibili in StarOffice.

Nota – Nella precedente API di StarOffice, un testo veniva essenzialmente formattato utilizzando l'oggetto `Selection` e i suoi oggetti subordinati (ad esempio, `Selection.Font`, `Selection.Paragraph` e `Selection.Border`). Nella nuova API, le proprietà di formattazione sono disponibili in ogni oggetto (`Paragraph`, `TextCursor` e così via) e possono essere applicate direttamente. Un elenco delle proprietà dei caratteri e dei paragrafi disponibili è riportato nei paragrafi seguenti.

Proprietà dei caratteri

Le proprietà di formattazione che si riferiscono ai singoli caratteri sono descritte come proprietà dei caratteri e includono il grassetto e il tipo di carattere. Gli oggetti che consentono l'impostazione delle proprietà dei caratteri devono supportare il servizio `com.sun.star.style.CharacterProperties`. StarOffice riconosce una gamma completa di servizi che supportano questo servizio, tra cui i servizi `com.sun.star.text.Paragraph` di cui sopra per i paragrafi nonché i servizi `com.sun.star.text.TextPortion` per le parti dei paragrafi.

Il servizio `com.sun.star.style.CharacterProperties` non fornisce interfacce, ma offre una serie di proprietà attraverso le quali è possibile definire e richiamare le proprietà dei caratteri. Un elenco completo di tutte le proprietà dei caratteri è reperibile nel riferimento della API StarOffice. L'elenco seguente descrive le proprietà più importanti:

- **CharFontName (String)** – nome del tipo di carattere selezionato.
- **CharColor (Long)** – colore del testo.
- **CharHeight (Float)** – altezza del carattere in punti (pt).
- **CharUnderline (Constant group)** – tipo di trattino di sottolineatura (costanti conformi a `com.sun.star.awt.FontUnderline`).
- **CharWeight (Constant group)** – spessore del carattere (costanti conformi a `com.sun.star.awt.FontWeight`).
- **CharBackColor (Long)** – colore di sfondo.
- **CharKeepTogether (Boolean)** – soppressione del testo a capo automatico.
- **CharStyleName (String)** – nome del modello di carattere.

Proprietà dei paragrafi

Le informazioni di formattazione che non fanno riferimento a singoli caratteri ma all'intero paragrafo sono considerate proprietà del paragrafo. Includono la distanza del paragrafo dal bordo della pagina nonché l'interlinea. Le proprietà dei paragrafi sono disponibili tramite il servizio `com.sun.star.style.ParagraphProperties`.

Anche le proprietà dei paragrafi sono disponibili in diversi oggetti. Tutti gli oggetti che supportano il servizio `com.sun.star.text.Paragraph` garantiscono anche il supporto delle proprietà dei paragrafi in `com.sun.star.style.ParagraphProperties`.

Un elenco completo di tutte le proprietà dei paragrafi è reperibile nel riferimento della API StarOffice. Le proprietà dei paragrafi di uso più comune sono le seguenti:

- **ParaAdjust (enum)** – orientamento verticale del testo (costanti conformi a `com.sun.star.style.ParagraphAdjust`).
- **ParaLineSpacing (struct)** – interlinea (struttura conforme a `com.sun.star.style.LineSpacing`).
- **ParaBackColor (Long)** – colore di sfondo.
- **ParaLeftMargin (Long)** – margine sinistro in centesimi di millimetro.
- **ParaRightMargin (Long)** – margine destro in centesimi di millimetro.
- **ParaTopMargin (Long)** – margine superiore in centesimi di millimetro.
- **ParaBottomMargin (Long)** – margine inferiore in centesimi di millimetro.
- **ParaTabStops (Array of struct)** – tipo e posizione delle tabulazioni (array con strutture di Typs `com.sun.star.style.TabStop`).
- **ParaStyleName (String)** – nome del modello del paragrafo.

Esempio: esportazione in HTML semplice

L'esempio seguente dimostra come lavorare con le informazioni di formattazione, esegue un'iterazione nel documento di testo e crea un semplice file HTML. A tal fine, ogni paragrafo è registrato nel proprio elemento HTML `<P>`. All'esportazione, le parti di paragrafo visualizzate in grassetto sono contrassegnate per mezzo di un elemento HTML `<B>`.

```
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

' ciclo su tutti i paragrafi
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
```

```
      Enum2 = TextElement.createEnumeration
      CurLine = "<P>"

      ' ciclo su tutte le parti del paragrafo
      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

      ' emette la riga
      CurLine = CurLine & "</P>"
      Print #FileNo, CurLine

   End If
Wend

' scrive il piè di pagina per l'HTML
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

La struttura base dell'esempio è orientata agli esempi per l'esecuzione su parti di paragrafi di un testo già presentato in precedenza. Sono state aggiunte le funzioni per scrivere il file HTML, nonché un codice di prova che controlla lo spessore del carattere delle parti di testo corrispondenti e assegna alle parti del paragrafo in grassetto un tag HTML.

Valori predefiniti per le proprietà di caratteri e paragrafi

La formattazione *diretta* ha sempre la priorità rispetto alla formattazione *indiretta*. In altre parole, alla formattazione tramite modelli è assegnata un priorità inferiore rispetto alla formattazione diretta in un testo.

Determinare se una sezione di un documento è stata formattata in modo diretto o indiretto non è compito facile. Le barre dei simboli fornite da StarOffice mostrano le proprietà di testo comuni quali tipo di carattere, spessore e dimensioni. Tuttavia, se le impostazioni corrispondenti siano basate su modelli o sulla formattazione diretta del testo non è tuttora chiaro.

StarOffice Basic mette a disposizione il metodo `getPropertyState`, con il quale i programmatori possono controllare come è stata formattata una determinata proprietà. Come parametro, questo assume il nome della proprietà e restituisce una costante che fornisce informazioni sull'origine della formattazione. Sono possibili le risposte seguenti, definite nell'enumerazione di `com.sun.star.beans.PropertyState`:

- **com.sun.star.beans.PropertyState.DIRECT_VALUE** – la proprietà è definita direttamente nel testo (formattazione diretta),

- **com.sun.star.beans.PropertyState.DEFAULT_VALUE** – la proprietà è definita tramite un modello (formattazione indiretta)
- **com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE** – la proprietà non è chiara. Questo stato si presenta, ad esempio, quando si ricercano la proprietà grassetto di un paragrafo, che include sia le parole riportate in grassetto che quelle in carattere normale.

L'esempio seguente mostra come modificare in StarOffice le proprietà di formattazione: Ricerca all'interno del testo le parti dei paragrafi che sono state rappresentate come grassetto utilizzando la formattazione diretta. Se incontra una parte di paragrafo corrispondente, elimina la formattazione diretta utilizzando il metodo `setPropertyToDefault` e assegna un modello di carattere `MyBold` alla parte di paragrafo corrispondente.

```
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

' ciclo su tutti i paragrafi
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration

      ' ciclo su tutte le parti del paragrafo
      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
```

Modifica dei documenti di testo

Nella sezione precedente è stata presentata una serie di opzioni per la modifica dei documenti di testo, concentrandosi sui servizi `com.sun.star.text.TextPortion` e `com.sun.star.text.Paragraph`, che garantiscono l'accesso alle parti dei paragrafi nonché ai paragrafi stessi. Questi servizi sono idonei per le applicazioni in cui si deve modificare il contenuto di un testo in un unico passaggio, tramite l'uso di un ciclo. Ciò non è tuttavia sufficiente per molti problemi. StarOffice dispone del servizio `com.sun.star.text.TextCursor` per le operazioni più complicate, compreso lo spostamento a ritroso all'interno di un documento o lo spostamento basato su frasi e parole piuttosto che `TextPortions`.

Il TextCursor

Un `TextCursor` nella API StarOffice è comparabile al cursore visibile utilizzato in un documento di StarOffice, ovvero contrassegna un determinato punto all'interno del documento di testo e può essere spostato in diverse direzioni tramite l'uso di comandi. Gli oggetti `TextCursor` disponibili in StarOffice Basic non vanno però confusi con il cursore visibile, in quanto si tratta di cose molto diverse.

Nota – La terminologia differisce da quella utilizzata in VBA: In termini di portata della funzione, l'oggetto Range di VBA può essere confrontato con l'oggetto TextCursor di StarOffice ma non – come potrebbe fuorviare il nome – con l'oggetto Range di StarOffice.

L'oggetto TextCursor di StarOffice, ad esempio, fornisce metodi per navigare e modificare il testo inclusi nell'oggetto Range in VBA (ad esempio, MoveStart, MoveEnd, InsertBefore, InsertAfter). Le controparti corrispondenti dell'oggetto TextCursor in StarOffice sono descritte nelle sezioni successive.

Spostamento all'interno di un testo

L'oggetto `TextCursor` in StarOffice Basic agisce in modo indipendente dal cursore visibile di un documento di testo. Una modifica di posizione controllata dal programma di un oggetto `TextCursor` non ha alcun impatto sul cursore visibile. Si possono aprire diversi oggetti `TextCursor` per lo stesso documento e utilizzarli in diverse posizioni reciprocamente indipendenti.

Un oggetto `TextCursor` viene creato utilizzando la chiamata `createTextCursor`:

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

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

L'oggetto Cursor così creato supporta il servizio `com.sun.star.text.TextCursor`, che a sua volta fornisce un'intera gamma di metodi per spostarsi all'interno dei documenti di testo. L'esempio seguente prima sposta il `TextCursor` di dieci caratteri verso sinistra e quindi di tre caratteri verso destra:

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

Un `TextCursor` può evidenziare un'area completa, con un'operazione comparabile all'evidenziazione di un punto del testo tramite il mouse. Il parametro `False` nella chiamata di funzione precedente specifica se l'area attraversata con il movimento del cursore è evidenziata o meno. Ad esempio, il `TextCursor` nell'esempio seguente

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

si sposta prima di dieci caratteri verso destra senza evidenziazione e poi torna indietro di tre caratteri e procede ad evidenziarli. L'area evidenziata dal `TextCursor` inizia pertanto dopo il settimo carattere del testo e termina dopo il decimo.

Di seguito vengono elencati i metodi principali forniti dal servizio `com.sun.star.text.TextCursor` per lo spostamento all'interno del testo:

- **goLeft (Count, Expand)** – salta verso sinistra di un numero di caratteri pari a Count.
- **goRight (Count, Expand)** – salta verso destra di un numero di caratteri pari a Count.
- **gotoStart (Expand)** – passa all'inizio del documento di testo.
- **gotoEnd (Expand)** – passa alla fine del documento di testo.
- **gotoRange (TextRange, Expand)** – passa all'oggetto `TextRange` specificato.
- **gotoStartOfWord (Expand)** – passa all'inizio della parola corrente.
- **gotoEndOfWord (Expand)** – passa alla fine della parola corrente.
- **gotoNextWord (Expand)** – passa all'inizio della parola successiva.
- **gotoPreviousWord (Expand)** – passa all'inizio della parola precedente.
- **isStartOfWord ()** - restituisce `True` se il `TextCursor` è all'inizio di una parola.
- **isEndOfWord ()** - restituisce `True` se il `TextCursor` è alla fine di una parola.
- **gotoStartOfSentence (Expand)** – passa all'inizio della frase corrente.
- **gotoEndOfSentence (Expand)** – passa alla fine della frase corrente.
- **gotoNextSentence (Expand)** – passa all'inizio della frase successiva.
- **gotoPreviousSentence (Expand)** – passa alla fine della frase precedente.
- **isStartOfSentence ()** - restituisce `True` se il `TextCursor` è all'inizio di una frase.

- **isEndOfSentence ()** - restituisce `True` se il `TextCursor` è alla fine di una frase.
- **gotoStartOfParagraph (Expand)** – passa all'inizio del paragrafo corrente.
- **gotoEndOfParagraph (Expand)** – passa alla fine del paragrafo corrente.
- **gotoNextParagraph (Expand)** – passa all'inizio del paragrafo successivo.
- **gotoPreviousParagraph (Expand)** – passa all'inizio del paragrafo precedente.
- **isStartOfParagraph ()** - restituisce `True` se il `TextCursor` è all'inizio di un paragrafo.
- **isEndOfParagraph ()** - restituisce `True` se il `TextCursor` è alla fine di un paragrafo.

Il testo è suddiviso in frasi sulla base dei simboli di frase. I punti, ad esempio, sono interpretati come simboli indicanti la fine delle frasi.

Il parametro `Expand` è un valore logico che specifica se l'area attraversata durante lo spostamento deve essere evidenziata o meno. Tutti i metodi di navigazione restituiscono inoltre un parametro che specifica se lo spostamento è riuscito o se l'azione è stata interrotta per mancanza di testo.

Di seguito vengono elencati alcuni metodi per la modifica delle aree evidenziate utilizzando un `TextCursor` e che supportano anche il servizio `com.sun.star.text.TextCursor`:

- **collapseToStart ()** – ripristina l'evidenziazione e posiziona il `TextCursor` all'inizio dell'area evidenziata in precedenza.
- **collapseToEnd ()** – ripristina l'evidenziazione e posiziona il `TextCursor` alla fine dell'area evidenziata in precedenza.
- **isCollapsed ()** – restituisce `True` se il `TextCursor` non copre attualmente alcuna evidenziazione.

Formattazione del testo con `TextCursor`

Il servizio `com.sun.star.text.TextCursor` supporta tutte le proprietà di caratteri e paragrafi presentate all'inizio di questo capitolo.

L'esempio seguente mostra come utilizzarle insieme a `TextCursor`: vaglia un documento completo e formatta in grassetto la prima parola di ogni frase.

```
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
```

L'esempio crea prima un oggetto documento per il testo che era stato appena aperto. Quindi esegue un'iterazione dell'interno testo frase per frase, evidenzia ogni prima parola e le formatta in grassetto.

Richiamo e modifica dei contenuti del testo

Se il `TextCursor` contiene un'area evidenziata, il testo è disponibile tramite la proprietà `String` dell'oggetto `TextCursor`. L'esempio seguente utilizza la proprietà `String` per visualizzare le prime parole di una frase in una finestra di messaggi:

```
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
```

Analogamente, potete modificare la prima parola di ogni frase avvalendovi della proprietà `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
```

Se il `TextCursor` contiene un'area evidenziata, un'assegnazione alla proprietà `String` permette di sostituirla con un nuovo testo. Se non vi sono aree evidenziate, il testo viene inserito in corrispondenza dell'attuale posizione del `TextCursor`.

Inserimento di codici di controllo

In alcune situazioni occorre modificare non tanto il testo di un documento, quanto la sua stessa struttura. StarOffice fornisce a tal fine speciali codici di controllo, inseriti nel testo e che ne influenzano la struttura. I codici di controllo sono definiti nel gruppo di costanti `com.sun.star.text.ControlCharacter`. Sono disponibili i seguenti codici:

- **PARAGRAPH_BREAK** – interruzione di paragrafo.
- **LINE_BREAK** – testo a capo all'interno di un paragrafo.
- **SOFT_HYPHEN** – possibile punto di sillabazione .
- **HARD_HYPHEN** – punto di sillabazione obbligatorio.
- **HARD_SPACE** – spazio protetto che non viene né distribuito né compresso nel testo giustificato.

Per inserire i codici di controllo, è necessario non solo il cursore ma anche gli oggetti documenti di testo associati. L'esempio seguente inserisce un paragrafo dopo il 20º carattere di un testo:

```
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)
```

Il parametro `False` nella chiamata del metodo `insertControlCharacter` assicura che l'area attualmente evidenziata dal `TextCursor` permanga dopo l'operazione di inserimento. Se il parametro `True` viene trasmesso qui, `insertControlCharacter` sostituisce il testo corrente.

Ricerca di parti del testo

In molte situazioni, è necessario eseguire ricerche all'interno del testo per individuare un termine particolare e modificare il punto corrispondente. Tutti i documenti di StarOffice forniscono a tal fine una speciale interfaccia che opera sempre in conformità allo stesso principio: Prima di un processo di ricerca, dovrete creare un `SearchDescriptor` che definisce l'oggetto della ricerca di StarOffice all'interno del documento. Un `SearchDescriptor` è un oggetto che supporta il servizio `com.sun.star.util.SearchDescriptor` e che può essere creato utilizzando il metodo `createSearchDescriptor` di un documento:

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

Una volta creato il `SearchDescriptor`, questo riceve il testo da cercare:

```
SearchDesc.searchString="testo"
```

Per quanto riguarda la funzione espletata, il `SearchDescriptor` può essere paragonato alla finestra di dialogo di ricerca di StarOffice. Come in quel caso, è possibile inserire le impostazioni necessarie per la ricerca nell'oggetto `SearchDescriptor`.

Le proprietà sono fornite dal servizio `com.sun.star.util.SearchDescriptor`:

- **SearchBackwards (Boolean)** – esegue la ricerca nel testo a ritroso invece che in avanti.
- **SearchCaseSensitive (Boolean)** – prende in considerazione i caratteri maiuscoli e minuscoli durante la ricerca.
- **SearchRegularExpression (Boolean)** – tratta l'espressione di ricerca come un'espressione regolare.
- **SearchStyles (Boolean)** – ricerca il modello di paragrafo specificato all'interno del testo.
- **SearchWords (Boolean)** – ricerca solo parole complete.

La funzione `SearchSimilarity` (o "ricerca per analogia") di StarOffice è disponibile anche in StarOffice Basic. Con questa funzione, StarOffice ricerca un'espressione c he può essere analoga ma non esattamente uguale all'espressione di ricerca. Il numero di caratteri addizionali, eliminati e modificati per queste espressioni può essere definito singolarmente. Di seguito vengono riportate le proprietà associate al servizio `com.sun.star.util.SearchDescriptor`:

- **SearchSimilarity (Boolean)** - esegue una ricerca per analogia.
- **SearchSimilarityAdd (Short)** – numero di caratteri che potete aggiungere per una ricerca per analogia.
- **SearchSimilarityExchange (Short)** - numero di caratteri che potete sostituire come parte di una ricerca per analogia.
- **SearchSimilarityRemove (Short)** – numero di caratteri che potete rimuovere come parte di una ricerca per analogia.
- **SearchSimilarityRelax (Boolean)** – prende in considerazione tutte le regole di deviazione contemporaneamente all'espressione di ricerca.

Una volta preparato il `SearchDescriptor` come richiesto, è possibile applicarlo al documento di testo. I documenti di StarOffice dispongono a tal fine dei metodi `findFirst` e `findNext`:

```
Found = Doc.findFirst (SearchDesc)

Do While Found
   ' Elaborazione risultato di ricerca
   Found = Doc.findNext( Found.End, Search)

Loop
```

L'esempio individua tutte le corrispondenze in un ciclo e restituisce un oggetto `TextRange`, che si riferisce al passaggio di testo reperito.

Esempio: Ricerca per simili

Questo esempio mostra come ricercare in un testo la parola "viaggi" e formattare i risultati in grassetto. Viene utilizzata una ricerca per analogia in modo da reperire non solo il termine "viaggi", ma anche la forma singolare "viaggio" e le eventuali declinazioni, come "viaggiare". Le espressioni individuate differiscono di un massimo di tre lettere dall'espressione di ricerca:

```
Dim SearchDesc As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

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

Nota – L'idea base della ricerca e sostituzione in StarOffice è paragonabile a quella utilizzata in VBA. Entrambe le interfacce forniscono un oggetto, attraverso il quale si possono definire le proprietà di ricerca e sostituzione. Questo oggetto viene quindi applicato all'area di testo richiesta per eseguire l'azione. Mentre l'oggetto ausiliare responsabile in VBA può essere raggiunto tramite la proprietà Find dell'oggetto Range, in StarOffice Basic viene creato dalla chiamata `createSearchDescriptor` o `createReplaceDescriptor` dell'oggetto documento. Persino le proprietà e i metodi di ricerca disponibili differiscono.

Come nella vecchia API di StarOffice, anche nella nuova API la ricerca e la sostituzione di testo vengono eseguite utilizzando l'oggetto documento. Mentre però in precedenza esisteva un oggetto denominato `SearchSettings` dedicato alla definizione delle opzioni di ricerca, le nuove ricerche di oggetti vengono ora eseguite utilizzando un oggetto `SearchDescriptor` o `ReplaceDescriptor` per la sostituzione automatica

del testo. Questi oggetti coprono non solo le opzioni, ma anche il testo di ricerca corrente e, se necessario, la sostituzione di testo associata. Gli oggetti descrittori vengono creati utilizzando l'oggetto documento, completati in conformità al requisiti pertinenti e quindi nuovamente trasferiti all'oggetto documento sotto forma di parametri per i metodi di ricerca.

Sostituzione di parti del testo

Proprio come la funzione di ricerca, la funzione di sostituzione di StarOffice è disponibile anche in StarOffice Basic. La gestione delle due funzioni è identica: anche per il processo di sostituzione è necessario per prima cosa un oggetto speciale che registri i parametri del processo. È denominato `ReplaceDescriptor` e supporta il servizio `com.sun.star.util.ReplaceDescriptor`. Tutte le proprietà del `SearchDescriptor` descritte nel paragrafo precedente sono supportate anche da `ReplaceDescriptor`. Ad esempio, durante un processo di sostituzione, potete attivare e disattivare la distinzione tra maiuscole e minuscole ed eseguire ricerche per analogia.

L'esempio seguente dimostra l'uso di `ReplaceDescriptors` per una ricerca all'interno di un documento di StarOffice.

```
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
```

Le espressioni di ricerca e sostituzione vengono impostate utilizzando le proprietà `SearchString` e `ReplaceString` dei `ReplaceDescriptors`. Il processo effettivo di sostituzione è infine implementato utilizzando il metodo `replaceAll` dell'oggetto documento, che sostituisce tutte le ricorrenze dell'espressione di ricerca.

Esempio: ricerca e sostituzione del testo con le espressioni regolari

La funzione di sostituzione di StarOffice è particolarmente efficace quando utilizzata assieme alle espressioni regolari. Queste ultime vi consentiranno infatti di definire un'espressione di ricerca variabile con segnaposto e caratteri speciali al posto di un valore fisso.

Le espressioni regolari supportate da StarOffice sono descritte in dettaglio nella sezione della guida in linea di StarOffice. Di seguito vengono riportati alcuni esempi:

- Un punto all'interno di un'espressione di ricerca rappresenta qualsiasi carattere. L'espressione di ricerca `s.rto` può quindi rappresentare sia `sirto` che `sarto`.
- Il carattere ^ contrassegna l'inizio di un paragrafo. Tutte le ricorrenze del nome `Pietro` che si trovano all'inizio di un paragrafo possono pertanto essere reperite utilizzando l'espressione di ricerca `^Pietro`.
- Il carattere $ contrassegna la fine di un paragrafo. Tutte le ricorrenze del nome `Pietro` che si trovano alla fine di un paragrafo possono pertanto essere reperite utilizzando l'espressione di ricerca `Pietro$`.
- Un * indica che il carattere precedente può essere ripetuto a piacere e può essere combinato con il punto come segnaposto per qualsiasi carattere. L'espressione `temper.*a`, ad esempio, può rappresentare sia l'espressione `temperanza` che `temperatura`.

L'esempio seguente mostra come rimuovere tutte le righe vuote di un documento di testo con l'ausilio dell'espressione regolare ^$:

```
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)
```

Documenti di testo: molto più che semplice testo

Fino a questo punto, nel capitolo sono stati trattati solo paragrafi di testo e relative parti. Ma i documenti di testo possono contenere anche altri oggetti, tra cui tabelle, disegni, campi di testo e directory. Tutti questi oggetti possono essere ancorati a qualsiasi punto all'interno del testo.

Grazie a queste caratteristiche comuni, tutti questi oggetti supportano in StarOffice un servizio di base comune denominato `com.sun.star.text.TextContent`, che fornisce le seguenti proprietà:

- **AnchorType (Enum)** – determina il tipo di ancoraggio di un oggetto `TextContent` (valori predefiniti conformi all'enumerazione `com.sun.star.text.TextContentAnchorType`).
- **AnchorTypes (sequence of Enum)** – enumerazione di tutti gli `AnchorTypes` che supportano un oggetto `TextContent` speciale.
- **TextWrap (Enum)** – determina il tipo di scorrimento del testo attorno a un oggetto `TextContent` (valori predefiniti conformi all'enumerazione `com.sun.star.text.WrapTextMode`).

Gli oggetti `TextContent` condividono anche alcuni metodi – in particolare, quelli per creare, inserire ed eliminare gli oggetti.

- Per **creare** un nuovo oggetto `TextContent`, utilizzate il metodo `createInstance` dell'oggetto documento.
- Per **inserire** un oggetto, usare il metodo `insertTextContent` dell'oggetto di testo.
- Per **eliminare** un oggetto `TextContent`, usare il metodo `removeTextContent`.

Nelle sezioni seguenti viene presentata una serie di esempi che si avvalgono di questi metodi.

Tabelle

L'esempio seguente crea una tabella con l'ausilio del metodo createInstance descritto in precedenza.

```
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)
```

Una volta creata, la tabella viene impostata sul numero di righe e colonne richieste utilizzando una chiamata `initialize` e quindi inserita nel documento di testo con `insertTextContent`.

Come illustrato nell'esempio, il metodo `insertTextContent` prevede non solo l'inserimento dell'oggetto `Content`, ma anche di altri due parametri:

- un oggetto `Cursor` che determina la posizione di inserimento
- una variabile booleana che specifica se l'oggetto `Content` deve sostituire la selezione corrente del cursore (valore `True`) o se deve essere inserito nel testo prima della selezione corrente (`False`)

Nota – Alla creazione e inserimento delle tabelle in un documento di testo, in StarOffice Basic vengono utilizzati oggetti simili a quelli disponibili in VBA: l'oggetto documento e un oggetto `TextCursor` in StarOffice Basic oppure l'oggetto `Range` come sua controparte in VBA. Il metodo `Document.Tables.Add` esegue l'attività di creare e impostare la tabella in VBA, mentre quest'ultima viene creata in StarOffice Basic, in conformità all'esempio precedente, utilizzando `createInstance`, quindi inizializzata e inserita nel documento tramite `insertTextContent`.

È possibile determinare le tabelle inserite in un documento di testo utilizzando un semplice ciclo. A tal fine è utilizzato il metodo `getTextTables()` dell'oggetto documento di testo:

```
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)
   ' modifica della tabella
Next I
```

Nota – Le tabelle di testo sono disponibili in StarOffice 8 tramite l'elenco `TextTables` dell'oggetto documento, che prende il posto del precedente elenco di tabelle fornito nell'oggetto `Selection`. L'esempio precedente mostra come creare una tabella di testo. Le opzioni per accedere alle tabelle di testo sono descritte nella sezione seguente.

Modifica delle tabelle

Una tabella è formata da singole righe che a loro volta contengono diverse celle. In senso stretto, in StarOffice non sono presenti colonne di tabelle, che vengono prodotte in modo implicito disponendo le righe (una sotto l'altra) una di fianco all'altra. Per semplificare l'accesso alle tabelle, StarOffice fornisce tuttavia alcuni metodi che operano utilizzando le colonne e risultano di particolare utilità se nella tabella non si è eseguita l'unione di più celle.

Ma prima passiamo ad analizzare le proprietà della tabella stessa. Esse sono definite nel servizio `com.sun.star.text.TextTable`. Di seguito viene riportato un elenco delle più importanti proprietà dell'oggetto tabella:

- **BackColor (Long)** – colore di sfondo della tabella.
- **BottomMargin (Long)** – margine inferiore in centesimi di millimetro.
- **LeftMargin (Long)** – margine sinistro in centesimi di millimetro.
- **RightMargin (Long)** – margine destro in centesimi di millimetro.
- **TopMargin (Long)** – margine superiore in centesimi di millimetro.
- **RepeatHeadline (Boolean)** – l'intestazione della tabella viene ripetuta su ogni pagina.
- **Width (Long)** – larghezza assoluta della tabella in centesimi di millimetro.

Righe

Una tabella è formata da un elenco contenente una serie di righe. L'esempio seguente mostra come richiamare e formattare le righe di una tabella.

```
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
```

L'esempio crea prima un elenco contenente tutte le righe utilizzando la chiamata `Table.getRows`. I metodi `getCount` e `getByIndex` consentono l'ulteriore elaborazione dell'elenco e appartengono all'interfaccia `com.sun.star.table.XtableRows`. Il metodo `getByIndex` restituisce un oggetto riga, che supporta il servizio `com.sun.star.text.TextTableRow`.

Di seguito vengono riportati i metodi principali dell'interfaccia `com.sun.star.table.XtableRows`:

- **getByIndex(Integer)** – restituisce un oggetto riga per l'indice specificato.
- **getCount()** – restituisce il numero di oggetti riga.
- **insertByIndex(Index, Count)** – inserisce un numero di righe pari a Count nella tabella alla posizione di `Index`.
- **removeByIndex(Index, Count)** – elimina un numero di righe pari a Count dalla tabella alla posizione di `Index`.

Mentre i metodi `getByIndex` e `getCount` sono disponibili in tutte le tabelle, i metodi `insertByIndex` e `removeByIndex` si possono utilizzare solo nelle tabelle che non contengono celle precedentemente unite.

Il servizio `com.sun.star.text.TextTableRow` fornisce le proprietà seguenti:

- **BackColor (Long)** – colore di sfondo della riga.
- **Height (Long)** – altezza della riga in centesimi di millimetro.
- **IsAutoHeight (Boolean)** – l'altezza della tabella è adattata dinamicamente al suo contenuto.
- **VertOrient (const)** – orientamento verticale della cornice di testo – dettagli sull'orientamento verticale del testo all'interno della tabella (valori conformi a `com.sun.star.text.VertOrientation`)

Colonne

Per accedere alle colonne si procede allo stesso modo delle righe, utilizzando i metodi `getByIndex`, `getCount`, `insertByIndex` e `removeByIndex` sull'oggetto `Column`, raggiungibile tramite `getColumns`. Li potrete però utilizzare solo nelle tabelle che non contengono celle precedentemente unite. In StarOffice Basic, le celle non possono essere formattate per colonna. Per procedere ugualmente in tal senso, dovete avvalervi del metodo di formattazione delle singole celle della tabella.

Celle

Ogni cella di un documento di StarOffice ha un nome univoco. Se il cursore di StarOffice si trova in una cella, il nome della cella viene visualizzato nella barra di stato. La cella superiore sinistra è generalmente denominata A1, mentre quella inferiore destra è denominata `Xn`, dove `X` rappresenta le lettere della colonna superiore

e n i numeri dell'ultima riga. Gli oggetti cella sono disponibili tramite il metodo `getCellByName()` dell'oggetto tabella. L'esempio seguente mostra un ciclo che vaglia tutte le celle di una tabella e inserisce i numeri di riga e colonna corrispondenti nelle celle.

```
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
```

La cella di una tabella è paragonabile a un testo standard e supporta l'interfaccia `createTextCursor` per creare un oggetto `TextCursor` associato.

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

Tutte le opzioni di formattazione per i singoli caratteri e paragrafi sono quindi automaticamente disponibili.

L'esempio seguente esegue una ricerca in tutte le tabelle di un documento di testo e applica il formato allineato a destra a tutte le celle con valori numerici per messo della proprietà di paragrafo corrispondente.

```
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
```

L'esempio crea un elenco `TextTables` contenente tutte le tabelle di un testo attraversate nel ciclo. StarOffice crea quindi un elenco dei nomi delle celle associate per ognuna di queste tabelle, che vengono vagliate a turno nel ciclo. Se una cella contiene un valore numerico, l'esempio modifica la formattazione di conseguenza. Per procedere in tal senso, crea prima un oggetto `TextCursor` che fa riferimento al contenuto della cella della tabella, quindi adatta le proprietà del paragrafo alla cella.

Cornici di testo

Le cornici di testo sono considerate oggetti `TextContent`, proprio come le tabelle e i grafici e possono essere costituite essenzialmente da testo standard, sebbene siano collocabili in qualsiasi posizione della pagina e non vengano incluse nel flusso del testo.

Come con tutti gli oggetti `TextContent`, nelle cornici di testo si opera anche una distinzione tra creazione effettiva e inserimento nel documento.

```
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)
```

Per creare la cornice di testo, utilizzate il metodo `createInstance` dell'oggetto documento. La cornice di testo così creata può essere poi inserita nel documento utilizzando il metodo `insertTextContent` dell'oggetto `Text`. In questa operazione dovrete specificare il nome del servizio `com.sun.star.text.TextFrame` corretto.

La posizione di inserimento della cornice di testo è determinata da un oggetto `Cursor`, che viene inoltre eseguito all'inserimento.

Nota – Le cornici di testo sono il corrispondente in StarOffice della cornice di posizione utilizzata in Word. Mentre VBA utilizza il metodo `Document.Frames.Add` per questo scopo, la creazione in VBA viene eseguita utilizzando la suddetta procedura con l'ausilio di un `TextCursor` nonché del metodo `createInstance` dell'oggetto documento.

Gli oggetti cornice di testo forniscono una serie di proprietà con le quali influenzare la posizione e il comportamento della cornice. La maggioranza di queste proprietà sono definite nel servizio `com.sun.star.text.BaseFrameProperties`, che è supportato anche da ogni servizio `TextFrame`. Le proprietà principali sono le seguenti:

- **BackColor (Long)** – colore di sfondo della cornice di testo.
- **BottomMargin (Long)** – margine inferiore in centesimi di millimetro.
- **LeftMargin (Long)** – margine sinistro in centesimi di millimetro.
- **RightMargin (Long)** – margine destro in centesimi di millimetro.
- **TopMargin (Long)** – margine superiore in centesimi di millimetro.
- **Height (Long)** – altezza della cornice di testo in centesimi di millimetro.
- **Width (Long)** – larghezza della cornice di testo in centesimi di millimetro.
- **HoriOrient (const)** – orientamento orizzontale della cornice di testo (conforme a `com.sun.star.text.HoriOrientation`).
- **VertOrient (const)** – orientamento verticale della cornice di testo (conforme a `com.sun.star.text.VertOrientation`).

L'esempio seguente crea una cornice di testo utilizzando le proprietà descritte in precedenza:

```
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)
```

L'esempio crea un `TextCursor` come segno di inserimento per la cornice di testo, posizionato tra la prima e la seconda parola del testo. La cornice di testo è creata utilizzando `Doc.createInstance`. Le proprietà degli oggetti cornice di testo sono impostate sui valori iniziali richiesti.

Si noti l'interazione tra le proprietà `AnchorType` (dal servizio `TextContent`) e `VertOrient` (dal servizio `BaseFrameProperties`): `AnchorType` riceve il valore `AS_CHARACTER`. La cornice viene pertanto inserita direttamente nel flusso del testo e si comporta come un carattere. Può, ad esempio, essere spostata nella riga successiva se si verifica un a capo. Il valore `LINE_TOP` della proprietà `VertOrient` assicura che il bordo superiore della cornice di testo sia alla stessa altezza del bordo superiore del carattere.

Una volta completata l'inizializzazione, la cornice di testo viene infine inserita nel documento di testo utilizzando una chiamata da `insertTextContent`.

Per modificare il contenuto di una cornice di testo, l'utente si avvale del `TextCursor`, già menzionato numerose volte e disponibile anche per le cornici di testo.

```
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 = "Questa è una piccola prova!"
```

L'esempio crea una cornice di testo, la inserisce nel documento corrente e apre un `TextCursor` per la cornice di testo. Questo cursore viene impiegato per impostare il carattere della cornice sul grassetto e per impostare l'orientamento del paragrafo su centrato. La cornice di testo viene infine assegnata alla stringa "Questa è una piccola prova.".

Campi di testo

I campi di testo sono oggetti `TextContent` perché forniscono una logica addizionale che si estende oltre il puro testo. Si possono inserire in un documento di testo utilizzando gli stessi metodi impiegati per gli altri oggetti `TextContent`:

```
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)
```

L'esempio inserisce un campo di testo con la data corrente all'inizio del documento di testo corrente. Il valore `True` della proprietà `IsDate` produce la visualizzazione della sola data e non dell'ora. Il valore `False` di `IsFixed` garantisce che la data venga automaticamente aggiornata all'apertura del documento.

Nota – Mentre il tipo di un campo è specificato in VBA da un parametro del metodo `Document.Fields.Add`, in StarOffice Basic è il nome del servizio responsabile del tipo di campo in questione a definirlo.

In passato, si accedeva ai campi di testo utilizzando una serie completa di metodi resi disponibili da StarOffice nel vecchio oggetto `Selection` (ad esempio `InsertField`, `DeleteUserField`, `SetCurField`).

In StarOffice 8, i campi vengono invece gestiti utilizzando un concetto orientato agli oggetti. Per creare un campo di testo, dovete prima creare un campo di testo del tipo richiesto e quindi inizializzarlo utilizzando le proprietà necessarie. Il campo di testo viene poi inserito nel documento con il metodo `insertTextContent`. Nell'esempio precedente è illustrato un testo sorgente corrispondente. I tipi di campi più importanti e le relative proprietà sono descritti nelle sezioni di seguito.

Oltre all'inserimento dei campi di testo, anche la ricerca dei campi in un documento può costituire un'attività importante. L'esempio seguente mostra come vagliare in un ciclo tutti i campi di testo di un documento di testo e controllarne il tipo di pertinenza.

```
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 "Data/ora"
   ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
      MsgBox "Annotazione"
   Else
      MsgBox "sconosciuto"
   End If

Wend
```

Il punto iniziale per determinare i campi di testo presenti è l'elenco `TextFields` dell'oggetto documento. L'esempio crea un oggetto `Enumeration` sulla base di questo elenco, con il quale vagliare uno per volta tutti i campi di testo in un ciclo. I campi di testo individuati vengono controllati per verificare il servizio supportato utilizzando il metodo `supportsService`. Se il campo risulta essere del tipo per data/ora o annotazioni, il tipo di campo corrispondente viene visualizzato in una casella di informazioni. Se, d'altro canto, l'esempio incontra un altro campo, visualizza l'informazione "sconosciuto".

Di seguito viene riportato un elenco dei campi di testo più importanti e delle relative proprietà associate. Un elenco completo di tutti i campi di testo è fornito nel riferimento della API nel modulo `com.sun.star.text.TextField`. (Per l'elencazione dei nomi di servizi di un campo di testo, utilizzate in StarOffice Basic caratteri in maiuscolo e in minuscolo, come nell'esempio precedente).

Numeri di pagine, parole e caratteri

I campi di testo

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

restituiscono il numero di pagine, parole e caratteri di un testo e supportano la proprietà seguente:

- **NumberingType (const)** – formato di numerazione (regole conformi alle costanti di `com.sun.star.style.NumberingType`).

Pagina corrente

Il numero della pagina corrente può essere inserito nel documento utilizzando il campo di testo `com.sun.star.text.TextField.PageNumber`. Si possono specificare le proprietà seguenti:

- **NumberingType (const)** – formato numerico (regole conformi alle costanti di `com.sun.star.style.NumberingType`).
- **Offset (short)** – correzione aggiunta al numero di pagine (è possibile anche la specifica negativa).

L'esempio seguente mostra come inserire il numero di pagine nel piè di pagina di un documento.

```
Dim Doc As Object
Dim DateTimeField As Object
Dim PageStyles As Object
Dim StdPage As Object
Dim FooterCursor As Object
Dim PageNumber As Object

Doc = StarDesktop.CurrentComponent

PageNumber = Doc.createInstance("com.sun.star.text.TextField.PageNumber")
PageNumber.NumberingType = com.sun.star.style.NumberingType.ARABIC

PageStyles = Doc.StyleFamilies.getByName("PageStyles")

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

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

L'esempio crea prima un campo di testo che supporta il servizio `com.sun.star.text.TextField.PageNumber`. Poiché le righe di intestazione e piè di pagina sono definite come parte dei modelli di pagina di StarOffice, la determinazione iniziale viene effettuata utilizzando l'elenco di tutti i `PageStyles`.

Per garantire che la riga del piè di pagina sia visibile, la proprietà `FooterIsOn` è impostata su `True`. Il campo di testo viene quindi inserito nel documento utilizzando l'oggetto testo associato della riga del piè di pagina di sinistra.

Annotazioni

I campi di annotazioni (`com.sun.star.text.TextField.Annotation`) vengono visualizzati per mezzo di un simbolino giallo nel testo. Facendo clic su questo simbolo si apre un campo di testo, in cui potrete registrare un commento sul punto corrente nel testo. Un campo di annotazione possiede le seguenti proprietà:

- **Author (String)** – nome dell'autore.
- **Content (String)** – testo di commento.
- **Date (Date)** – data in cui è scritta l'annotazione.

Data/ora

Un campo per data/ora (`com.sun.star.text.TextField.DateTime`) rappresenta la data o l'ora correnti e supporta le proprietà seguenti:

- **IsFixed (Boolean)** – se `True`, i dettagli dell'ora di inserimento rimangono invariati, se `False`, vengono aggiornati ad ogni apertura del documento.
- **IsDate (Boolean)** – se `True`, il campo visualizza la data corrente, altrimenti l'ora corrente.
- **DateTimeValue (struct)** – contenuto corrente del campo (struttura `com.sun.star.util.DateTime`)
- **NumberFormat (const)** – formato di raffigurazione dell'ora o della data.

Nome/numero del capitolo

Il nome del capitolo corrente è disponibile tramite un campo di testo del tipo `com.sun.star.text.TextField.Chapter`. La forma può essere definita utilizzando le due proprietà seguenti:

- **ChapterFormat (const)** – determina se è riportato il nome o il numero del capitolo (conformemente a `com.sun.star.text.ChapterFormat`)
- **Level (Integer)** – determina il livello del capitolo per il quale visualizzare nome e/o numero. Il valore 0 rappresenta il livello più alto disponibile.

Segnalibri

I segnalibri (servizio `com.sun.star.text.Bookmark`) sono oggetti `TextContent`. I segnalibri vengono creati e inseriti utilizzando il concetto seguente, già descritto in precedenza:

```
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 = "Miei segnalibri"
Doc.Text.insertTextContent(Cursor, Bookmark, True)
```

L'esempio crea un `Cursor`, che contrassegna la posizione di inserimento del segnalibro e quindi l'effettivo oggetto segnalibro (`Bookmark`). Al segnalibro viene quindi assegnato un nome ed è inserito nel documento tramite `insertTextContent`, in corrispondenza della posizione del cursore.

È possibile accedere ai segnalibri di un testo tramite l'elenco denominato `Bookmarks`. Inoltre, potete accedere ai segnalibri per numero o per nome.

L'esempio seguente mostra come individuare un segnalibro all'interno del testo e inserire un testo in quella posizione.

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

Doc = StarDesktop.CurrentComponent

Bookmark = Doc.Bookmarks.getByName("Miei segnalibri")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Ecco il segnalibro"
```

In questo esempio, il metodo `getByName` è utilizzato per reperire il segnalibro richiesto tramite il nome. La chiamata `createTextCursorByRange` crea allora un `Cursor`, che viene posizionato nella posizione di ancoraggio del segnalibro. Il cursore quindi inserisce il testo richiesto in questo punto.

CAPITOLO 7

Fogli elettronici

StarOffice Basic fornisce un'interfaccia estesa per la creazione e la modifica dei fogli elettronici controllate da programma. Questo capitolo descrive come controllare i servizi, i metodi e le proprietà pertinenti dei fogli elettronici.

La prima sezione tratta la struttura di base dei fogli elettronici e mostra come accedere e modificare i contenuti delle singole celle.

La seconda descrive invece su come modificare i fogli elettronici in modo efficiente, concentrandosi sulle aree di celle e sulle opzioni di ricerca e sostituzione dei contenuti delle celle.

Nota – L'oggetto `Range`, che permette di trattare qualsiasi area delle tabelle, è stato ampliato nella nuova API.

Struttura dei documenti basati su tabelle (fogli elettronici)

L'oggetto documento di un foglio elettronico si basa sul servizio `com.sun.star.sheet.SpreadsheetDocument`. Ciascuno dei documenti può contenere diversi fogli elettronici. Nel presente manuale, per *documento basato su tabelle* o *foglio elettronico* si intende l'intero documento, mentre per *tabella* si intende una singola tabella del documento.

Nota – VBA e StarOffice Basic utilizzando una terminologia diversa per i fogli elettronici e il loro contenuto. In VBA l'oggetto documento è denominato *Workbook* (cartella di lavoro), mentre le singole pagine sono *Worksheet* (fogli di lavoro), mentre i corrispondenti di StarOffice Basic sono rispettivamente *foglio elettronico* e *tabella*.

Fogli elettronici

Si accede ai singoli fogli di un foglio elettronico tramite l'elenco `Sheets`.

Gli esempi seguenti mostrano come accedere a un foglio tramite il numero o il nome.

Esempio 1: accesso tramite il numero (la numerazione inizia da 0)

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet  = Doc. Sheets (0)
```

Esempio 2: accesso tramite il nome

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Foglio 1")
```

Nel primo esempio, si accede al foglio mediante il suo numero (il conteggio inizia da 0). Nel secondo esempio, si accede al foglio utilizzando il suo nome e il metodo `getByName`.

L'oggetto `Sheet` ottenuto dal metodo `getByName` supporta il servizio `com.sun.star.sheet.Spreadsheet`. Oltre a fornire diverse interfacce per la modifica dei contenuti, questo servizio fornisce le proprietà seguenti:

- **IsVisible (Boolean)** – il foglio elettronico è visibile.
- **PageStyle (String)** – il nome del modello di pagina per il foglio elettronico.

Creazione, eliminazione e ridenominazione dei fogli

L'elenco `Sheets` di un documento `spreadsheet` viene utilizzato anche per creare, eliminare e rinominare le singole tabelle. L'esempio seguente si avvale del metodo `hasByName` per controllare se esiste una tabella denominata *Tabella*. In caso affermativo, il metodo determina un riferimento dell'oggetto corrispondente utilizzando il metodo `getByName`, quindi salva il riferimento in una variabile in `Sheet`. Se invece la tabella corrispondente non esiste, viene creata dalla chiamata `createInstance` e inserita nel foglio elettronico dal metodo `insertByName`.

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

If Doc.Sheets.hasByName("Tabella") Then
   Sheet = Doc.Sheets.getByName("Tabella")
Else
   Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
   Doc.Sheets.insertByName("Tabella", Sheet)
End If
```

I metodi `getByName` e `insertByName` derivano dall'interfaccia `com.sun.star.container.XnameContainer`, come descritto nel Capitolo 4.

Righe e colonne

Ogni foglio contiene un elenco delle righe e delle colonne, che sono disponibili tramite le proprietà `Rows` e `Columns` dell'oggetto foglio elettronico e supporta i servizi `com.sun.star.table.TableColumns` e/o `com.sun.star.table.TableRows`.

L'esempio seguente crea due oggetti che fanno riferimento alla prima riga e alla prima colonna di un foglio e memorizza i riferimenti nelle variabili e oggetto `FirstCol` e `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)
```

Gli oggetti colonna supportano il servizio `com.sun.star.table.TableColumn`, che dispone delle seguenti proprietà:

- **Width (long)** – larghezza di una colonna in centesimi di millimetro.
- **OptimalWidth (Boolean)** – imposta una colonna sulla sua larghezza ottimale.
- **IsVisible (Boolean)** – visualizza una colonna.
- **IsStartOfNewPage (Boolean)** – nella stampa, crea un'interruzione di pagina prima di una colonna.

La larghezza di una colonna viene ottimizzata solo quando la proprietà `OptimalWidth` è impostata su `True`. Se viene modifica la larghezza di una singola cella, la larghezza della colonna che contiene la cella rimane invariata. In termini di funzionalità, `OptimalWidth` è più un metodo che una proprietà.

Gli oggetti riga sono basati sul servizio `com.sun.star.table.RowColumn`, che dispone delle seguenti proprietà:

- **Height (Long)** – altezza della riga in centesimi di millimetro.
- **OptimalHeight (Boolean)** – imposta una colonna sulla sua altezza ottimale.
- **IsVisible (Boolean)** – visualizza la riga.
- **IsStartOfNewPage (Boolean)** – nella stampa, crea un'interruzione di pagina prima della riga.

Se la proprietà `OptimalHeight` di una riga è impostata su `True`, l'altezza della riga viene modificata automaticamente quando si varia l'altezza di una cella della riga. L'ottimizzazione automatica prosegue finché alla riga non è assegnata un l'altezza assoluta mediante la proprietà `Height`.

L'esempio seguente attiva l'ottimizzazione automatica dell'altezza per le prime cinque righe del foglio e rende invisibile la seconda colonna.

```
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
```

Nota – In StarOffice Basic, si accede agli elenchi `Rows` e `Columns` tramite un indice. A differenza di VBA, la prima colonna ha indice 0 e non indice 1.

Inserimento ed eliminazione di righe o colonne

Gli oggetti `Rows` e `Columns` di un foglio possono accedere alle righe e alle colonne già esistenti nonché inserirle o modificarle.

```
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)
```

Questo esempio utilizza il metodo `insertByIndex` per inserire una nuova colonna nella posizione della quarta colonna nel foglio (indice 3 – la numerazione inizia da 0). Il secondo parametro specifica il numero di colonne da inserire (in questo esempio: uno).

Il metodo `removeByIndex` elimina la sesta colonna (indice 5). Ancora una volta, il secondo parametro specifica il numero di colonne da eliminare.

I metodi di inserimento ed eliminazione delle righe utilizzando la funzione dell'oggetto `Rows` nello stesso modo dei metodi illustrati per la modifica delle colonne utilizzando l'oggetto `Columns`.

Celle

Un foglio elettronico è formato da un elenco bidimensionale contenente le celle. Ogni cella è definita dalla sua posizione X e Y rispetto alla cella superiore sinistra che ha la posizione (0,0).

L'esempio seguente crea un oggetto che fa riferimento alla cella superiore sinistra e inserisce un testo nella cella:

```
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"
```

Oltre alle coordinate numeriche, a ogni cella di un foglio è assegnato un nome, ad esempio, la cella superiore sinistra (0,0) di un foglio elettronico è denominata `A1`. La lettera `A` rappresenta la colonna e il numero 1 la riga. È importante che il *nome* e la *posizione* di una cella non vengano confusi perché il conteggio delle righe per nomi inizia con 1 ma il conteggio per posizione inizia con 0.

In StarOffice, una cella di tabella può essere vuota o contenere testo, numeri o formule. Il tipo di cella non è determinato dal contenuto salvato nella cella, ma piuttosto dalla proprietà dell'oggetto utilizzata per la sua immissione. I numeri possono essere inseriti e richiamati con la proprietà `Value`, il testo con la proprietà `String` e le formule con la proprietà `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"
```

L'esempio inserisce un numero, un testo e una formula nei campi da A1 ad A3.

Nota – Le proprietà `Value`, `String` e `Formula` sostituiscono il metodo `PutCell` per l'impostazione dei valori di una cella di tabella.

StarOffice tratta come testo il contenuto delle celle inserito utilizzando la proprietà `String`, anche se il contenuto è un numero. I numeri inseriti sono allineati a sinistra all'interno della cella invece di essere allineati a destra. Si noti inoltre la differenza tra testo e numeri quando si utilizzano le formule:

```
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
```

Sebbene la cella A1 contenga il valore 100 e la cella A2 il valore 1000, la formula A1+A2 restituisce il valore 100, perché i contenuti della cella A2 erano stati immessi come stringa e non come numero.

Per controllare se i contenuti di una cella contengono un numero o una stringa, utilizzate la proprietà `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 "Contenuto: vuoto"
Case com.sun.star.table.CellContentType.VALUE
   MsgBox "Contenuto: valore"
Case com.sun.star.table.CellContentType.TEXT
   MsgBox "Contenuto: testo"
Case com.sun.star.table.CellContentType.FORMULA
   MsgBox "Contenuto: formula"
End Select
```

La proprietà `Cell.Type` restituisce un valore per l'enumerazione `com.sun.star.table.CellContentType` che identifica il tipo di contenuti di una cella. Sono ammessi i seguenti valori:

- **EMPTY** – nessun valore
- **VALUE** – numero
- **STRING** - stringhe
- **FORMULA** - formula

Inserimento, eliminazione, copia e spostamento delle celle

Oltre a modificare direttamente il contenuto delle celle, StarOffice Calc fornisce anche un'interfaccia che permette di inserire, eliminare, copiare o unire le celle. L'interfaccia (`com.sun.star.sheet.XRangeMovement`) è disponibile tramite l'oggetto foglio elettronico e fornisce quattro metodi per modificare il contenuto delle celle.

Il metodo `insertCell` è utilizzato per inserire le celle in un foglio.

```
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)
```

Questo esempio inserisce un'area di celle di dimensioni pari a due righe per due colonne nella seconda colonna e riga (ciascuna reca il numero 1) del primo foglio (numero 0) nel foglio elettronico. I valori già esistenti nell'area di celle specificata vengono quindi spostati al di sotto di essa.

Per definire l'area di celle da inserire, utilizzate la struttura `com.sun.star.table.CellRangeAddress`. I valori seguenti sono inclusi in questa struttura:

- **Sheet (short)** – numero del foglio (la numerazione inizia da 0).
- **StartColumn (long)** – la prima colonna nell'area di celle (la numerazione inizia da 0).
- **StartRow (long)** – la prima riga nell'area di celle (la numerazione inizia da 0).
- **EndColumn (long)** – la colonna finale nell'area di celle (la numerazione inizia da 0).
- **EndRow (long)** – la riga finale nell'area di celle (la numerazione inizia da 0).

La struttura `CellRangeAddress` completata deve essere trasmessa come primo parametro al metodo `insertCells`. Il secondo parametro di `insertCells` contiene un valore dell'enumerazione `com.sun.star.sheet.CellInsertMode` e definisce cosa fare con i valori situati davanti al punto di inserimento. L'enumerazione `CellInsertMode` riconosce i valori seguenti:

- **NONE** – i valori correnti rimangono nella loro attuale posizione.
- **DOWN** – le celle in corrispondenza della posizione di inserimento e sotto di esso sono spostate verso il basso.
- **RIGHT** – le celle in corrispondenza della posizione di inserimento e alla sua destra vengono spostate verso destra.
- **ROWS** – le righe dopo la posizione di inserimento vengono spostate verso il basso.
- **COLUMNS** – le colonne dopo la posizione di inserimento vengono spostate verso destra.

Il metodo `removeRange` è la controparte del metodo `insertCells`. Elimina l'area definita nella struttura `CellRangeAddress` dalla tabella.

```
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)
```

Questo esempio rimuove l'area di celle `B2:C3` dalla tabella, quindi sposta verso l'alto le celle sottostanti di due righe. Il tipo di rimozione è definito da uno dei valori seguenti dall'enumerazione `com.sun.star.sheet.CellDeleteMode`:

- **NONE** – i valori correnti rimangono nella loro attuale posizione.

- **UP** – le celle in corrispondenza della posizione di inserimento e sotto di essa vengono spostate verso l'alto.
- **LEFT** – le celle in corrispondenza della posizione di inserimento e alla sua destra vengono spostate verso sinistra.
- **ROWS** – le righe dopo la posizione di inserimento vengono spostate verso l'alto.
- **COLUMNS** – le colonne dopo la posizione di inserimento vengono spostate verso sinistra.

L'interfaccia `XRangeMovement` offre due ulteriori metodi per spostare (`moveRange`) o copiare (`copyRange`) le aree di celle. L'esempio seguente sposta l'area `B2:C3` in modo da farla iniziare alla posizione `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)
```

Oltre alla struttura `CellRangeAdress`, il metodo `moveRange` prevede una struttura `com.sun.star.table.CellAddress` per definire l'origine della regione di destinazione dello spostamento. Il metodo `CellAddress` fornisce i seguenti valori:

- **Sheet (short)** – numero del foglio elettronico (la numerazione inizia da 0).
- **Column (long)** – numero della colonna identificata (la numerazione inizia da 0).
- **Row (long)** – numero della riga identificata (la numerazione inizia da 0).

I contenuti della cella nell'area di destinazione vengono sempre sovrascritti dal metodo `moveRange`. A differenza del metodo `InsertCells`, nel metodo `removeRange` non è fornito un parametro per l'esecuzione degli spostamenti automatici.

Il metodo `copyRange` funziona nello stesso modo del metodo `moveRange`, eccetto che `copyRange` inserisce una copia dell'area di celle invece di spostarle.

Nota – In termini di funzionalità, i metodi StarOffice Basic StarOffice Basic `insertCell`, `removeRange` e `copyRange` sono paragonabili ai metodi `Range.Insert`, `Range.Delete` e `Range.Copy` di VBA. In VBA, però, i metodi sono applicati all'oggetto `Range` corrispondente, mentre in StarOffice Basic sono applicati all'oggetto `Sheet` associato.

Formattazione

Un foglio elettronico dispone di proprietà e metodi per formattare celle e pagine.

Proprietà delle celle

Esistono numerose opzioni per formattare le celle, come specificare tipo e dimensioni del carattere del testo. Ogni cella supporta i servizi `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`, le cui proprietà principali sono descritte nel Capitolo 6. La formattazione speciale delle celle è gestita dal servizio `com.sun.star.table.CellProperties`, le cui proprietà principali sono descritte nelle sezioni seguenti.

Tutte le proprietà si possono applicare sia alle singole celle che ad aree di celle.

Nota – L'oggetto `CellProperties` della API StarOffice è paragonabile all'oggetto `Interior` di VBA, che definisce anche le proprietà specifiche delle celle.

Colore di sfondo e ombre

Il servizio `com.sun.star.table.CellProperties` fornisce le seguenti proprietà per definire i colori di sfondo e le ombre:

- **CellBackColor (Long)** – colore di sfondo della cella della tabella.
- **IsCellBackgroundTransparent (Boolean)** – imposta su trasparente il colore di sfondo.
- **ShadowFormat (struct)** – specifica l'ombra per le celle (struttura conforme a `com.sun.star.table.ShadowFormat`).

La struttura `com.sun.star.table.ShadowFormat` e le specifiche dettagliate per le ombre delle celle hanno la struttura seguente:

- **Location (enum)** – posizione dell'ombra (valore della struttura `com.sun.star.table.ShadowLocation`).

- **ShadowWidth (Short)** – dimensioni dell'ombra in centesimi di millimetro.
- **IsTransparent (Boolean)** – imposta l'ombra su trasparente.
- **Color (Long)** – colore dell'ombra.

L'esempio seguente scrive il numero 1000 nella cella B2, cambia in rosso il colore dello sfondo avvalendosi della proprietà `CellBackColor`, quindi crea per la cella un'ombra di colore grigio chiaro, spostata di 1 mm verso sinistra e verso il basso.

```
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
```

Giustificazione

StarOffice offre varie funzioni che permettono di modificare la giustificazione del testo nella cella di una tabella.

Le proprietà seguenti definiscono la giustificazione orizzontale e verticale di un testo:

- **HoriJustify (enum)** - giustificazione orizzontale del testo (valore da `com.sun.star.table.CellHoriJustify`)
- **VertJustify (enum)** - giustificazione verticale del testo (valore da `com.sun.star.table.CellVertJustify`)
- **Orientation (enum)** – orientamento del testo (valore conforme a `com.sun.star.table.CellOrientation`)
- **IsTextWrapped (Boolean)** - consente interruzioni di riga automatiche all'interno della cella
- **RotateAngle (Long)** – angolo di rotazione del testo in centesimi di grado

L'esempio seguente mostra come "impilare" i contenuti di una cella in modo che i singoli caratteri vengano stampati uno sotto l'altro nell'angolo superiore sinistro della cella. I caratteri non vengono ruotati.

```
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
```

Formati di numeri, data e testo

StarOffice fornisce una serie completa di formati predefiniti di data e ora. Ciascuno di questi formati ha un numero interno utilizzato per assegnare il formato alle celle con la proprietà `NumberFormat`. StarOffice dispone dei metodi `queryKey` e `addNew` con cui accedere ai formati numerici esistenti nonché crearne di propri. I metodi sono accessibili tramite la seguente chiamata:

```
NumberFormats = Doc.NumberFormats
```

Il formato viene specificato utilizzando una stringa di formattazione strutturata in modo analogo alla funzione di formattazione di StarOffice Basic. Va tuttavia segnalata un'importante differenza: mentre quest'ultima richiede l'uso delle abbreviazioni inglesi e del punto decimale o dei caratteri come separatori delle migliaia, per la struttura di un comando di formattazione per l'oggetto `NumberFormats` dovete utilizzare le abbreviazioni specifiche del paese.

L'esempio seguente formatta la cella B2 in modo che i numeri vengano visualizzati con tre posizioni decimali e utilizzino le virgole come separatore delle migliaia.

```
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
```

La finestra di dialogo **Formatta celle** di StarOffice Calc offre una panoramica delle diverse opzioni di formattazione per le celle.

Proprietà delle pagine

Le proprietà delle pagine sono le opzioni di formattazione che posizionano il contenuto del documento su una pagina, nonché gli elementi visivi che vengono ripetuti pagina dopo pagina. Esse includono:

- Formati dei fogli
- Margini
- Intestazioni e piè di pagina.

La procedura per definire i formati della carta differisce da quella di altre modalità di formattazione. Infatti, mentre gli elementi di celle, paragrafi e caratteri possono essere immessi direttamente, il formato va definito e applicato in modo indiretto utilizzando i modelli di pagina. Intestazioni e piè di pagina, ad esempio, vengono aggiungi al modello di pagina.

Le sezioni seguenti descrivono le principali opzioni di formattazione per le pagine dei fogli elettronici. Molti degli stili descritti sono disponibili anche per i documenti di testo. Le proprietà della pagina valide per entrambe i tipi di documenti sono definite nel servizio `com.sun.star.style.PageProperties`. Le proprietà della pagina che si applicano invece solo ai fogli elettronici sono definite nel servizio `com.sun.star.sheet.TablePageStyle`.

Nota – Le proprietà della pagina (margini, bordi e così via) per i documenti Microsoft Office sono definite per mezzo di un oggetto `PageSetup` a livello di oggetto `Worksheet` (Excel) o `Document` (Word). In StarOffice, queste proprietà vengono invece definite usando un modello di pagina, che è a sua volta collegato al documento associato.

Sfondo pagina

Il servizio `com.sun.star.style.PageProperties` definisce le seguenti proprietà dello sfondo delle pagine:

- **BackColor (long)** – colore di sfondo

- **BackGraphicURL (String)** – URL dell'immagine di sfondo da usare
- **BackGraphicFilter (String)** – nome del filtro che interpreta le immagini di sfondo
- **BackGraphicLocation (Enum)** – posizione delle immagini di sfondo (valore conforme all'enumerazione)
- **BackTransparent (Boolean)** - rende trasparente lo sfondo

Formato della pagina

Il formato della pagina viene definito utilizzando le seguenti proprietà del servizio `com.sun.star.style.PageProperties`:

- **IsLandscape (Boolean)** – formato orizzontale
- **Width (long)** – larghezza della pagina in centesimi di millimetro
- **Height (long)** – altezza della pagina in centesimi di millimetro
- **PrinterPaperTray (String)** – nome del cassetto della stampante da usare

L'esempio seguente imposta le dimensioni della pagina del modello di pagina "Default" sul formato orizzontale DIN A5 (altezza 14,8 cm, larghezza 21 cm):

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.IsLandscape = True
DefPage.Width = 21000
DefPage.Height = 14800
```

Margine, bordo e ombra della pagina

Il servizio `com.sun.star.style.PageProperties` fornisce le seguenti proprietà per regolare i margini della pagina, i bordi e le ombre:

- **LeftMargin (long)** – larghezza del margine sinistro della pagina in centesimi di millimetro
- **RightMargin (long)** – larghezza del margine destro della pagina in centesimi di millimetro
- **TopMargin (long)** – larghezza del margine superiore della pagina in centesimi di millimetro
- **BottomMargin (long)** – larghezza del margine inferiore della pagina in centesimi di millimetro

- **LeftBorder (struct)** – specifiche per il bordo della linea sinistra della pagina (struttura `com.sun.star.table.BorderLine`)
- **RightBorder (struct)** – specifiche per il bordo della linea destra della pagina (struttura `com.sun.star.table.BorderLine`)
- **TopBorder (struct)** – specifiche per il bordo della linea superiore della pagina (struttura `com.sun.star.table.BorderLine`)
- **BottomBorder (struct)** – specifiche per il bordo della linea inferiore della pagina (struttura `com.sun.star.table.BorderLine`)
- **LeftBorderDistance (long)** – distanza tra il bordo sinistro della pagina e i suoi contenuti in centesimi di millimetro
- **RightBorderDistance (long)** – distanza tra il bordo destro della pagina e i suoi contenuti in centesimi di millimetro
- **TopBorderDistance (long)** – distanza tra il bordo superiore della pagina e i suoi contenuti in centesimi di millimetro
- **BottomBorderDistance (long)** – distanza tra il bordo inferiore della pagina e i suoi contenuti in centesimi di millimetro
- **ShadowFormat (struct)** – specifiche per l'ombra dell'area dei contenuti di una pagina (struttura `com.sun.star.table.ShadowFormat`)

L'esempio seguente imposta i bordi sinistro e destro del modello di pagina "Default" su 1 centimetro.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.LeftMargin = 1000
DefPage.RightMargin = 1000
```

Intestazioni e piè di pagina

Le intestazioni e i piè di pagina di un documento fanno parte delle proprietà della pagina e vengono definiti mediante il servizio `com.sun.star.style.PageProperties`. Le proprietà per la formattazione delle intestazioni sono le seguenti:

- **HeaderIsOn (Boolean)** – l'intestazione viene attivata
- **HeaderLeftMargin (long)** – distanza tra intestazione e margine sinistro della pagina, espressa in centesimi di millimetro

- **HeaderRightMargin (long)** – distanza tra intestazione e margine destro della pagina, espressa in centesimi di millimetro
- **HeaderBodyDistance (long)** – distanza tra intestazione e corpo principale del documento, espressa in centesimi di millimetro
- **HeaderHeight (long)** – altezza dell'intestazione in centesimi di millimetro
- **HeaderIsDynamicHeight (Boolean)** – l'altezza dell'intestazione viene automaticamente adattata al contenuto
- **HeaderLeftBorder (struct)** – dettagli del bordo sinistro della cornice attorno all'intestazione (struttura `com.sun.star.table.BorderLine`)
- **HeaderRightBorder (struct)** – dettagli del bordo destro della cornice attorno all'intestazione (struttura `com.sun.star.table.BorderLine`)
- **HeaderTopBorder (struct)** – dettagli del bordo superiore del bordo attorno all'intestazione (struttura `com.sun.star.table.BorderLine`)
- **HeaderBottomBorder (struct)** – dettagli del bordo inferiore del bordo attorno all'intestazione (struttura `com.sun.star.table.BorderLine`)
- **HeaderLeftBorderDistance (long)** – distanza tra il bordo sinistro e il contenuto dell'intestazione, espressa in centesimi di millimetro
- **HeaderRightBorderDistance (long)** – distanza tra il bordo destro e il contenuto dell'intestazione, espressa in centesimi di millimetro
- **HeaderTopBorderDistance (long)** – distanza tra il bordo superiore e il contenuto dell'intestazione, espressa in centesimi di millimetro
- **HeaderBottomBorderDistance (long)** – distanza tra il bordo inferiore e il contenuto dell'intestazione, espressa in centesimi di millimetro
- **HeaderIsShared (Boolean)** – le intestazioni sulle pagine pari e quelle dispari hanno lo stesso contenuto (vedere `HeaderText` , `HeaderTextLeft` e `HeaderTextRight`)
- **HeaderBackColor (long)** – colore di sfondo dell'intestazione
- **HeaderBackGraphicURL (String)** – URL dell'immagine di sfondo da usare
- **HeaderBackGraphicFilter (String)** – nome del filtro che interpreta le immagini di sfondo per l'intestazione
- **HeaderBackGraphicLocation (Enum)** – posizione delle immagini di sfondo per l'intestazione (valore conforme all'enumerazione `com.sun.star.style.GraphicLocation`)
- **HeaderBackTransparent (Boolean)** – mostra come trasparente lo sfondo dell'intestazione
- **HeaderShadowFormat (struct)** – dettagli dell'ombra dell'intestazione (struttura `com.sun.star.table.ShadowFormat`)

Le proprietà per la formattazione dei piè di pagina sono le seguenti:

- **FooterIsOn (Boolean)** – il piè di pagina viene attivato
- **FooterLeftMargin (long)** – distanza tra piè di pagina e margine sinistro della pagina, espressa in centesimi di millimetro

- **FooterRightMargin (long)** – distanza tra piè di pagina e margine destro della pagina, espressa in centesimi di millimetro
- **FooterBodyDistance (long)** – distanza tra piè di pagina e corpo principale del documento, espressa in centesimi di millimetro
- **FooterHeight (long)** – altezza del piè di pagina in centesimi di millimetro
- **FooterIsDynamicHeight (Boolean)** – l'altezza del piè di pagina viene automaticamente adattata al contenuto
- **FooterLeftBorder (struct)** – dettagli del bordo sinistro della cornice attorno al piè di pagina (struttura `com.sun.star.table.BorderLine`)
- **FooterRightBorder (struct)** – dettagli del bordo destro della cornice attorno al piè di pagina (struttura `com.sun.star.table.BorderLine`)
- **FooterTopBorder (struct)** – dettagli del bordo superiore della cornice attorno al piè di pagina (struttura `com.sun.star.table.BorderLine`)
- **FooterBottomBorder (struct)** – dettagli del bordo inferiore della cornice attorno al piè di pagina (struttura `com.sun.star.table.BorderLine`)
- **FooterLeftBorderDistance (long)** – distanza tra il bordo sinistro e il contenuto del piè di pagina, espressa in centesimi di millimetro
- **FooterRightBorderDistance (long)** – distanza tra il bordo destro e il contenuto del piè di pagina, espressa in centesimi di millimetro
- **FooterTopBorderDistance (long)** – distanza tra il bordo superiore e il contenuto del piè di pagina, espressa in centesimi di millimetro
- **FooterBottomBorderDistance (long)** – distanza tra il bordo inferiore e il contenuto del piè di pagina, espressa in centesimi di millimetro
- **FooterIsShared (Boolean)** – le intestazioni sulle pagine pari e quelle dispari hanno lo stesso contenuto (vedere `FooterText`, `FooterTextLeft` e `FooterTextRight`)
- **FooterBackColor (long)** – colore di sfondo del piè di pagina
- **FooterBackGraphicURL (String)** – URL dell'immagine di sfondo da usare
- **FooterBackGraphicFilter (String)** – nome del filtro che interpreta le immagini di sfondo per il piè di pagina
- **FooterBackGraphicLocation (Enum)** – posizione delle immagini di sfondo per il piè di pagina (valore conforme all'enumerazione `com.sun.star.style.GraphicLocation`)
- **FooterBackTransparent (Boolean)** – mostra come trasparente lo sfondo del piè di pagina
- **FooterShadowFormat (struct)** – dettagli dell'ombra del piè di pagina (struttura `com.sun.star.table.ShadowFormat`)

Modifica del testo di intestazioni e piè di pagina

Si accede al contenuto di intestazioni e piè di pagina di un foglio elettronico per mezzo delle seguenti proprietà:

- **LeftPageHeaderContent (Object)** – contenuto delle intestazioni delle pagine pari (servizio `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageHeaderContent (Object)** – contenuto delle intestazioni delle pagine dispari (servizio `com.sun.star.sheet.HeaderFooterContent`)
- **LeftPageFooterContent (Object)** – contenuto dei piè di pagina delle pagine pari (servizio `com.sun.star.sheet.HeaderFooterContent`)
- **LeftPageFRightPageFooterContentooterContent (Object)** – contenuto dei piè di pagina delle pagine dispari (servizio `com.sun.star.sheet.HeaderFooterContent`)

Se non è necessario distinguere tra intestazioni e piè di pagina di pagine pari e dispari (ovvero la proprietà `FooterIsShared` è `False`), impostare le proprietà delle intestazioni e dei piè di pagina sulle pagine dispari.

Tutti gli oggetti citati restituiscono un oggetto che supporta il servizio `com.sun.star.sheet.HeaderFooterContent`. Per mezzo delle proprietà (non originali) `LeftText`, `CenterText` e `RightText`, questo servizio mette a disposizione tre elementi di testo per le intestazioni e i piè di pagina di StarOffice Calc.

L'esempio seguente scrive il valore "Solo un test." nel campo di testo di sinistra dell'intestazione del modello "Default".

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object
Dim HContent As Object
Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Solo una prova."
DefPage.RightPageHeaderContent = HContent
```

Si noti l'ultima riga dell'esempio: una volta modificato il testo, per rendere effettiva la modifica l'oggetto `TextContent` deve essere di nuovo assegnato all'intestazione.

Per i documenti di testo (StarOffice Writer) è disponibile un altro meccanismo di modifica del testo di intestazioni e piè di pagina, perché in questo contesto sono formati da un singolo blocco di testo. Le proprietà seguenti sono definite nel servizio `com.sun.star.style.PageProperties`:

- **HeaderText (Object)** – oggetto testo con i contenuti dell'intestazione (servizio `com.sun.star.text.XText`)
- **HeaderTextLeft (Object)** – oggetto testo con i contenuti dell'intestazione delle pagine di sinistra (servizio `com.sun.star.text.XText`)

- **HeaderTextRight (Object)** – oggetto testo con i contenuti dell'intestazione delle pagine di destra (servizio `com.sun.star.text.XText`)
- **FooterText (Object)** – oggetto testo con i contenuti del piè di pagina (servizio `com.sun.star.text.XText`)
- **FooterTextLeft (Object)** – oggetto testo con i contenuti dei piè di pagina delle pagine di sinistra (servizio `com.sun.star.text.XText`)
- **FooterTextRight (Object)** – oggetto testo con i contenuti dei piè di pagina delle pagine di destra (servizio `com.sun.star.text.XText`)

L'esempio seguente crea un'intestazione nel modello di pagina "Default" per i documenti di testo e aggiunge il testo "Solo un test" all'intestazione.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Solo una prova."
```

In questo caso, l'accesso viene fornito direttamente tramite la proprietà `HeaderText` del modello di pagina invece che dall'oggetto `HeaderFooterContent`.

Centraggio (solo fogli elettronici)

Il servizio `com.sun.star.sheet.TablePageStyle` è utilizzato solo nei modelli di pagina di StarOffice Calc e permette di centrare sulla pagina le aree di celle da stampare. Questo servizio dispone delle proprietà seguenti:

- **CenterHorizontally (Boolean)** – i contenuti della tabella vengono centrati orizzontalmente
- **CenterVertically (Boolean)** – i contenuti della tabella vengono centrati verticalmente

Definizione degli elementi da stampare (solo fogli elettronici)

Durante la formattazione dei fogli è possibile definire gli elementi della pagina da rendere visibili. A tal fine, il servizio `com.sun.star.sheet.TablePageStyle` fornisce le proprietà seguenti:

- **PrintAnnotations (Boolean)** – stampa i commenti delle celle
- **PrintGrid (Boolean)** – stampa la griglia delle celle
- **PrintHeaders (Boolean)** – stampa le intestazioni di righe e colonne
- **PrintCharts (Boolean)** – stampa i diagrammi contenuti in un foglio
- **PrintObjects (Boolean)** – stampa gli oggetti incorporati
- **PrintDrawing (Boolean)** – stampa gli oggetti di disegno
- **PrintDownFirst (Boolean)** – se i contenuti di un foglio si estendono su diverse pagine, vengono prima stampati in ordine decrescente verticale, procedendo in direzione inferiore destra.
- **PrintFormulas (Boolean)** – stampa le formule invece dei valori calcolati
- **PrintZeroValues (Boolean)** – stampa gli zero

Modifica efficiente dei fogli elettronici

La sezione precedente descriveva la struttura principale dei fogli elettronici, mentre la presente illustra i servizi che permettono di accedere rapidamente alle singole celle o a gruppi di celle.

Aree di celle

Oltre a un oggetto per singole celle (servizio `com.sun.star.table.Cell`), StarOffice fornisce anche oggetti che rappresentano aree di celle. Gli oggetti `CellRange` vengono creati utilizzando la chiamata `getCellRangeByName` dell'oggetto foglio elettronico:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Foglio 1")
CellRange = Sheet.getCellRangeByName("A1:C15")
```

Nel foglio elettronico, si utilizza un segno di due punti (:) per specificare un'area di celle. Ad esempio, A1:C15 rappresenta tutte le celle nelle righe dalla 1 alla 15 nelle colonne A, B e C.

La posizione delle singole celle in un'area di celle può essere determinata con il metodo `getCellByPosition`, in cui le coordinate della cella superiore sinistra dell'area sono (0, 0). L'esempio seguente si avvale precisamente di questo metodo per creare un oggetto della cella C3.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Foglio 1")
CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

Formattazione di aree di celle

Proprio come alle singole celle, è possibile applicare la formattazione anche ad aree di celle, utilizzando il servizio `com.sun.star.table.CellProperties`. Per ulteriori informazioni ed esempi di questo servizio, consultare la sezione *Formattazione*.

Calcolo con le aree di celle

Il metodo `computeFunction` permette di eseguire operazioni matematiche su aree di celle. `computeFunction` prevede come parametro una costante che descrive la funzione matematica da utilizzare. Le costanti associate sono definite nell'enumerazione `com.sun.star.sheet.GeneralFunction`. Sono disponibili i seguenti valori:

- **SUM** – somma di tutti i valori numerici
- **COUNT** – numero totale di tutti i valori (compresi i valori non numerici)
- **COUNTNUMS** - numero totale di tutti i valori numerici
- **AVERAGE** - media di tutti i valori numerici
- **MAX** – valore numerico più grande
- **MIN** - valore numerico più piccolo
- **PRODUCT** – prodotto di tutti i valori numerici
- **STDEV** – deviazione standard
- **VAR** - varianza
- **STDEVP** - deviazione standard basata sulla popolazione totale
- **VARP** – varianza basata sulla popolazione totale

L'esempio seguente calcola il valore medio dell'area di celle `A1:C3` e stampa il risultato in una casella di messaggi:

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Foglio 1")
CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Eliminazione dei contenuti delle celle

Il metodo `clearContents` semplifica il processo di eliminazione dei contenuti di celle ed aree di celle in quanto elimina un tipo specifico di contenuto da un'area di celle.

L'esempio seguente rimuove tutte le stringhe e le informazioni di formattazione diretta dall'area `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)
```

I flag specificati in `clearContents` provengono dall'elenco di costanti `com.sun.star.sheet.CellFlags`, che dispone degli elementi seguenti:

- **VALUE** – valori numerici non formattati come data o ora
- **DATETIME** – valori numerici formattati come data o ora
- **STRING** - stringhe
- **ANNOTATION** – commenti collegati alle celle
- **FORMULA** - formule
- **HARDATTR** – formattazione diretta delle celle
- **STYLES** – formattazione indiretta
- **OBJECTS** – oggetti disegno collegati alle celle
- **EDITATTR** – formattazione dei caratteri applicabile solo a parte delle celle

È inoltre possibile aggiungere le costanti assieme per eliminare informazioni diverse utilizzando una chiamata da `clearContents`.

Ricerca e sostituzione dei contenuti delle celle

I fogli elettronici, come i documenti di testo, forniscono una funzione per la ricerca e la sostituzione.

Gli oggetti descrittori per la ricerca e la sostituzione nei fogli elettronici non vengono creati direttamente tramite l'oggetto documento, ma tramite l'elenco `Sheets`. Il seguente è un esempio di un processo di ricerca e sostituzione:

```
Dim Doc As Object
Dim Sheet As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

ReplaceDescriptor = Sheet.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.Sheets.Count - 1
   Sheet = Doc.Sheets(I)
   Sheet.ReplaceAll(ReplaceDescriptor)
Next I
```

Questo esempio utilizza la prima pagina del documento per creare un `ReplaceDescriptor` e quindi lo applica a tutte le pagine in un ciclo.

CAPITOLO **8**

Disegni e presentazioni

Questo capitolo fornisce un'introduzione alla creazione e alla modifica dei disegni controllata da macro. La prima sezione descrive la struttura dei disegni, compresi gli elementi di base che contengono disegni. La seconda sezione tratta funzioni di modifica più complesse, come il raggruppamento, la rotazione e la scalatura.

Le informazioni sulla creazione, l'apertura e il salvataggio dei disegni sono riportante nel Capitolo 5, *Lavorare con i documenti di StarOffice*.

La struttura dei disegni

StarOffice non limita il numero delle pagine di un documento disegno. Inoltre, è possibile progettare ogni pagina separatamente e non esistono limiti al numero degli elementi del disegno che si possono aggiungere a un pagina.

Il quadro è reso leggermente più complesso dalla presenza dei *livelli*. Per impostazione predefinita, ogni disegno contiene i livelli *Layout*, *Campi di controllo* e *Linee di quotatura* e tutti gli elementi di disegno vengono aggiunti al livello *Layout*. Si ha inoltre l'opzione di aggiungere nuovi livelli. Per ulteriori informazioni sui livelli del disegno, consultare la StarOffice Developer's Guide.

Pagine

Le pagine di un disegno sono disponibili tramite l'elenco `DrawPages`. È possibile accedere alle singole pagine sia attraverso il numero che dal nome. Se un documento ha una pagina ed è denominato *Slide 1*, gli esempi seguenti sono identici.

Esempio 1:

```
Dim Doc As Object
Dim Page As Object
```

```
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

Esempio 2:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Slide 1")
```

Nell'esempio 1, la pagina è identificata dal suo numero (il conteggio inizia a 0). Nel secondo esempio, si accede alla pagina utilizzando il suo nome e il metodo `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
```

La chiamata precedente restituisce un oggetto pagina che supporta il servizio `com.sun.star.drawing.DrawPage`. Questo servizio riconosce le proprietà seguenti:

- **BorderLeft (Long)** – bordo sinistro in centesimi di millimetro
- **BorderRight (Long)** – bordo destro in centesimi di millimetro
- **BorderTop (Long)** – bordo superiore in centesimi di millimetro
- **BorderBottom (Long)** – bordo inferiore in centesimi di millimetro
- **Width (long)** – larghezza della pagina in centesimi di millimetro
- **Height (Long)** – altezza della pagina in centesimi di millimetro
- **Number (Short)** – numero di pagine (la numerazione inizia con 1), sola lettura
- **Orientation (Enum)** – orientamento della pagina (conforme all'enumerazione `com.sun.star.view.PaperOrientation`)

La modifica di queste impostazioni incide su *tutte* le pagine del documento.

L'esempio seguente imposta il formato della pagina di un disegno aperto su 20 x 20 centimetri con un margine della pagina di 0,5 centimetri:

```
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
```

Proprietà elementari degli oggetti di disegno

Gli oggetti disegno includono forme (rettangoli, cerchi e così via), linee e oggetti testo, che condividono una serie di funzioni comuni e supportano il servizio `com.sun.star.drawing.Shape`. Questo servizio definisce le proprietà `Size` e `Position` di un oggetto di disegno.

StarOffice Basic offre inoltre diversi altri servizi tramite i quali modificare tali proprietà, come la formattazione o l'applicazione di riempimenti. Le opzioni di formattazione disponibili dipendono dal tipo di oggetto disegno.

L'esempio seguente crea a inserisce un rettangolo nel disegno:

```
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)
```

Questo esempio utilizza la chiamata `StarDesktop.CurrentComponent` per determinare quale documento è aperto. L'oggetto documento così determinato restituisce la prima pagina del disegno tramite la chiamata `drawPages(0)`.

Le strutture `Point` e `Size` con il punto di origine (angolo sinistro) e il formato dell'oggetto disegno vengono quindi inizializzate. Le lunghezze sono specificate in centesimi di millimetro.

Il codice del programma utilizza quindi la chiamata `Doc.createInstance` per creare l'oggetto disegno rettangolo come specificato dal servizio `com.sun.star.drawing.RectangleShape`. Al termine, l'oggetto disegno è assegnato a una pagina utilizzando una chiamata `Page.add`.

Proprietà di riempimento

Questa sezione descrive quattro servizi e in ciascun caso il codice del programma di esempio utilizza un elemento di forma rettangolare che combina diversi tipi di formattazione. Le proprietà di riempimento vengono combinate nel servizio `com.sun.star.drawing.FillProperties`.

StarOffice riconosce quattro tipi principali di formattazione per un'area di riempimento. La variante più semplice è il riempimento a colori singoli. Le opzioni per definire le sfumature di colore e i tratteggi permettono di creare e introdurre altri colori. La quarta variante è l'opzione che consente di proiettare immagini preesistenti nell'area di riempimento.

Il modo riempimento di un oggetto disegno viene definito tramite la proprietà `FillStyle`. I valori consentiti sono definiti in `com.sun.star.drawing.FillStyle`.

Riempimenti a colori singoli

La proprietà principale per i riempimenti a colori singoli è la seguente:

- **FillColor (Long)** – colore di riempimento di un'area.

Per utilizzare il modo riempimento, dovete impostare la proprietà `FillStyle` sul modo di riempimento `SOLID`.

L'esempio seguente crea una forma rettangolare e un riempimento rosso (valore RGB 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)
```

Sfumatura di colore

Se si imposta la proprietà `FillStyle` su `GRADIENT`, è possibile applicare una sfumatura di colore a qualsiasi area di riempimento di un documento StarOffice.

Per applicare una sfumatura di colore predefinito, è possibile assegnare il nome associato della proprietà `FillTransparenceGradientName`. Per definire una sfumatura di colore personalizzata, dovete completare una struttura `com.sun.star.awt.Gradient` per assegnare la proprietà `FillGradient`, che dispone delle opzioni seguenti:

- **Style (Enum)** – tipo di sfumatura, ad esempio, lineare o radiale (valori predefiniti conformi a `com.sun.star.awt.GradientStyle`)
- **StartColor (Long)** – colore iniziale di una sfumatura di colore
- **EndColor (Long)** - colore finale di una sfumatura di colore
- **Angle (Short)** – angolo della sfumatura di colore in decimi di grado
- **XOffset (Short)** – coordinata X in corrispondenza della quale inizia la sfumatura di colore, specificata in centesimi di millimetro
- **YOffset (Short)** - coordinata Y in corrispondenza della quale inizia la sfumatura di colore, specificata in centesimi di millimetro
- **StartIntensity (Short)** - intensità di `StartColor` come percentuale (in StarOffice Basic, si possono specificare anche valori superiori al 100 percento)
- **EndIntensity (Short)** - intensità di `EndColor` come percentuale (in StarOffice Basic, si possono specificare anche valori superiori al 100 percento)
- **StepCount (Short)** – numero di gradazioni di colore che StarOffice deve calcolare per le sfumature

L'esempio seguente dimostra l'uso delle sfumature di colore con l'ausilio della struttura `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)
```

Questo esempio crea una sfumatura di colore lineare (`Style = LINEAR`). La sfumatura inizia con il rosso (`StartColor`) nell'angolo superiore sinistro e si estende a un angolo di 45 gradi (`Angle`) fino al verde (`EndColor`) nell'angolo inferiore destro. L'intensità di colore dei colori iniziale e finale è 150 percento (`StartIntensity` e `EndIntensity`) che produce il fatto che i colori sembrino più brillanti rispetto ai valori specificati nelle proprietà `StartColor` e `EndColor`. La sfumatura di colore è rappresentata utilizzando un centinaio di singoli colori sfumati (`StepCount`).

Tratteggi

Per creare un riempimento tratteggiato, la proprietà `FillStyle` deve essere impostata su `HATCH`. Il codice del programma per definire il tratteggio è molto simile al codice per le sfumature di colore. Ancora una struttura ausiliaria, in questo caso `com.sun.star.drawing.Hatch`, è utilizzata per definire l'aspetto dei tratteggi. La struttura dei tratteggi ha le seguenti proprietà:

- **Style (Enum)** – tipo di tratteggio: semplice, squadrato, o squadrato con diagonali (valori predefiniti conformi a `com.sun.star.awt.HatchStyle`)
- **Color (Long)** – colore delle linee
- **Distance (Long)** – distanza tra le linee, espressa in centesimi di millimetro
- **Angle (Short)** – angolo del tratteggio in decimi di grado

L'esempio seguente dimostra l'uso di una struttura a tratteggio:

```
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)
```

Questo codice crea una struttura a tratteggi semplice (`HatchStyle = SINGLE`) le cui linee sono ruotate di 45 gradi (`Angle`). Le linee sono grigio scuro (`Color`) e sono distanziate di 0,2 millimetri (`Distance`).

Bitmap

Per utilizzare la proiezione bitmap come riempimento, impostate la proprietà `FillStyle` su `BITMAP`. Se la bitmap è già disponibile in StarOffice, è necessario specificare il nome nella proprietà `FillBitMapName` e il modello di visualizzazione (semplice, affiancata, o allungata) nella proprietà `FillBitmapMode` (valori predefiniti conformi a `com.sun.star.drawing.BitmapMode`).

Per utilizzare un file bitmap esterno, specificatene l'URL nella proprietà `FillBitmapURL`.

L'esempio seguente crea una rettangolo e affianca la bitmap Sky disponibile in StarOffice per riempire l'area del rettangolo:

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
```

```
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
RectangleShape.FillBitmapName = "Sky"
RectangleShape.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT

Page.add(RectangleShape)
```

Trasparenza

È possibile regolare la trasparenza di qualsiasi riempimento applicato. Il modo più semplice di modificare la trasparenza di un elemento di disegno è quello di utilizzare la proprietà `FillTransparence`.

L'esempio seguente crea un rettangolo rosso con una trasparenza del 50 percento.

```
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)
```

Per rendere trasparente il riempimento, impostare la proprietà `FillTransparence` su 100.

Oltre alla proprietà `FillTransparence`, il servizio `com.sun.star.drawing.FillProperties` fornisce anche la proprietà `FillTransparenceGradient`, che permette di definire una sfumatura che specifica la trasparenza dell'area di riempimento.

Proprietà delle linee

Tutti gli oggetti disegno che hanno un bordo supportano il servizio `com.sun.star.drawing.LineStyle`. Di seguito sono elencate alcune delle proprietà offerte da questo servizio:

- **LineStyle (Enum)** – tipo di linea (valori predefiniti conformi a `com.sun.star.drawing.LineStyle`)
- **LineColor (Long)** – colore della linea
- **LineTransparence (Short)** – trasparenza della linea
- **LineWidth (Long)** – spessore della linea espresso in centesimi di millimetro
- **LineJoint (Enum)** - transizioni ai punti di connessione (valori predefiniti conformi a `com.sun.star.drawing.LineJoint`)

L'esempio seguente crea un rettangolo con bordo pieno (`LineStyle = SOLID`) di 5 millimetri di spessore (`LineWidth`) e trasparente al 50 percento. I bordi destro e sinistro della linea si estendono ai loro punti di reciproca intersezione (`LineJoint = MITER`) per formare un angolo retto.

```
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)
```

Oltre alle proprietà elencate, il servizio com.sun.star.drawing.LineStyle fornisce opzioni per il disegno di linee punteggiate o tratteggiate. Per maggiori informazioni, consultate il riferimento della API StarOffice.

Proprietà del testo (oggetti di disegno)

I servizi com.sun.star.style.CharacterProperties e com.sun.star.style.ParagraphProperties possono formattare il testo negli oggetti di disegno. Questi servizi sono relativi ai singoli caratteri e paragrafi e sono descritti in maggiore dettaglio nel Capitolo 6 (*Documenti di testo*).

L'esempio seguente inserisce il testo in un rettangolo e formatta il servizio dei caratteri 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 = "Questo è un test"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

Questo codice utilizza la proprietà String del rettangolo per inserire il testo e le proprietà CharWeight e CharFontName del servizio com.sun.star.style.CharacterProperties per formattare il carattere del testo.

Il testo può essere inserito solo una volta aggiunto l'oggetto di disegno alla pagina di disegno. Potete inoltre utilizzare il servizio com.sun.star.drawing.Text per posizionare e formattare il testo nell'oggetto di disegno. Di seguito sono riportate alcune delle proprietà più importanti di questo servizio:

- **TextAutoGrowHeight (Boolean)** – adatta l'altezza dell'elemento di disegno al testo che contiene
- **TextAutoGrowWidth (Boolean)** - adatta la larghezza dell'elemento di disegno al testo che contiene

- **TextHorizontalAdjust (Enum)** – posizione orizzontale del testo all'interno dell'elemento di disegno (valori predefiniti conformi a `com.sun.star.drawing.TextHorizontalAdjust`)
- **TextVerticalAdjust (Enum)** – posizione verticale del testo all'interno dell'elemento di disegno (valori predefiniti conformi a `com.sun.star.drawing.TextVerticalAdjust`)
- **TextLeftDistance (Long)** – distanza a sinistra tra l'elemento di disegno e il testo in centesimi di millimetro
- **TextRightDistance (Long)** – distanza a destra tra l'elemento di disegno e il testo in centesimi di millimetro
- **TextUpperDistance (Long)** - distanza superiore tra l'elemento di disegno e il testo in centesimi di millimetro
- **TextLowerDistance (Long)** - distanza inferiore tra l'elemento di disegno e il testo in centesimi di millimetro

L'esempio seguente dimostra l'uso delle proprietà citate.

```
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 = "Questa è una prova"   ' Solo dopo 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
```

Questo codice inserisce un elemento di disegno in una pagina, quindi aggiunge il testo all'angolo superiore sinistro dell'oggetto disegno utilizzando le proprietà `TextVerticalAdjust` e `TextHorizontalAdjust`. La distanza minima tra il bordo del testo dell'oggetto disegno è impostato su tre millimetri.

Proprietà dell'ombra

È possibile aggiungere un'ombra alla maggior parte degli oggetti disegno con il servizio `com.sun.star.drawing.ShadowProperties`. Le proprietà di questo servizio sono:

- **Shadow (Boolean)** – attiva l'ombra
- **ShadowColor (Long)** – colore dell'ombra
- **ShadowTransparence (Short)** - trasparenza dell'ombra
- **ShadowXDistance (Long)** – distanza verticale dell'ombra dall'oggetto disegno in centesimi di millimetro
- **ShadowYDistance (Long)** – distanza orizzontale dell'ombra dall'oggetto disegno in centesimi di millimetro

L'esempio seguente crea una rettangolo con un'ombra scostata di 2 millimetri dal rettangolo in senso verticale e orizzontale. L'ombra è resa in grigio scuro con il 50 percento di trasparenza.

```
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)
```

Panoramica dei diversi oggetti di disegno

Forme rettangolari

Gli oggetti di forma rettangolare (`com.sun.star.drawing.RectangleShape`) supportano i seguenti servizi di formattazione:

- **Proprietà di riempimento** – `com.sun.star.drawing.FillProperties`
- **Proprietà delle linee** – `com.sun.star.drawing.LineProperties`
- **Proprietà del testo** – `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`)
- **Proprietà dell'ombra** – `com.sun.star.drawing.ShadowProperties`
- **CornerRadius (Long)** – raggio per l'arrotondamento degli angoli espresso in centesimi di millimetro

Cerchi ed ellissi

Il servizio `com.sun.star.drawing.EllipseShape` è responsabile di cerchi ed ellissi e supporta i seguenti servizi:

- **Proprietà di riempimento** – `com.sun.star.drawing.FillProperties`
- **Proprietà delle linee** – `com.sun.star.drawing.LineProperties`
- **Proprietà del testo** – `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`)
- **Proprietà dell'ombra** – `com.sun.star.drawing.ShadowProperties`

Oltre a questi servizi, cerchi ed ellissi offrono anche le seguenti proprietà:

- **CircleKind (Enum)** – tipo di cerchio o ellissi (valori predefiniti conformi a `com.sun.star.drawing.CircleKind`)
- **CircleStartAngle (Long)** – angolo iniziale in decimi di grado (solo per segmenti di cerchio o ellissi)
- **CircleEndAngle (Long)** – angolo finale in decimi di grado (solo per segmenti di cerchio o ellissi)

La proprietà `CircleKind` determina se un oggetto è un cerchio completo, una porzione circolare o una sezione di un cerchio. Sono disponibili i seguenti valori:

- **com.sun.star.drawing.CircleKind.FULL** – cerchio completo o ellissi completa
- **com.sun.star.drawing.CircleKind.CUT** – sezione di cerchio (cerchio parziale le cui interfacce sono collegate direttamente)
- **com.sun.star.drawing.CircleKind.SECTION** – porzione di cerchio
- **com.sun.star.drawing.CircleKind.ARC** – angolo (non comprendente la linea del cerchio)

L'esempio seguente crea una porzione circolare con un angolo di 70 gradi (prodotta dalla differenza tra l'angolo iniziale di 20 gradi e l'angolo finale di 90 gradi)

```
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)
```

Linee

StarOffice fornisce il servizio `com.sun.star.drawing.LineShape` per gli oggetti linea. Gli oggetti linea supportano tutti i servizi di formattazione generale ad eccezione delle aree. Di seguito sono riportate tutte le proprietà associate al servizio `LineShape`:

- **Proprietà delle linee** – `com.sun.star.drawing.LineProperties`
- **Proprietà del testo** – `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`)
- **Proprietà dell'ombra** – `com.sun.star.drawing.ShadowProperties`

L'esempio seguente crea e formatta una linea con l'ausilio delle proprietà citate. L'origine della linea è specificata nella proprietà `Location`, mentre le coordinate elencate nella proprietà `Size` specificano il punto finale della linea.

```
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)
```

Forme polipoligonali

StarOffice supporta anche forme poligonali complesse tramite il servizio `com.sun.star.drawing.PolyPolygonShape`. In senso stretto, un *polipoligono* non è un poligono semplice ma un poligono multiplo. Si possono pertanto specificare elenchi indipendenti contenenti punti angolari e combinarli per formare un oggetto completo.

Come per le forme rettangolari, tutte le proprietà di formattazione degli oggetti di disegno vengono fornite anche per i polipoligoni:

- **Proprietà di riempimento** – `com.sun.star.drawing.FillProperties`
- **Proprietà delle linee** – `com.sun.star.drawing.LineProperties`
- **Proprietà del testo** – `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`)
- **Proprietà dell'ombra** – `com.sun.star.drawing.ShadowProperties`

Il servizio `PolyPolygonShape` dispone anche di una proprietà che permette di definire le coordinate di un poligono:

- `PolyPolygon (Array)` – campo contenente le coordinate del poligono (doppio array con punti del tipo `com.sun.star.awt.Point`)

L'esempio seguente mostra come definire un triangolo con il servizio `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 deve essere eseguito prima
                             ' dell'impostazione delle coordinate

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())
```

Dato che i punti di un poligono sono definiti come valori assoluti, non è necessario specificare le dimensioni o la posizione iniziale di un poligono. Al contrario, occorre creare un array dei punti, includere questo array in un secondo array (utilizzando la chiamata `Array(Coordinates())` e quindi assegnare l'array al poligono. Prima di effettuare la chiamata corrispondente, dovrete inserire il poligono nel documento.

Il doppio array nella definizione permette di creare forme complesse mediante l'unione di diversi poligoni. Ad esempio, è possibile creare un rettangolo e quindi inserirne un altro al suo interno per creare un foro nel rettangolo originale:

```
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 deve essere eseguito prima
                             ' dell'impostazione delle coordinate

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())
```

Per quanto riguarda quali aree sono riempite e quali sono vuote, StarOffice applica una semplice regola: il bordo della forma esterna corrisponde sempre al bordo esterno del polipoligono. La linea interna successiva costituisce il bordo interno della forma e marca la transizione al primo foro. Se è presente un'altra linea interna, quest'ultima contraddistingue la transizione a un'area riempita.

Immagini

Gli ultimi elementi di disegno qui presentati sono oggetti grafici basati sul servizio `com.sun.star.drawing.GraphicObjectShape`. Potrete avvalervene con qualsiasi immagine in StarOffice il cui aspetto possa essere adattato utilizzando un'intera gamma di proprietà.

Gli oggetti grafici supportano due delle proprietà generali di formattazione:

- **Proprietà del testo** – `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` e `com.sun.star.style.ParagraphProperties`)
- **Proprietà dell'ombra** – `com.sun.star.drawing.ShadowProperties`

Le proprietà addizionali supportate dagli oggetti grafici sono le seguenti:

- **GraphicURL (String)** - URL dell'immagine
- **AdjustLuminance (Short)** – luminanza dei colori, espressa come percentuale (sono ammessi anche i valori negativi)
- **AdjustContrast (Short)** – contrasto, espresso come percentuale (sono ammessi anche i valori negativi)
- **AdjustRed (Short)** – valore del rosso, espresso come percentuale (sono ammessi anche i valori negativi)
- **AdjustGreen (Short)** – valore del verde, espresso come percentuale (sono ammessi anche i valori negativi)
- **AdjustBlue (Short)** – valore del blu, espresso come percentuale (sono ammessi anche i valori negativi)
- **Gamma (Short)** - valore gamma di un'immagine
- **Transparency (Short)** – trasparenza di un'immagine, espressa come percentuale
- **GraphicColorMode (enum)** – modalità del colore, per esempio, standard, scala di grigi, bianco e nero (valore predefinito conforme a `com.sun.star.drawing.ColorMode`)

L'esempio seguente mostra come inserire una pagina in un oggetto grafico 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          ' specifiche, non significative perché le ultime
                        coordinate sono vincolanti
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)
```

Questo codice inserisce l'immagine `test.jpg` e ne adatta l'aspetto utilizzando le proprietà `Adjust`. In questo esempio, le immagini sono raffigurate come trasparenti al 40 percento senza che abbiano luogo altre conversioni di colore (`GraphicColorMode = STANDARD`).

Modifica degli oggetti di disegno

Raggruppare gli oggetti

In molte situazioni è utile raggruppare diversi singoli oggetti di disegno in modo che si comportino come un unico grande oggetto.

L'esempio seguente combina due oggetti di disegno:

```
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
' crea un elemento di disegno quadrato
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' crea un elemento di disegno circolare
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)
' combina gli elementi di disegno quadrato e circolare
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' centra gli elementi di disegno combinati
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
```

Questo codice crea un rettangolo e un cerchio e li inserisce nella pagina, quindi crea un oggetto che supporta il servizio `com.sun.star.drawing.ShapeCollection` e utilizza il metodo `Add` per aggiungere il rettangolo e il cerchio a questo oggetto. `ShapeCollection` viene aggiunto alla pagina utilizzando il metodo `Group` e restituisce l'oggetto `Group` effettivo che può essere modificato come una singola `Shape`.

Per formattare i singoli oggetti di un gruppo, applicare la formattazione prima di aggiungerli al gruppo. Una volta inserti nel gruppo non è più possibile modificare gli oggetti.

Rotazione e troncatura degli oggetti di disegno

Tutti gli oggetti di disegno descritti nelle sezioni precedenti possono essere ruotati e troncati anche utilizzando il servizio `com.sun.star.drawing.RotationDescriptor`.

Questo servizio dispone delle proprietà seguenti:

- **RotateAngle (Long)** – angolo di rotazione espresso in centesimi di grado
- **ShearAngle (Long)** – angolo di troncatura espresso in centesimi di grado

L'esempio seguente crea un rettangolo e lo ruota di 30 gradi utilizzando la proprietà `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)
```

L'esempio seguente crea lo stesso rettangolo che nell'esempio precedente, ma procede a troncarlo a 30 gradi utilizzando la proprietà `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)
```

Ricerca e sostituzione

I documenti di disegno, come i documenti di testo, forniscono una funzione per la ricerca e la sostituzione. Questa funzione è simile a quella utilizzata nei documenti di testo come descritto nel Capitolo 6, *Documenti di testo*. Tuttavia, nei documenti disegno gli oggetti descrittori per la ricerca e la sostituzione non vengono creati direttamente tramite l'oggetto documento, ma tramite il livello carattere associato. L'esempio seguente delinea il processo di sostituzione con un disegno:

```
Dim Doc As Object
Dim Page As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

ReplaceDescriptor = Page.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.drawPages.Count - 1
   Page = Doc.drawPages(I)
   Page.ReplaceAll(ReplaceDescriptor)
Next I
```

Questo codice utilizza la prima `DrawPage` del documento per creare un `ReplaceDescriptor` e quindi lo applica a tutte le pagine del documento di disegno in un ciclo.

Presentazioni

Le presentazioni di StarOffice si basano sui documenti disegno. Ogni pagina della presentazione è una diapositiva. Si accede alle diapositive nello stesso modo in cui si accede a un disegno standard tramite l'elenco `DrawPages` dell'oggetto documento. Il servizio `com.sun.star.presentation.PresentationDocument`, responsabile dei documenti di presentazione, fornisce anche il servizio `com.sun.star.drawing.DrawingDocument` completo.

Operazioni sulle presentazioni

Oltre alle funzioni di disegno fornite dalla proprietà `Presentation`, il documento presentazione ha un oggetto presentazione che garantisce l'accesso alle proprietà principali e ai meccanismi di controllo per le presentazioni. Ad esempio, questo oggetto fornisce un metodo `start` che può avviare le presentazioni.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

Il codice utilizzato in questo esempio crea un oggetto `Doc` che fa riferimento al documento di presentazione corrente e stabilisce l'oggetto presentazione associato. Il metodo `start()` dell'oggetto è utilizzato per avviare l'esempio ed eseguire la presentazione a video.

I metodi seguenti sono forniti come oggetti presentazione:

- **start** – avvia la presentazione
- **end** - termina la presentazione
- **rehearseTimings** – avvia la presentazione dall'inizio e determina il suo tempo di esecuzione

Sono disponibili le seguenti proprietà:

- **AllowAnimations (Boolean)** – esegue le animazioni nella presentazione
- **CustomShow (String)** – permette di specificare il nome della presentazione in modo che faccia riferimento al nome nella presentazione
- **FirstPage (String)** – nome della diapositiva con cui avviare la presentazione
- **IsAlwaysOnTop (Boolean)** – visualizza sempre la finestra della presentazione come prima finestra sullo schermo
- **IsAutomatic (Boolean)** – esegue automaticamente la presentazione
- **IsEndless (Boolean)** – riavvia la presentazione dall'inizio una volta terminata
- **IsFullScreen (Boolean)** – avvia automaticamente la presentazione in modalità a tutto schermo
- **IsMouseVisible (Boolean)** – visualizza il mouse durante la presentazione
- **Pause (long)** – durata di visualizzazione dello schermo vuoto al termine della presentazione
- **StartWithNavigator (Boolean)** – visualizza la finestra del Navigatore, quindi avvia la presentazione
- **UsePn (Boolean)** – visualizza il puntatore durante la presentazione

CAPITOLO 9

Diagrammi (grafici)

StarOffice permette di visualizzare i dati sotto forma di diagramma, che crea collegamenti grafici tra i dati sotto forma di barre, diagrammi a torta, linee ed altri elementi. I dati si possono visualizzare come immagini 2D o 3D e l'aspetto degli elementi dei diagrammi può essere adattato singolarmente in modo simile al processo utilizzato per gli elementi di disegno.

Se i dati sono disponibili in forma di foglio elettronico, questo può essere collegato dinamicamente al diagramma. Tutte le modifiche ai dati base possono in questo caso essere visualizzate immediatamente nel diagramma assegnato. Questo capitolo offre una presentazione generale dell'interfaccia di programmazione per i moduli di diagrammi di StarOffice e si concentra sull'uso dei diagrammi all'interno dei fogli elettronici.

Uso dei diagrammi nei fogli elettronici

I diagrammi non vengono trattati come documenti indipendenti in StarOffice, ma come oggetti integrati in un documento preesistente.

Mentre i diagrammi nei documenti di testo e disegni rimangono isolati dal contenuto del documento, quando sono utilizzati nei fogli elettronici viene fornito un meccanismo che permette di stabilire un collegamento tra i dati del documento e i diagrammi integrati. L'esempio seguente spiega l'interazione tra il foglio elettronico e il diagramma:

```
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("Grafico", Rect, RangeAddress(), True, True)
```

Sebbene il codice utilizzato nell'esempio possa apparire complesso, i processi centrali sono limitati a tre linee: la prima linea centrale crea la variabile del documento `Doc`, che fa riferimento al foglio elettronico corrente (riga `Doc = StarDesktop.CurrentComponent`). Il codice utilizzato nell'esempio crea quindi un elenco contenente tutti i diagrammi del primo foglio elettronico (riga `Charts` line = `Doc.Sheets(0).Charts`). Infine, viene aggiunto un nuovo diagramma all'ultima riga di questo elenco utilizzando il metodo `addNewByName`. Questo nuovo diagramma diventa quindi visibile all'utente.

L'ultima riga inizializza le strutture ausiliarie `Rect` e `RangeAddress`, che il metodo `addNewByName` fornisce anche come parametro. `Rect` determina la posizione del diagramma all'interno del foglio elettronico. `RangeAddress` determina l'area per la quale collegare i dati al diagramma.

L'esempio precedente crea un diagramma a barre. Qualora fosse necessario un tipo diverso di grafico, dovrete sostituire esplicitamente il diagramma a barre:

```
Chart = Charts.getByName("Grafico").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

Le prime righe definiscono l'oggetto grafico corrispondente. La seconda riga sostituisce il diagramma corrente con uno nuovo – in questo esempio, un diagramma a linee.

Nota – In Excel, si opera una distinzione tra i diagrammi che sono stati inseriti come pagina separata in un documento di Excel e i grafici che sono integrati in una pagina della tabella. Analogamente, per i grafici sono qui definiti due diversi metodi di accesso. Questa distinzione non è operata in StarOffice Basic, perché i grafici in StarOffice Calc vengono sempre creati come oggetti integrati di una pagina di tabella. Si accede ai grafici sempre con l'elenco `Charts` dell'oggetto `Sheet` associato.

La struttura dei diagrammi

La struttura di un diagramma – e pertanto l'elenco di servizi e interfacce supportate – dipende dal tipo del diagramma. I metodi e le proprietà dell'asse Z sono, ad esempio, disponibili solo nei diagrammi 3D, ma non nei diagrammi 2D. Nei grafici a torta, non vi sono interfacce per lavorare con gli assi.

I singoli elementi di un diagramma

Titolo, sottotitolo e legenda

Titolo, sottotitolo e legenda fanno parte degli elementi base di ogni diagramma. I diagrammi forniscono i propri oggetti per ciascuno di questi elementi. L'oggetto diagramma `Chart` fornisce le seguenti proprietà per l'amministrazione degli elementi:

- **HasMainTitle (Boolean)** – attiva il titolo.
- **Title (Object)** – oggetto con informazioni dettagliate sul titolo del diagramma (supporta il servizio `com.sun.star.chart.ChartTitle`).
- **HasSubTitle(Boolean)** – attiva il sottotitolo.
- **Subtitle (Object)** – oggetto con informazioni dettagliate sul sottotitolo del diagramma (supporta il servizio `com.sun.star.chart.ChartTitle`).
- **HasLegend (Boolean)** – attiva la legenda.
- **Legend (Object)** – oggetto con informazioni dettagliate sul titolo del diagramma (supporta il servizio `com.sun.star.chart.ChartLegendPosition`).

Per molti aspetti, gli elementi specificati corrispondono a un elemento di disegno. Ciò è dovuto al fatto che i servizi `com.sun.star.chart.ChartTitle` e `com.sun.star.chart.ChartLegendPosition` supportano entrambe il servizio `com.sun.star.drawing.Shape`, che forma la base del programma tecnico per gli elementi di disegno.

Gli utenti hanno pertanto l'opportunità di determinare la posizione e le dimensioni dell'elemento utilizzando le proprietà `Size` e `Position`.

Per la formattazione degli elementi sono disponibili altre proprietà di riempimento e linee (i servizi `com.sun.star.drawing.FillProperties` e `com.sun.star.drawing.LineStyle`) nonché le altre proprietà dei caratteri (servizio `com.sun.star.style.CharacterProperties`).

`com.sun.star.chart.ChartTitle` contiene non solo le proprietà di formattazione citate, ma anche le altre due proprietà seguenti:

- **TextRotation (Long)** – angolo di rotazione del testo in centesimi di grado.
- **String (String)** – testo da visualizzare come titolo o sottotitolo.

La legenda (servizio `com.sun.star.chart.ChartLegend`) contiene la seguente proprietà addizionale:

- **Alignment (Enum)** – posizione in cui compare la legenda (valore predefinito conforme a `com.sun.star.chart.ChartLegendPosition`).

L'esempio seguente crea un diagramma e vi assegna 'Test' come titolo, 'Test 2' come sottotitolo e una legenda. La legenda ha un colore di sfondo grigio, è situata in fondo al diagramma e dimensioni del carattere di 7 punti.

```
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("Grafico", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("Grafico").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
```

Sfondo

Ogni diagramma ha un'area di sfondo. Ogni area ha un oggetto, accessibile utilizzando le proprietà seguenti dell'oggetto diagramma:

- **Area (Object)** – area di sfondo del diagramma (supporta il servizio `com.sun.star.chart.ChartArea`).

Lo sfondo di un diagramma copre la sua area completa, compresa l'area sotto al titolo, sottotitolo e legenda del diagramma. Il servizio `com.sun.star.chart.ChartArea` associato supporta le proprietà di linee e riempimento e non fornisce altre proprietà estese.

Pareti e basi dei diagrammi

Sebbene lo sfondo del diagramma copra l'intera area del diagramma, la parete posteriore del diagramma è limitata all'area direttamente dietro all'area dei dati.

Per i diagrammi 3D esistono generalmente due pareti: una dietro l'area dei dati e una come demarcazione sinistra verso l'asse Y. I diagrammi 3D hanno in genere anche una base o pavimento.

- **Floor (Object)** – pannello della base del diagramma (solo per i diagrammi 3D, supporta il servizio `com.sun.star.chart.ChartArea`).
- **Wall (Object)** – pareti del diagramma (solo per i diagrammi 3D, supporta il servizio `com.sun.star.chart.ChartArea`).

Gli oggetti specificati supportano il servizio `com.sun.star.chart.ChartArea`, che a sua volta fornisce le classiche proprietà di riempimento e linee (servizi `(com.sun.star.drawing.FillProperties` e `com.sun.star.drawing.LineStyle`, consultare il Capitolo 8).

Si accede alle pareti e alla base del diagramma tramite l'oggetto `Chart`, che a sua volta fa parte dell'oggetto Chart:

```
Chart.Area.FillBitmapName = "Cielo"
```

L'esempio seguente mostra come l'immagine (denominata Cielo) già contenuta in StarOffice può essere utilizzata come sfondo per un diagramma.

```
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("Grafico", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("Grafico").EmbeddedObject

Chart.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = "Cielo"
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Assi

StarOffice riconosce cinque diversi assi utilizzabili in un diagramma. Nel suo scenario più semplice, sono gli assi X e Y. Quando si utilizzano i diagrammi 3D, viene talvolta fornito anche un asse Z. Per i diagrammi in cui i valori delle diverse righe di dati si scostano reciprocamente in modo significativo, StarOffice fornisce un secondo asse X e un secondo asse Y per le altre operazioni di scalatura.

Primi assi X, Y e Z

Oltre all'asse effettivo, per ciascuno dei primi assi X, Y e Z può essere presente un titolo, una descrizione, una griglia e una griglia ausiliaria. Si ha l'opzione se visualizzare o nascondere tutti questi elementi. L'oggetto diagramma fornisce le seguenti proprietà per la gestione di tali funzioni (prendendo l'esempio di un asse X; le proprietà per l'asse Y e Z sono strutturate nello stesso modo):

- **HasXAxis (Boolean)** – attiva l'asse X.
- **XAxis (Object)** – oggetto con informazioni dettagliate sull'asse X (supporta il servizio `com.sun.star.chart.ChartAxis`).
- **HasXAxisDescription (Boolean)** – attiva la descrizione per l'asse X.
- **HasXAxisGrid (Boolean)** – attiva la griglia principale per l'asse X.
- **XMainGrid (Object)** – oggetto con informazioni dettagliate sulla griglia principale per l'asse X (supporta il servizio `com.sun.star.chart.ChartGrid`).
- **HasXAxisHelpGrid (Boolean)** – attiva la griglia ausiliaria per l'asse X.
- **XHelpGrid (Object)** – oggetto con informazioni dettagliate sulla griglia ausiliaria per l'asse X (supporta il servizio `com.sun.star.chart.ChartGrid`).
- **HasXAxisTitle (Boolean)** – attiva il titolo dell'asse X.
- **XAxisTitle (Object)** – oggetto con informazioni dettagliate sul titolo dell'asse X (supporta il servizio `com.sun.star.chart.ChartTitle`).

Secondo asse X e asse Y

Le proprietà seguenti sono disponibili per i secondi assi X e Y (proprietà prendendo esempio dal secondo asse X):

- **HasSecondaryXAxis (Boolean)** – attiva il secondo asse X.
- **SecondaryXAxis (Object)** – oggetto con informazioni dettagliate sul secondo asse X (supporta il servizio `com.sun.star.chart.ChartAxis`).
- **HasSecondaryXAxisDescription (Boolean)** – attiva la descrizione dell'asse X.

Proprietà degli assi

Gli oggetti assi di un diagramma di StarOffice supportano il servizio `com.sun.star.chart.ChartAxis`. Oltre alle proprietà per i caratteri (servizio `(com.sun.star.style.CharacterProperties`, consultare il Capitolo 6) e per le linee (servizio `com.sun.star.drawing.LineStyle`, consultare il Capitolo 8), fornisce anche le seguenti proprietà:

- **Max (Double)** – valore massimo per l'asse.
- **Min (Double)** - valore minimo per l'asse.
- **Origin (Double)** – punto di intersezione per l'attraversamento degli assi.
- **StepMain (Double)** – distanza tra due linee primarie dell'asse.
- **StepHelp (Double)** - distanza tra due linee secondarie dell'asse.
- **AutoMax (Boolean)** – determina automaticamente il valore massimo per l'asse.
- **AutoMin (Boolean)** - determina automaticamente il valore minimo per l'asse.
- **AutoOrigin (Boolean)** – determina automaticamente il punto di intersezione per l'attraversamento degli assi.
- **AutoStepMain (Boolean)** - determina automaticamente la distanza tra le linee primarie di un asse.
- **AutoStepHelp (Boolean)** - determina automaticamente la distanza tra le linee secondarie di un asse.
- **Logarithmic (Boolean)** – scala gli assi in modo logaritmico (invece che lineare).
- **DisplayLabels (Boolean)** – attiva l'etichetta di testo per gli assi.
- **TextRotation (Long)** – angolo di rotazione dell'etichetta di testo degli assi, espresso in centesimi di grado.
- **Marks (Const)** – costante che specifica se le linee primarie dell'asse devono essere all'interno o all'esterno dell'area del diagramma (valori predefiniti conformi a `com.sun.star.chart.ChartAxisMarks`)
- **HelpMarks (Const)** – costante che specifica se le linee secondarie dell'asse devono essere all'interno e/o all'esterno dell'area del diagramma (valori predefiniti conformi a `com.sun.star.chart.ChartAxisMarks`)
- **Overlap (Long)** – valore percentuale che specifica in che misura le barre dei diversi gruppi di dati possono sovrapporsi (al 100%, le barre sono visualizzate completamente sovrapposte, a -100%, vi è tra esse una distanza pari alla larghezza di una barra).

- **GapWidth (long)** - valore percentuale che specifica la distanza che può esservi tra i diversi gruppi di barre di un diagramma (al 100%, vi è tra esse una distanza pari alla larghezza di una barra).
- **ArrangeOrder (enum)** – dettagli della posizione di iscrizione; oltre al posizionamento su una riga, è possibile anche suddividere l'etichetta alternativamente su due righe (valore predefinito conforme a `com.sun.star.chart.ChartAxisArrangeOrderType`)
- **TextBreak (Boolean)** - consente le interruzioni di riga.
- **TextCanOverlap (Boolean)** - consente le sovrapposizioni di testo.
- **NumberFormat (Long)** – formato numerico (consultare la sezione "Formati di numeri, data e testo" a pagina 136)

Proprietà della griglia degli assi

L'oggetto per la griglia degli assi è basato sul servizio `com.sun.star.chart.ChartGrid`, che a sua volta supporta le proprietà delle linee del servizio `com.sun.star.drawing.LineStyle` (consultare il Capitolo 8).

Proprietà del titolo degli assi

Gli oggetti per la formattazione del titolo degli assi sono basati sul servizio `com.sun.star.chart.ChartTitle`, utilizzato anche per i titoli dei diagrammi.

Esempio

L'esempio seguente crea un diagramma a linee. Il colore della parete posteriore del diagramma è impostato sul bianco. Gli assi X e Y hanno una griglia ausiliaria grigia per l'orientamento visivo. Il valore minimo dell'asse Y è fissato su 0, mentre quello massimo è 100, in modo che la risoluzione del diagramma venga conservata anche se i valori vengono modificati.

```
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("Grafico", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("Grafico").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
```

Diagrammi 3D

La maggior parte dei diagrammi di StarOffice può essere visualizzata con immagini 3D. Tutti i tipi di diagrammi che forniscono questa opzione supportano il servizio `com.sun.star.chart.Dim3Ddiagram`, che dispone di una sola proprietà:

- **Dim3D (Boolean)** – attiva la visualizzazione 3D.

Diagrammi sovrapposti

I diagrammi sovrapposti sono diagrammi organizzati con diversi singoli valori impilati per produrre un valore totale. Questa vista mostra non solo i singoli valori, ma anche una panoramica di tutti i valori.

In StarOffice, vari tipi di diagrammi possono essere visualizzati in forma sovrapposta. Tutti questi diagrammi supportano il servizio `com.sun.star.chart.StackableDiagram`, che a sua volta fornisce le proprietà seguenti:

- **Stacked (Boolean)** – attiva la modalità di visualizzazione sovrapposta.
- **Percent (Boolean)** – invece dei valori assoluti, visualizza la loro distribuzione percentuale.

Tipi di diagrammi

Diagrammi a linee

I diagrammi a linee (servizio `com.sun.star.chart.LineDiagram`) supportano un asse X, due assi Y e un asse Z. Sono visualizzabili come immagini 2D o 3D (servizio `com.sun.star.chart.Dim3Ddiagram`). Le linee si possono sovrapporre (`com.sun.star.chart.StackableDiagram`).

I diagrammi a linee forniscono le proprietà seguenti:

- **SymbolType (const)** – simbolo per la visualizzazione dei punti dati (costante conforme a `com.sun.star.chart.ChartSymbolType`).
- **SymbolSize (Long)** – dimensioni del simbolo per la visualizzazione dei punti dati in centesimi di millimetro.
- **SymbolBitmapURL (String)** – nome dei file delle immagini per la visualizzazione dei punti dati.
- **Lines (Boolean)** – collega i punti dati per mezzo di linee.
- **SplineType (Long)** – funzione Spline per l'attenuazione delle linee (0: nessuna funzione spline, 1: spline cubiche, 2: spline B).
- **SplineOrder (Long)** – spessore polinomiale per spline (solo spline B).
- **SplineResolution (Long)** – numero di punti di supporto per il calcolo di spline.

Diagrammi ad area

I diagrammi ad area (servizio `com.sun.star.chart.AreaDiagram`) supportano un asse X, due assi Y e un asse Z. Sono visualizzabili come immagini 2D o 3D (servizio `com.sun.star.chart.Dim3Ddiagram`). Le aree si possono sovrapporre (`com.sun.star.chart.StackableDiagram`).

Diagrammi a barre

I diagrammi a barre (servizio `com.sun.star.chart.BarDiagram`) supportano un asse X, due assi Y e un asse Z. Sono visualizzabili come immagini 2D o 3D (servizio `com.sun.star.chart.Dim3Ddiagram`). Le barre si possono sovrapporre (`com.sun.star.chart.StackableDiagram`).

Forniscono le proprietà seguenti:

- **Vertical (Boolean)** – visualizza le barre verticalmente, altrimenti le visualizza in orizzontale.
- **Deep (Boolean)** - in modalità Vista 3D, posiziona le barre una dietro all'altro invece di affiancarle.
- **StackedBarsConnected (Boolean)** – collega le barre associate in un diagramma sovrapposto per mezzo di linee (disponibile solo con i grafici orizzontali).
- **NumberOfLines (Long)** – numero delle linee da visualizzare in un diagramma sovrapposto sotto forma di linee e non di barre.

Diagrammi a torta

I diagrammi a torta (servizio `com.sun.star.chart.PieDiagram`) non contengono assi e non possono essere sovrapposti. Sono visualizzabili come immagini 2D o 3D (servizio `com.sun.star.chart.Dim3Ddiagram`).

CAPITOLO **10**

Accesso ai database

StarOffice dispone di un'interfaccia di database integrata (indipendente da qualsiasi sistema) denominata Star Database Connectivity (SDBC). Questa interfaccia è stata sviluppata allo scopo specifico di fornire accesso al maggior numero possibile di sorgenti di dati

Questa operazione viene eseguita tramite driver. Le sorgenti dalle quali i driver prelevano i loro dati è irrilevante per un utente della SDBC. Alcuni driver accedono ai database basati su file e traggono i dati direttamente da essi. Altri utilizzano interfacce standard come JDBC o ODBC. Esistono tuttavia anche speciali driver che accedono alla rubrica MAPI, alle directory LDAP o ai fogli elettronici di StarOffice come sorgenti di dati.

Poiché questi driver si basano sui componenti UNO, potrete sviluppare altri driver e aprire così nuove sorgenti di dati. Ulteriori dettagli su questa attività sono disponibili nel documento StarOffice Developer's Guide.

Nota – Per quanto riguarda la concezione, la SDBC è paragonabile alle librerie ADO e DAO disponibili in VBA, ovvero consente un accesso di alto livello ai database, indipendentemente dai backend del database sottostante.

Nota – L'interfaccia con i database di StarOffice è cresciuta con il lancio di StarOffice 8. Sebbene in passato si accedesse ai database principalmente utilizzando una serie di metodi dell'oggetto `Application`, l'interfaccia in StarOffice 6 lo suddivide ora in diversi oggetti. Un `DatabaseContext` è utilizzato come oggetto radice per le funzioni del database.

Il linguaggio SQL (Structured Query Language)

Il linguaggio usato per le ricerche SDBC è SQL. Per confrontare le differenze tra le diverse varianti dell'SQL, i componenti dell'SDBC di StarOffice dispongono di un proprio parser SQL, che utilizza la finestra di ricerca per controllare i comandi SQL inseriti e corregge gli errori di sintassi più semplici, come quelli associati ai caratteri in maiuscolo e in minuscolo.

Se un driver consente di accedere a una sorgente di dati che non supporta l'SQL, occorre convertire in modo indipendente i comandi SQL trasferiti necessari all'accesso nativo.

Nota – L'implementazione SQL dalla SDBC è orientata allo standard SQL-ANSI. Le estensioni specifiche di Microsoft, come il costrutto `INNER JOIN`, non sono supportate. Dovrete quindi sostituirle con comandi standard (`INNER JOIN`, ad esempio, dovrebbe essere sostituito con una clausola `WHERE` corrispondente).

Tipi di accesso ai database

L'interfaccia di database di StarOffice è disponibile nelle applicazioni StarOffice Writer e StarOffice Calc, nonché in tutti i formulari basati su database.

In StarOffice Writer, si possono creare lettere standard con l'ausilio delle sorgenti di dati SDBC, quindi procedere alla loro stampa. È disponibile anche un'opzione per spostare i dati dalla finestra del database al documento di testo utilizzando la funzione trascina e rilascia.

Se l'utente sposta una tabella di database in un foglio elettronico, StarOffice crea un'area della tabella aggiornabile facendo clic con il mouse se i dati originali sono stati modificati. Viceversa, i dati del foglio elettronico possono essere spostati in una tabella di database ed eseguire l'importazione del database.

Infine, StarOffice fornisce un meccanismo per i formulari basati su database. Per procedere in tal senso, l'utente crea prima un formulario standard di StarOffice Writer o StarOffice Calc, quindi collega i campi a un database.

Tutte le opzioni qui specificate sono basate sull'interfaccia utente di StarOffice. Per utilizzare le funzioni corrispondenti, non è necessaria alcuna esperienza di programmazione.

Questo capitolo non fornisce tuttavia informazioni sulle funzioni specificate, ma si concentra sull'interfaccia di programmazione della SDBC, che permette ricerche automatiche del database e pertanto consente di utilizzare una serie molto maggiore di applicazioni.

È però necessario disporre di una conoscenza di base del funzionamento dei database e del linguaggio SQL per una comprensione ottimale delle sezioni seguenti.

Sorgenti di dati

È possibile incorporare un database in StarOffice mediante la creazione di una cosiddetta *sorgente di dati*. L'interfaccia utente fornisce un'opzione corrispondente per creare le sorgenti di dati nel menu **Strumenti**. Si possono però creare anche proprie sorgenti di dati e utilizzarle con StarOffice Basic.

Un oggetto contesto di database creato utilizzando la funzione `createUnoService` funge da punto iniziale per l'accesso a una sorgente di dati. Si basa sul servizio `com.sun.star.sdb.DatabaseContext` e costituisce l'oggetto radice per tutte le operazioni con i database.

L'esempio seguente mostra come creare un contesto di database e quindi utilizzarlo per determinare i nomi di tutte le sorgenti di dati disponibili, i cui nomi vengono visualizzati in una casella di messaggi.

```
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
```

Le singole sorgenti di dati si basano sul servizio `com.sun.star.sdb.DataSource` e si possono determinare dal contesto del database utilizzando il metodo `getByName`:

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Clienti")
```

L'esempio crea un oggetto `DataSource` per una sorgente di dati denominata *Clienti*.

Le sorgenti di dati offrono una serie di proprietà, che a loro volta forniscono informazioni generali sull'origine dei dati e informazioni sui metodi di accesso. Le proprietà principali sono le seguenti:

- **Name (String)** – nome della sorgente di dati.
- **URL (String)** – URL della sorgente di dati nella forma *jdbc: subprotocollo: subnome* o *sdbc: subprotocollo: subnome*.
- **Info (Array)** – array contenente coppie `PropertyValue` con parametri di collegamento (in genere almeno il nome utente e la password).
- **User (String)** – nome dell'utente.
- **Password (String)** – password dell'utente (non viene salvata).
- **IsPasswordRequired (Boolean)** – la password è necessaria e richiesta in modo interattivo dall'utente.
- **IsReadOnly (Boolean)** – consente l'accesso in sola lettura al database.
- **NumberFormatsSupplier (Object)** – oggetto contenente i formati numerici disponibili per il database (supporta l'interfaccia `com.sun.star.util.XNumberFormatsSupplier`, consultare la sezione "Formati di numeri, data e testo" a pagina 136).
- **TableFilter (Array)** – elenco dei nomi delle tabelle da visualizzare.
- **TableTypeFilter (Array)** – elenco dei tipi di tabelle da visualizzare. I valori disponibili sono `TABLE`, `VIEW` e `SYSTEM TABLE`.
- **SuppressVersionColumns (Boolean)** – nasconde la visualizzazione delle colonne utilizzate per la gestione delle versioni.

Nota – Le sorgenti di dati di StarOffice non sono paragonabili 1:1 con le sorgenti di dati in ODBC. Mentre una sorgente di dati ODBC copre solo le informazioni sull'origine dei dati, una sorgente di dati in StarOffice include anche una serie di informazioni sulle modalità di visualizzazione dei dati nelle finestre di database di StarOffice.

Ricerche

È possibile assegnare ricerche predefinite a una sorgente di dati. StarOffice prende nota dei comandi SQL delle ricerche in modo che siano disponibili in qualsiasi momento. Le ricerche (query), permettono di semplificare le operazioni con i database perché si possono aprire con un semplice clic del mouse e consentono anche agli utenti che non conoscono il linguaggio SQL di impartire comandi SQL.

Nelle ricerche è integrato un oggetto che supporta il servizio `com.sun.star.sdb.QueryDefinition`. Potete accedere alle ricerche per mezzo del metodo `QueryDefinitions` della sorgente di dati.

L'esempio seguente elenca i nomi delle ricerche delle sorgenti di dati che possono essere determinate in una casella di messaggi.

```
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("Clienti")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
   QueryDefinition = QueryDefinitions(I)
   MsgBox QueryDefinition.Name
Next I
```

Oltre alla proprietà Name utilizzata nell'esempio, `com.sun.star.sdb.QueryDefinition` fornisce una serie completa di altre proprietà:

- **Name (String)** – nome della ricerca.
- **Command (String)** – comando SQL (generalmente un comando `SELECT`).
- **UpdateTableName (String)** – per le ricerche basate su diverse tabelle: nome della tabella in cui sono possibili le modifiche di valori.
- **UpdateCatalogName (String)** – nome dei cataloghi di aggiornamento delle tabelle.
- **UpdateSchemaName (String)** – nome dei diagrammi di aggiornamento delle tabelle.

L'esempio seguente mostra come creare un oggetto ricerca in modo controllato da programma e assegnarlo a una sorgente di dati.

```
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("Clienti")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELEZIONA * DA cliente"

QueryDefinitions.insertByName("NuovaRicerca", QueryDefinition)
```

L'oggetto ricerca viene prima creato utilizzando la chiamata `createUnoService`, quindi inizializzato e successivamente inserito nell'oggetto `QueryDefinitions` per mezzo di `insertByName`.

Collegamenti con i formulari basati su database

Per semplificare il lavoro con le sorgenti di dati, StarOffice fornisce un'opzione per collegare le sorgenti di dati ai formulari basati su database. I collegamenti sono disponibili tramite il metodo `getBookmarks()`. L'operazione restituisce un contenitore (`com.sun.star.sdb.DefinitionContainer`) che ospita tutti i collegamenti della sorgente di dati. Per accedere ai segnalibri potete procedere mediante `Name` o `Index`.

L'esempio seguente determina l'URL del segnalibro *Segnalibro*.

```
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("Clienti")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("Segnalibro")
MsgBox URL
```

Accesso ai database

Per accedere a un database è necessario un collegamento con il database, ovvero un canale di trasferimento che consente la comunicazione diretta con il database. A differenza delle sorgenti di dati presentate nella sezione precedente, il collegamento al database deve essere ristabilito ad ogni riavvio del programma.

StarOffice offre diversi modi di determinare i collegamenti al database. Di seguito viene riportata una spiegazione per il metodo basato su una sorgente di dati preesistente.

```
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("Clienti")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
```

```
End If
```

Il codice utilizzato nell'esempio controlla prima se il database è protetto da una password. In caso contrario, crea il collegamento al database richiesto utilizzando la chiamata `GetConnection`. Le due stringhe vuote nella riga di comando rappresentano il nome utente e la password.

Se il database è protetto da password, l'esempio crea un `InteractionHandler` e apre il collegamento al database utilizzando il metodo `ConnectWithCompletion`. L'InteractionHandler assicura che StarOffice richieda all'utente i dati di login richiesti.

Iterazione delle tabelle

Per accedere a una tabella in StarOffice si procede generalmente dall'oggetto `ResultSet`. Un `ResultSet` è un tipo di marcatore che indica un gruppo corrente di dati all'interno di un volume di risultati ottenuti utilizzando il comando `SELECT`.

L'esempio mostra come utilizzare un `ResultSet` per ricercare i valori da una tabella del database.

```
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("Clienti")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If

Statement = Connection.createStatement()
ResultSet = Statement.executeQuery("SELECT CustomerNumber FROM Customer")

If Not IsNull(ResultSet) Then
   While ResultSet.next
      MsgBox ResultSet.getString(1)
   Wend
End If
```

Una volta stabilito il collegamento al database, il codice utilizzato nell'esempio utilizza prima la chiamata `Connection.createObject` per creare un oggetto `Statement`. Questo oggetto `Statement` utilizza quindi la chiamata `executeQuery` per restituire il `ResultSet` effettivo. Il programma controlla ora se il `ResultSet` esiste effettivamente e vaglia i record di dati utilizzando un ciclo. I valori richiesti (nell'esempio, quelli dal campo `CustomerNumber`) restituisce il `ResultSet` utilizzando il metodo `getString`, mentre il parametro 1 determina che la chiamata è relativa ai valori della prima colonna.

Nota – L'oggetto `ResultSet` dell'SDBC è paragonabile all'oggetto `Recordset` di DAO e ADO, dato che anch'esso fornisce un accesso iterativo a un database.

Nota – In StarOffice 8 si accede di fatto ai database tramite un oggetto `ResultSet`, che riflette il contenuto di una tabella o il risultato di un comando SQL-SELECT. In passato, l'oggetto `ResultSet` forniva i metodi residenti nell'oggetto `Application` per lo spostamento all'interno dei dati (per esempio `DataNextRecord`).

Metodi specifici del tipo per richiamare i valori

Come illustrato dall'esempio della sezione precedente, StarOffice fornisce un metodo `getString` per accedere ai contenuti delle tabelle. Il risultato di questo metodo assume la forma di una stringa. Sono disponibili i seguenti metodi `get`:

- `getByte()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getShort()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getInt()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getLong()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getFloat()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getDouble()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getBoolean()` – supporta i tipi di dati SQL per numeri, caratteri e stringhe.
- `getString()` – supporta tutti i tipi di dati SQL.
- `getBytes()` – supporta i tipi di dati SQL per i valori binari.
- `getDate()` – supporta i tipi di dati SQL per numeri, stringhe, data e contrassegno dell'ora.
- `getTime()` – supporta i tipi di dati SQL per numeri, stringhe, data e contrassegno dell'ora.
- `getTimestamp()` – supporta i tipi di dati SQL per numeri, stringhe, data e contrassegno dell'ora.

- `getCharacterStream()` – supporta i tipi di dati SQL per numeri, stringhe e valori binari.
- `getUnicodeStream()` – supporta i tipi di dati SQL per numeri, stringhe e valori binari.
- `getBinaryStream()` – valori binari.
- `getObject()` – supporta tutti i tipi di dati SQL.

In tutti i casi, il numero di colonne dovrebbe essere elencato come parametro di cui ricercare i valori.

Le varianti ResultSet

L'accesso ai database è spesso questione di velocità critica. StarOffice offre pertanto diversi modi per ottimizzare `ResultSets` e controllare quindi la velocità di accesso. Più funzioni fornisce `ResultSet` e più complessa sarà l'implementazione e di conseguenza più lente saranno le funzioni.

Un semplice `ResultSet`, come quello presentato nella sezione "Iterazione delle tabelle", fornisce la più ampia gamma di funzioni disponibili. Consente di applicare l'iterazione solo in avanti e per i valori oggetto dell'interrogazione. Opzioni di navigazione più estese, come la possibilità di modificare i valori, non sono pertanto incluse.

L'oggetto Statement utilizzato per creare il `ResultSet` fornisce alcune proprietà che consentono di incidere sulle funzioni del `ResultSet`:

- **ResultSetConcurrency (const)** – specifiche sulla possibilità di modificare i dati o meno (specifiche conformi a `com.sun.star.sdbc.ResultSetConcurrency`).
- **ResultSetType (const)** – specifiche riguardanti il tipo di `ResultSets` (specifiche conformi a `com.sun.star.sdbc.ResultSetType`).

I valori definiti in `com.sun.star.sdbc.ResultSetConcurrency` sono:

- **UPDATABLE** - `ResultSet` consente di modificare i valori.
- **READ_ONLY** – `ResultSet` non consente modifiche.

Il gruppo di costanti `com.sun.star.sdbc.ResultSetConcurrency` fornisce le seguenti specifiche:

- **FORWARD_ONLY** – `ResultSet` consente solo lo spostamento in avanti.
- **SCROLL_INSENSITIVE** – `ResultSet` consente qualsiasi tipo di spostamento, tuttavia le modifiche ai dati originali non sono registrate.
- **SCROLL_SENSITIVE** – `ResultSet` consente qualsiasi tipo di spostamento; le modifiche ai dati originali incidono su `ResultSet`.

Nota – Un `ResultSet` contenente le proprietà `READ_ONLY` e `SCROLL_INSENSITIVE` corrisponde a un gruppo di record del tipo `Snapshot` in ADO e DAO.

Quando si utilizzano le proprietà `UPDATEABLE` e `SCROLL_SENSITIVE` di `ResultSet`, la portata della funzione di un `ResultSet` è paragonabile a un `Recordset` di tipo `Dynaset` di ADO e DAO.

Metodi per lo spostamento nei ResultSets

Se un `ResultSet` è di tipo `SCROLL_INSENSITIVE` o `SCROLL_SENSITIVE`, supporta una serie completa di metodi di spostamento nel gruppo di dati. I metodi principali sono i seguenti:

- **next()** – spostamento al record di dati successivo.
- **previous()** – spostamento al record di dati precedente.
- **first()** – spostamento al primo record di dati.
- **last()** – spostamento all'ultimo record di dati.
- **beforeFirst()** – spostamento a prima del primo record di dati.
- **afterLast()** – spostamento a dopo l'ultimo record di dati.

Tutti i metodi restituiscono un parametro booleano (logico) che specifica se lo spostamento ha avuto successo.

Per determinare la posizione corrente del cursore, sono disponibili i seguenti metodi di test, che restituiscono tutti un valore logico:

- **isBeforeFirst()** – `ResultSet` è prima del primo record di dati.
- **isAfterLast()** – `ResultSet` è dopo l'ultimo record di dati.
- **isFirst()** – `ResultSet` è il primo record di dati.
- **isLast()** – `ResultSet` è l'ultimo record di dati.

Modifica dei record di dati

Se è stato creato un `ResultSet` con il valore `ResultSetConcurrency = UPDATEABLE`, il suo contenuto può essere modificato. Ciò è applicabile solo finché il comando SQL consente di riscrivere i dati nel database (dipende dal principio). Non è possibile, ad esempio, con i comandi SQL complessi con colonne collegate o valori accumulati.

L'oggetto `ResultSet` fornisce i metodi `Update` per modificare i valori, che sono strutturati nello stesso modo dei metodi `get` per il richiamo dei valori. Il metodo `updateString`, ad esempio, consente di scrivere una stringa.

Dopo la modifica, i valori devono essere trasferiti nel database utilizzando il metodo `updateRow()`. La chiamata ha luogo prima del comando di navigazione successivo, altrimenti i valori andranno persi.

Se viene commesso un errore durante le modifiche, è possibile procedere all'annullamento utilizzando il metodo `cancelRowUpdates()`. Questa chiamata è disponibile solo se i dati non sono stati sovrascritti nel database con `updateRow()`.

CAPITOLO **11**

Finestre di dialogo

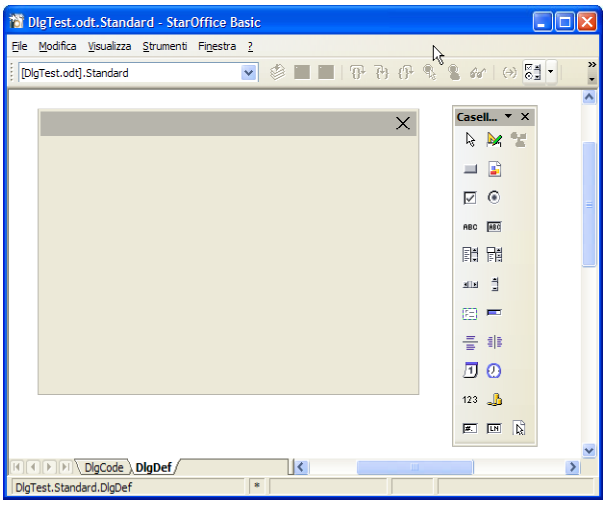
È possibile aggiungere finestre di dialogo e formulari personalizzati ai documenti di StarOffice, quindi collegarli a loro volta alle macro di StarOffice Basic per ampliare notevolmente la gamma di applicazione di StarOffice Basic. Le finestre di dialogo possono visualizzare, ad esempio, informazioni del database o guidare gli utenti in un processo dettagliato per la creazione di un nuovo documento sotto forma di Pilota automatico.

Uso delle finestre di dialogo

Le finestre di dialogo di StarOffice Basic sono formate da una finestra di dialogo contenente campi di testo, caselle di riepilogo, pulsanti di scelta e altri elementi di controllo.

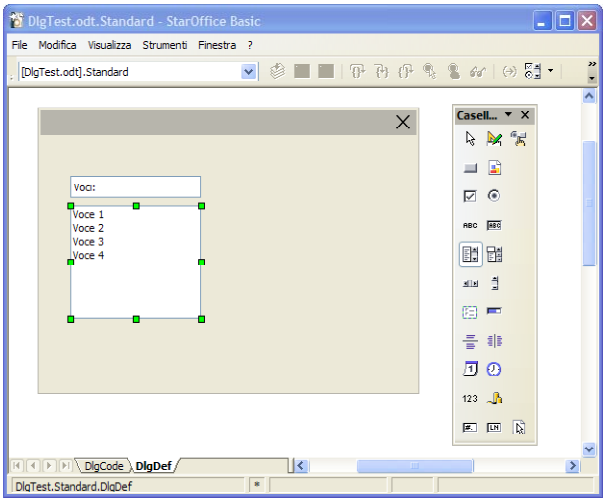
Creazione di finestre di dialogo

È possibile creare e strutturare le finestre di dialogo avvalendosi del Dialog Editor di StarOffice, utilizzabile nello stesso modo di StarOffice Draw:



In pratica, si trascinano gli elementi di controllo dalla tavolozza di progettazione (a destra) nell'area delle finestre di dialogo in cui definire posizione e dimensioni.

L'esempio mostra una finestra di dialogo contenente un'etichetta e una casella di riepilogo.



Per aprire una finestra di dialogo potete utilizzare il codice seguente:

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
```

```
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` crea un oggetto denominato `Dlg` che fa riferimento alla finestra di dialogo associata. Prima di creare la finestra di dialogo, accertarsi che la libreria utilizzata (in questo esempio, la libreria `Standard`) sia caricata. In caso contrario, è il metodo `LoadLibrary` ad eseguire questa operazione.

Una volta inizializzato l'oggetto `Dlg`, potete utilizzare il metodo `Execute` per visualizzare la finestra di dialogo. Le finestre di dialogo come questa sono descritte come modali perché non consentono nessun altra azione del programma fino alla loro chiusura. Con la finestra di dialogo aperta, il programma rimane nella chiamata `Execute`.

Il metodo `dispose` alla fine del codice approva le risorse utilizzate dalla finestra di dialogo al termine del programma.

Chiusura delle finestre di dialogo

Chiusura con OK o Annulla

Se una finestra di dialogo contiene un pulsante **OK** o **Annulla**, è possibile chiuderla automaticamente con la pressione di uno di questi pulsanti. Per ulteriori informazioni sull'utilizzo di questi pulsanti, consultate la sezione Dettagli sugli elementi di controllo delle finestre di dialogo nel presente capitolo.

Se si chiude una finestra di dialogo facendo clic sul pulsante **OK**, il metodo `Execute` restituisce il valore 1, altrimenti viene restituito il valore 0.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
   MsgBox "Premuto Ok"
Case 0
   MsgBox "Premuto Annulla"
End Select
```

Chiusura con il pulsante Chiudi nella barra del titolo

Se lo si desidera, è possibile chiudere una finestra di dialogo facendo clic sul pulsante Chiudi nella barra del titolo della finestra di dialogo. In questo esempio, il metodo `Execute` della finestra di dialogo restituisce il valore 0, equivalente alla pressione del pulsante Annulla.

Chiusura con una chiamata esplicita del programma

Per chiudere una finestra di dialogo aperta è possibile procedere anche con il metodo `endExecute`:

```
Dlg.endExecute()
```

Accesso ai singoli elementi di controllo

Una finestra di dialogo può contenere qualsiasi numero di elementi di controllo. Si accede a questi elementi tramite il metodo `getControl` che restituisce il nome dell'elemento di controllo.

```
Dim Ctl As Object

Ctl = Dlg.getControl("Pulsante")
Ctl.Label = "Nuova etichetta"
```

Questo codice determina l'oggetto per l'elemento di controllo `Pulsante` quindi inizializza la variabile oggetto `Ctl` con un riferimento all'elemento. Infine, il codice imposta la proprietà `Label` dell'elemento di controllo sul valore `Nuova dicitura`.

Nota – StarOffice Basic distingue tra i caratteri in maiuscolo e quelli in minuscolo per i nomi degli elementi di controllo.

Uso del *modello* di finestre di dialogo ed elementi di controllo

La divisione tra elementi del programma visibili (*Vista*) e dati o documenti retrostanti (*Modello*) si verifica in molti punti della API StarOffice. Oltre ai metodi e alle proprietà degli elementi di controllo, gli oggetti finestre di dialogo e gli elementi di controllo dispongono di un oggetto `Model` subordinato, che permette di accedere direttamente ai contenuti di una finestra di dialogo o di un elemento di controllo.

Nelle finestre di dialogo, la distinzione tra dati e raffigurazione non è sempre così chiara come in altre aree della API di StarOffice. Gli elementi della API sono disponibili tramite la Vista e il Modello.

La proprietà `Model` fornisce un accesso controllato da programma al modello degli oggetti finestra di dialogo ed elemento di controllo.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

Questo esempio disattiva il pulsante `cmdNtext` nella finestra di dialogo `Dlg` con l'ausilio dell'oggetto modello di `cmdNtext`.

Proprietà

Nome e titolo

Tutti gli elementi di controllo hanno un proprio nome che può essere ricercato utilizzando la seguente proprietà del modello:

- **Model.Name (String)** – nome dell'elemento di controllo

È possibile specificare il titolo che compare nella barra del titolo di una finestra di dialogo con la seguente proprietà del modello:

- **Model.Title (String)** – titolo della finestra di dialogo (si applica solo alle finestre di dialogo).

Posizione e dimensione

Potete ricercare la dimensione e la posizione di un elemento di controllo utilizzando le seguenti proprietà dell'oggetto modello:

- **Model.Height (long)** – altezza dell'elemento di controllo (in unità ma)
- **Model.Width (long)** – larghezza dell'elemento di controllo (in unità ma)
- **Model.PositionX (long)** – posizione X dell'elemento di controllo, misurata dal bordo interno sinistro della finestra di dialogo (in unità ma)
- **Model.PositionY (long)** – posizione Y dell'elemento di controllo, misurata dal bordo interno superiore della finestra di dialogo (in unità ma)

Per garantire l'indipendenza dalla piattaforma per l'aspetto delle finestre di dialogo, StarOffice utilizza l'unità interna *Map AppFont (ma)* per specificare la posizione e le dimensioni nelle finestre di dialogo. Un'unità ma è definita come un ottavo dell'altezza media di un carattere rispetto al font di sistema definito nel sistema operativo e un quarto della sua larghezza. Mediante l'uso delle unità ma, StarOffice garantisce che una finestra di dialogo abbia lo stesso aspetto su sistemi diversi che utilizzando impostazioni diverse.

Per modificare le dimensioni o la posizione degli elementi di controllo per il runtime, determinare le dimensioni totali della finestra di dialogo e regolare i valori per gli elementi di controllo sui corrispondenti rapporti delle parti.

Nota – Map AppFont (ma) sostituisce le unità Twips per consentire una migliore indipendenza dalla piattaforma.

Attivazione e sequenza delle tabulazioni

È possibile spostarsi attraverso gli elementi di controllo in qualsiasi finestra di dialogo mediante la pressione del tasto Tab. Le proprietà seguenti sono disponibili in questo contesto nel modello degli elementi di controllo:

- **Model.Enabled (Boolean)** – attiva l'elemento di controllo
- **Model.Tabstop (Boolean)** – consente di raggiungere l'elemento di controllo con il tasto Tab
- **Model.TabIndex (Long)** – posizione dell'elemento di controllo nell'ordine di attivazione

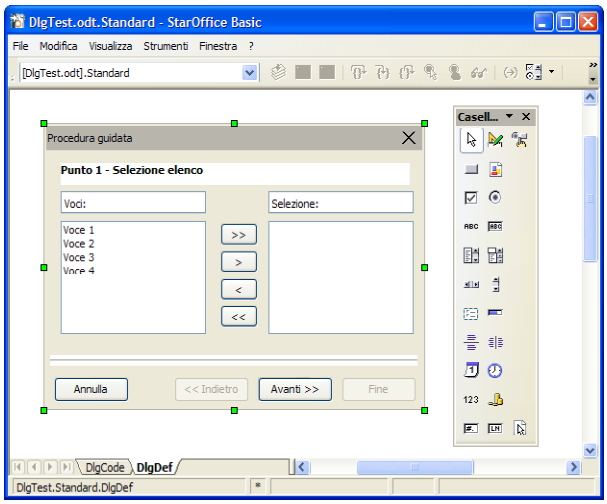
Infine, l'elemento di controllo fornisce un metodo `getFocus` che garantisce l'attivazione dell'elemento sottostante:

- **getFocus** – l'elemento di controllo viene attivato (solo per le finestre di dialogo)

Finestre di dialogo a più pagine

Una finestra di dialogo in StarOffice può disporre di più schede. La proprietà `Stepp` di una finestra di dialogo definisce la scheda della finestra di dialogo, mentre la stessa proprietà `Step` per un elemento di controllo specifica la scheda in cui deve essere visualizzato l'elemento di controllo.

Il valore `Step` di 0 è un caso speciale. Se si imposta questo valore su zero in una finestra di dialogo, tutti gli elementi di controllo saranno visibili, indipendentemente dal loro valore di `Step`. Analogamente, se impostate il valore su zero per un elemento di controllo, quest'ultimo verrà visualizzato in tutte le schede della finestra di dialogo.



Nell'esempio precedente, potete assegnare il valore di `Step` pari a 0 alla linea di divisione nonché ai pulsanti `Cancel`, `Prev`, `Next` e `Done` per visualizzare questi elementi su tutte le pagine. Potrete assegnare gli elementi anche a una singola scheda (per esempio, la 1).

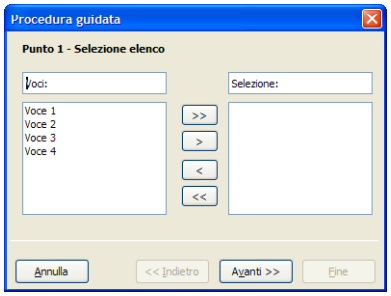
Il codice di programma riportato di seguito mostra come aumentare o ridurre il valore `Step` nei gestori di eventi dei pulsanti `Next` e `Prev` e come modificare lo stato dei pulsanti.

```
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
```

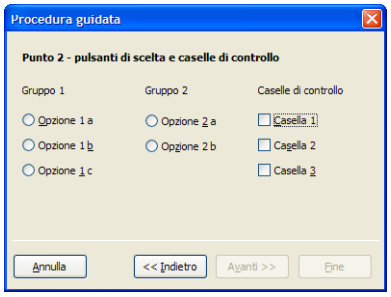
```
End Sub
```

Per rendere possibile questo esempio, dovete includere una variabile globale `Dlg` che faccia riferimento a una finestra di dialogo aperta. L'aspetto della finestra cambia nel modo seguente:

Pagina 1:



Pagina 2:



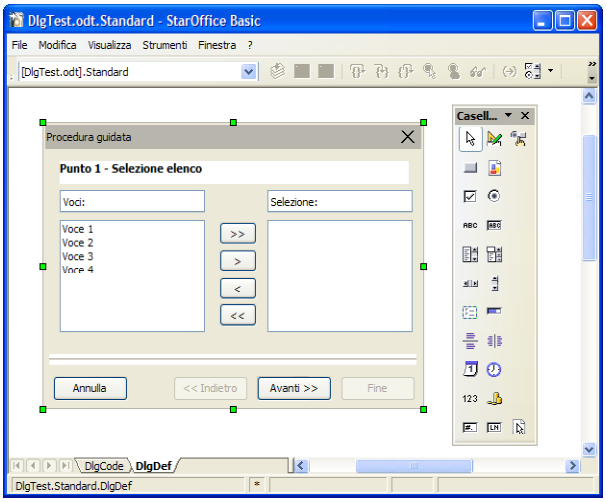
Eventi

Le finestre di dialogo e i formulari di StarOffice si basano su un modello di programmazione orientato agli eventi in cui è possibile assegnare i *gestori di eventi* agli elementi di controllo. Un gestore di eventi esegue una procedura predefinita quando si verifica un'azione particolare, persino quando tale azione è costituita da un altro evento. Con i gestori di eventi si possono inoltre modificare documenti o aprire database nonché accedere ad altri elementi di controllo.

Gli elementi di controllo di StarOffice riconoscono i diversi tipi di eventi attivabili nelle diverse situazioni. Questi tipi di eventi si possono suddividere in quattro categorie:

- **Controllo con il mouse**: eventi che corrispondono alle azioni del mouse (ad esempio, semplici movimenti del mouse o clic su una particolare posizione dello schermo).
- **Controllo con la tastiera**: eventi che vengono attivati dalla pressione dei diversi tasti della tastiera.
- **Modifica dell'attivazione**: eventi eseguiti da StarOffice quando gli elementi di controllo sono attivati o disattivati.
- **Eventi specifici degli elementi di controllo**: eventi che si verificano solo in relazione a determinati elementi di controllo.

Per lavorare proficuamente con gli eventi, accertatevi di creare la finestra di dialogo associata nell'ambiente di sviluppo StarOffice e che contenga gli elementi di controllo o i documenti richiesti (se intendete applicare gli eventi a un formulario).



La figura sopra mostra l'ambiente di sviluppo StarOffice Basic con una finestra di dialogo contenente due caselle di riepilogo. È possibile spostare i dati da un elenco all'altro utilizzando i pulsanti tra le due caselle di riepilogo.

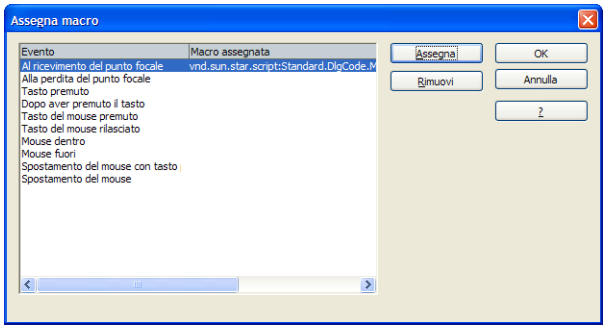
Per visualizzare il layout a video, occorre creare le procedure StarOffice Basic associate, in modo che possano essere richiamate dai gestori di eventi. Anche se potete utilizzare queste procedure in ogni modulo, è consigliabile limitarne l'uso a due soli moduli. Per semplificare la lettura del codice, si consiglia inoltre di assegnare nomi espliciti e rappresentativi alle diverse procedure. Saltare direttamente a una procedura

di programma generale da una macro può infatti produrre del codice non chiaro. Al contrario, per semplificare la manutenzione e la risoluzione degli errori del codice, si dovrebbe creare un'altra procedura che funga da punto di ingresso per la gestione degli eventi – anche se esegue solo una singola chiamata alla procedura di destinazione.

Il codice riportato nell'esempio seguente sposta una voce dalla casella di riepilogo di sinistra a quella di destra di una finestra di dialogo.

```
Sub cmdSelect_Initiated
   Dim objList As Object
   lstEntries = Dlg.getControl("lstEntries")
   lstSelection = Dlg.getControl("lstSelection")

   If lstEntries.SelectedItem > 0 Then
      lstSelection.AddItem(lstEntries.SelectedItem, 0)
      lstEntries.removeItems(lstEntries.SelectItemPos, 1)
   Else
      Beep
   End If
End Sub
```

Se questa procedura è stata creata in StarOffice Basic, è possibile assegnarla a un evento richiesto utilizzando la finestra delle proprietà del Dialog Editor.



La finestra di dialogo elencherà tutte le procedure di StarOffice Basic. Per assegnare una procedura a un evento, selezionarla, quindi fare clic su **Assegna**.

Parametri

Il verificarsi di un particolare evento non è sempre sufficiente ad ottenere una risposta adeguata: potrebbero infatti essere richieste informazioni addizionali. Ad esempio, per elaborare un clic del mouse, può essere necessaria la posizione dello schermo in cui è stato premuto il mouse.

In StarOffice Basic, potete utilizzare i parametri degli oggetti per fornire a una procedura ulteriori informazioni su un evento, ad esempio:

```
Sub ProcessEvent(Event As Object)

End Sub
```

La precisione con cui è strutturato l'oggetto `Event` e le sue proprietà dipendono dal tipo di evento attivato dalla chiamata della procedura. Le sezioni successive descrivono i tipi di eventi in dettaglio.

Indipendentemente dal tipo di evento, tutti gli oggetti permettono di accedere all'elemento di controllo pertinente e al suo modello. L'elemento di controllo può essere raggiunto utilizzando

```
Event.Source
```

e il suo modello utilizzando

```
Event.Source.Model
```

Si possono impiegare queste proprietà per attivare un evento all'interno di un gestore di eventi.

Eventi del mouse

StarOffice Basic riconosce i seguenti eventi del mouse:

- **Mouse moved** – l'utente sposta il mouse
- **Mouse moved while key pressed** – l'utente trascina il mouse mentre tiene premuto un tasto
- **Mouse button pressed** – l'utente preme un pulsante del mouse
- **Mouse button released** – l'utente rilascia un pulsante del mouse
- **Mouse outside** – l'utente sposta il mouse fuori dalla finestra corrente

La struttura degli oggetti eventi associati è definita nella struttura `com.sun.star.awt.MouseEvent` che fornisce le informazioni seguenti:

- **Buttons (short)** – pulsante premuto (una o più costanti in conformità a `com.sun.star.awt.MouseButton`).
- **X (long)** – coordinata X del mouse, misurata in pixel dall'angolo superiore sinistro dell'elemento di controllo
- **Y (long)** – coordinata Y del mouse, misurata in pixel dall'angolo superiore sinistro dell'elemento di controllo
- **ClickCount (long)** – numero di clic associati all'evento del mouse (se StarOffice può rispondere con velocità sufficiente, `ClickCount` è 1 anche per un doppio clic perché viene iniziato solo un singolo evento).

Le costanti definite in `com.sun.star.awt.MouseButton` per i pulsanti del mouse sono:

- **LEFT** – pulsante di sinistra del mouse
- **RIGHT** – pulsante di destra del mouse
- **MIDDLE** – pulsante centrale del mouse

L'esempio seguente restituisce la posizione del mouse nonché il pulsante del mouse premuto:

```
Sub MouseUp(Event As Object)
   Dim Msg As String
   Msg = "Pulsanti: "
   If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
      Msg = Msg & "LEFT "
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
      Msg = Msg & "RIGHT "
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
      Msg = Msg & "MIDDLE "
   End If
   Msg = Msg & Chr(13) & "Posizione: "
   Msg = Msg & Event.X & "/" & Event.Y
   MsgBox Msg
End Sub
```

Nota – Gli eventi `Click` e `Doubleclick` di VBA non sono disponibili in StarOffice Basic. Al loro posto, utilizzate l'evento `MouseUp` di StarOffice Basic per l'evento `click` e imitate l'evento `Doubleclick` modificando la logica dell'applicazione.

Eventi della tastiera

In StarOffice Basic sono disponibili i seguenti eventi della tastiera:

- **Key pressed** – l'utente preme un tasto
- **Key released** – l'utente rilascia un tasto

Entrambe gli eventi sono relativi alle azioni di tasti *logici* e non ad azioni *fisiche*. Se l'utente preme diversi tasti per produrre un singolo carattere (ad esempio, per aggiungere un accento a un carattere), StarOffice Basic crea solo un evento.

Una singola azione del tasto su un tasto di modifica, come il tasto Maiusc o il tasto Alt non crea un evento indipendente.

Le informazioni su un tasto premuto vengono fornite dall'oggetto evento che StarOffice Basic fornisce alla procedura per la gestione degli eventi, contenente le proprietà seguenti:

- **KeyCode (short)** – codice del tasto premuto (valori predefiniti conformi a `com.sun.star.awt.Key`)
- **KeyChar (String)** – carattere immesso (tenendo conto dei tasti di modifica)

L'esempio seguente utilizza la proprietà `KeyCode` per stabilire se è stato premuto il tasto Invio, il tasto Tab o un altro tasto di controllo. Se è stato premuto uno di questi tasti, viene restituito il nome del tasto, altrimenti viene restituito il carattere immesso:

```
Sub KeyPressed(Event As Object)
   Dim Msg As String
   Select Case Event.KeyCode
   Case com.sun.star.awt.Key.RETURN
      Msg = "Premuto Invio"
   Case com.sun.star.awt.Key.TAB
      Msg = "Premuto Tab"
   Case com.sun.star.awt.Key.DELETE
      Msg = "Premuto Canc"
   Case com.sun.star.awt.Key.ESCAPE
      Msg = "Premuto Esc"
   Case com.sun.star.awt.Key.DOWN
      Msg = "Premuta freccia giù"
   Case com.sun.star.awt.Key.UP
      Msg = "Premuta freccia su"
   Case com.sun.star.awt.Key.LEFT
      Msg = "Premuta freccia sinistra"
   Case com.sun.star.awt.Key.RIGHT
      Msg = "Premuta freccia destra"
   Case Else
      Msg = "Carattere " & Event.KeyChar & " immesso"
   End Select
   MsgBox Msg
End Sub
```

Per reperire informazioni su altre costanti per la tastiera sono reperibili nel riferimento della API nel gruppo di costanti `com.sun.star.awt.Key`.

Eventi di attivazione

Gli eventi di attivazione indicano se un elemento di controllo riceve o perde l'attivazione. Potete utilizzare questi eventi per determinare, ad esempio, se un utente ha terminato l'elaborazione di un elemento di controllo consentendovi così di aggiornare gli altri elementi di una finestra di dialogo. Sono disponibili i seguenti eventi di attivazione:

- **When receiving focus** – l'elemento riceve l'attivazione
- **When losing focus** – l'elemento perde l'attivazione

Gli oggetti `Event` per gli eventi di attivazione sono strutturati come segue:

- **FocusFlags (short)** – causa della variazione di attivazione (valore predefinito conforme a `com.sun.star.awt.FocusChangeReason`).
- **NextFocus (Object)** – oggetto che riceve l'attivazione (solo per l'evento `When losing focus`)
- **Temporary (Boolean)** – l'attivazione è temporaneamente persa

Eventi specifici degli elementi di controllo

Oltre agli eventi sopracitati, supportati da tutti gli elementi di controllo, esistono anche alcuni eventi specifici definiti unicamente per alcuni elementi di controllo, tra i quali vengono riportati di seguito i più importanti:

- **When Item Changed** – il valore di un elemento di controllo cambia
- **Item Status Changed** – lo stato di un elemento di controllo cambia
- **Text modified** – il testo di un elemento di controllo cambia
- **When initiating** – un'azione che viene eseguita quando è attivato l'elemento di controllo (ad esempio, è premuto un pulsante)

Per lavorare in modo ottimale con gli eventi, si noti che alcuni eventi, come l'evento `When initiating`, possono essere iniziati ogni volta che si fa clic con il mouse su alcuni elementi di controllo (ad esempio, sui pulsanti di scelta). Nessuna azione viene eseguita per controllare se lo stato dell'elemento di controllo è effettivamente variato. Per evitare questi "eventi ciechi", salvate il precedente valore dell'elemento di controllo in una variabile globale, quindi verificate che il valore sia variato quando è in esecuzione un evento.

Le proprietà dell'evento `Item Status Changed` sono le seguenti:

- **Selected (long)** – voce attualmente selezionata
- **Highlighted (long)** – voce attualmente evidenziata
- **ItemId (long)** – ID della voce

Elementi di controllo delle finestre di dialogo in dettaglio

StarOffice Basic riconosce una serie di elementi di controllo che si possono suddividere nei seguenti gruppi:

Campi di immissione:

- campi di testo
- campi per la data
- campi per l'ora
- campi numerici
- campi di valuta
- campi che adottano qualsiasi formato

Pulsanti:

- pulsanti standard

- Caselle di controllo
- pulsanti di scelta

Caselle di selezione:

- caselle di riepilogo
- caselle combinate

Altri elementi di controllo:

- barre di scorrimento (orizzontali e verticali)
- campi di gruppi
- barre di avanzamento
- linee di divisione (orizzontali e verticali)
- Immagini
- campi di selezione file

I più importanti tra questi elementi di controllo sono presentati in maggior dettaglio di seguito.

Pulsanti

Un pulsante esegue un'azione quando si fa clic su di esso.

Lo scenario più semplice è quello dell'attivazione di un evento `When Initiating` alla pressione del pulsante da parte dell'utente. È inoltre possibile collegare un'altra azione al pulsante per aprire una finestra di dialogo utilizzando la proprietà `PushButtonType`. Quando si fa clic su un pulsante per il quale questa proprietà è stata impostata sul valore 0, la finestra di dialogo rimarrà invariata. Se si fa clic su un pulsante per il quale questa proprietà è stata impostata sul valore 1, la finestra di dialogo viene chiusa e il metodo `Execute` della finestra di dialogo restituisce il valore 1 (la sequenza è stata terminata correttamente). Se il valore di `PushButtonType` è 2, la finestra di dialogo viene chiusa e il metodo `Execute` della finestra di dialogo restituisce 0 (finestra di dialogo chiusa).

Di seguito sono riportate tutte le proprietà disponibili tramite il modello del pulsante:

- **Model.BackgroundColor (long)** – colore dello sfondo
- **Model.DefaultButton (Boolean)** – il pulsante è utilizzato come valore predefinito e risponde al tasto Invio se non è attivato.
- **Model.FontDescriptor (struct)** – struttura che specifica i dettagli del carattere da utilizzare (in conformità alla struttura `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)** – etichetta visualizzata sul pulsante
- **Model.Printable (Boolean)** – l'elemento di controllo può essere stampato
- **Model.TextColor (Long)** – colore del testo dell'elemento di controllo
- **Model.HelpText (String)** – testo della guida in linea che viene visualizzato quando si porta il puntatore del mouse sull'elemento di controllo

- **Model.HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente
- **PushButtonType (short)** – azione collegata al pulsante (0: nessuna azione, 1: OK, 2: Annulla)

Pulsanti di scelta

Questi pulsanti sono generalmente utilizzati in gruppi e permettono di selezionare un'opzione da una serie. Quando si seleziona un'opzione, tutte le altre opzioni del gruppo vengono disattivate. Ciò garantisce che in ogni momento, sia impostato un solo pulsante di scelta.

L'elemento di controllo di un pulsante di scelta fornisce due proprietà:

- **State (Boolean)** – attiva il pulsante
- **Label (String)** – etichetta visualizzata sul pulsante

Potete utilizzare le seguenti proprietà del modello dei pulsanti di scelta:

- **Model.FontDescriptor (struct)** – struttura che specifica i dettagli del carattere da utilizzare (in conformità alla struttura `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)** – etichetta visualizzata sull'elemento di controllo
- **Model.Printable (Boolean)** – l'elemento di controllo può essere stampato
- **Model.State (Short)** – se questa proprietà è uguale a 1, l'opzione è attivata, altrimenti è disattivata
- **Model.TextColor (Long)** – colore del testo dell'elemento di controllo
- **Model.HelpText (String)** – testo della guida in linea che viene visualizzato quando il puntatore del mouse si trova sull'elemento di controllo
- **Model.HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente

Per combinare diversi pulsanti di scelta in un gruppo, occorre posizionarli uno dopo l'altro nella sequenza di attivazione senza spazi (proprietà `Model.TabIndex`, descritta come Sequenza nel Dialog Editor). Se la sequenza di attivazione viene interrotta da un altro elemento di controllo, StarOffice è avviato automaticamente con un nuovo gruppo di elementi di controllo attivabile indipendentemente dal primo gruppo di elementi di controllo.

Nota – Diversamente che in VBA, in StarOffice Basic non è possibile inserire i pulsanti di scelta in un gruppo di elementi di controllo. Il raggruppamento degli elementi di controllo è utilizzato in StarOffice Basic solo per garantire una divisione visiva disegnando una cornice intorno agli elementi di controllo.

Caselle di controllo

Le caselle di controllo vengono utilizzate per registrare un valore Sì o No e, a seconda della modalità, possono adottare due o tre stati. Oltre agli stati Sì e No, una casella di controllo possono disporre di uno stato intermedio se lo stato Sì o No corrispondente ha più di un significato o non è chiaro.

Le caselle di controllo forniscono le proprietà seguenti:

- **State (Short)** – stato della casella di controllo (0: no, 1: sì, 2: stato intermedio)
- **Label (String)** – etichetta dell'elemento di controllo
- **enableTriState (Boolean)** – oltre agli stati attivati e disattivati, potete utilizzare anche lo stato intermedio

L'oggetto modello di una casella di controllo fornisce le seguenti proprietà:

- **Model.FontDescriptor (struct)** – struttura che specifica i dettagli del carattere da utilizzare (in conformità alla struttura `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)** – etichetta dell'elemento di controllo
- **Model.Printable (Boolean)** – l'elemento di controllo può essere stampato
- **Model.State (Short)** – stato della casella di controllo (0: no, 1: sì, 2: stato intermedio)
- **Model.Tabstop (Boolean)** – consente di raggiungere l'elemento di controllo con il tasto Tab
- **Model.TextColor (Long)** – colore del testo dell'elemento di controllo
- **Model.HelpText (String)** – testo della guida in linea che viene visualizzato quando si posiziona il puntatore del mouse sull'elemento di controllo
- **Model.HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente

Campi di testo

I campi di testo permettono agli utenti di inserire numeri e testo. Il servizio `com.sun.star.awt.UnoControlEdit` forma la base per i campi di testo.

Un campo di testo può contenere una o più righe e può essere modificato o bloccato per le immissioni degli utenti. I campi di testo si possono utilizzare anche come campi numerici e di valuta speciali nonché per attività speciali. Poiché questi elementi di controllo sono basati sul servizio `UnoControlEdit` Uno, la loro gestione controllata da programma è analoga.

I campi di testo forniscono le proprietà seguenti:

- **Text (String)** – testo attivo
- **SelectedText (String)** – testo attualmente evidenziato

- **Selection (Struct)** – evidenziazione in sola lettura dei dettagli (struttura conforme a `com.sun.star.awt.Selection`, con le proprietà `Min` e `Max` per specificare l'inizio e la fine dell'evidenziazione)
- **MaxTextLen (short)** – numero massimo di caratteri che si possono inserire nel campo
- **Editable (Boolean)** – `True` attiva l'opzione per l'inserimento del testo, `False` blocca l'opzione di inserimento (la proprietà non può essere richiamata direttamente, ma solo tramite `IsEditable`)
- **IsEditable (Boolean)** – il contenuto dell'elemento di controllo può essere modificato, in sola lettura.

Inoltre, sono fornite le seguenti proprietà tramite l'oggetto del modello associato:

- **Model.Align (short)** – orientamento del testo (0: allineato a sinistra, 1: centrato, 2: allineato a destra)
- **Model.BackgroundColor (long)** – colore di sfondo dell'elemento di controllo
- **Model.Border (short)** – tipo di bordo (0: nessun bordo, 1: bordo 3D, 2: bordo semplice)
- **Model.EchoChar (String)** – carattere eco per i campi delle password
- **Model.FontDescriptor (struct)** – struttura che specifica i dettagli del carattere da utilizzare (in conformità alla struttura `com.sun.star.awt.FontDescriptor`)
- **Model.HardLineBreaks (Boolean)** – interruzioni di riga automatiche inserite in modo permanente nel testo dell'elemento di controllo
- **Model.HScroll (Boolean)** – il testo ha una barra di scorrimento orizzontale
- **Model.MaxTextLen (Short)** – lunghezza massima del testo, dove 0 corrisponde a nessun limite di lunghezza
- **Model.MultiLine (Boolean)** – consente alla voce di occupare diverse righe
- **Model.Printable (Boolean)** – l'elemento di controllo può essere stampato
- **Model.ReadOnly (Boolean)** – il contenuto dell'elemento di controllo è in sola lettura
- **Model.Tabstop (Boolean)** – consente di raggiungere l'elemento di controllo con il tasto Tab
- **Model.Text (String)** – testo associato all'elemento di controllo
- **Model.TextColor (Long)** – colore del testo dell'elemento di controllo
- **Model.VScroll (Boolean)** – il testo ha una barra di scorrimento verticale
- **Model.HelpText (String)** – testo della guida in linea che viene visualizzato quando il puntatore del mouse si trova sull'elemento di controllo
- **Model.HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente

Caselle di riepilogo

Le caselle di riepilogo (servizio `com.sun.star.awt.UnoControlListBox`) supportano le seguenti proprietà:

- **ItemCount (Short)** – numero di elementi, sola lettura
- **SelectedItem (String)** – testo della voce evidenziata, sola lettura
- **SelectedItems (Array Of Strings)** – campo di dati con voci evidenziate, sola lettura
- **SelectedItemPos (Short)** – numero della voce attualmente evidenziata, sola lettura
- **SelectedItemsPos (Array of Short)** – campo di dati con il numero di voci evidenziate (per gli elenchi che supportano la selezione multipla), sola lettura
- **MultipleMode (Boolean)** – `True` attiva l'opzione per la selezione di più elementi, `False` blocca la selezione multipla (la proprietà non può essere richiamata direttamente, ma solo tramite `IsMultipleMode`)
- **IsMultipleMode (Boolean)** – consente la selezione multipla all'interno degli elenchi, sola lettura

Le caselle di riepilogo forniscono i metodi seguenti:

- **addItem (Item, Pos)** – inserisce la stringa specificata in `Item` nell'elenco, nella posizione `Pos`
- **addItems (ItemArray, Pos)** – inserisce le voci elencate nel campo dati `ItemArray` della stringa nell'elenco nella posizione `Pos`
- **removeItems (Pos, Count)** – rimuove le voci `Count` della posizione `Pos`
- **selectItem (Item, SelectMode)** – attiva o disattiva l'evidenziazione dell'elemento specificato nella stringa `Item` in base alla variabile booleana `SelectMode`
- **makeVisible (Pos)** – scorre il campo in modo che la voce specificata con `Pos` sia visibile

L'oggetto modello delle caselle di riepilogo fornisce le seguenti proprietà:

- **Model.BackgroundColor (long)** – colore di sfondo dell'elemento di controllo
- **Model.Border (short)** – tipo di bordo (0: nessun bordo, 1: bordo 3D, 2: bordo semplice)
- **Model.FontDescriptor (struct)** – struttura che specifica i dettagli del carattere da utilizzare (in conformità alla struttura `com.sun.star.awt.FontDescriptor`)
- **Model.LineCount (Short)** – numero di righe nell'elemento di controllo
- **Model.MultiSelection (Boolean)** – consente la selezione multipla delle voci
- **Model.SelectedItems (Array of Strings)** – elenco delle voci evidenziate
- **Model.StringItemList (Array of Strings)** – elenco di tutte le voci
- **Model.Printable (Boolean)** – l'elemento di controllo può essere stampato
- **Model.ReadOnly (Boolean)** – il contenuto dell'elemento di controllo è in sola lettura

- **Model.Tabstop (Boolean)** – consente di raggiungere l'elemento di controllo con il tasto Tab.
- **Model.TextColor (Long)** – colore del testo dell'elemento di controllo
- **Model.HelpText (String)** – visualizza automaticamente il testo della guida in linea che compare se il puntatore del mouse si trova sopra l'elemento di controllo
- **Model.HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente

---

Nota – L'opzione di VBA per l'emissione delle voci di elenco con un valore numerico addizionale (`ItemData`) non esiste in StarOffice Basic. Per gestire un valore numerico (ad esempio un ID di database) oltre al testo in linguaggio naturale, occorre creare un campo di dati ausiliario di gestione parallela alla casella di riepilogo.

CAPITOLO **12**

Formulari

Per molti aspetti, la struttura dei formulari di StarOffice corrisponde agli elementi dialog presentati nel capitolo precedente. Sussistono tuttavia alcune differenze chiave:

- Gli elementi dialog assumono la forma di una singola finestra di dialogo, che viene visualizzata sul documento e non consente altre azioni oltre all'elaborazione della finestra di dialogo fino al suo termine. I formulari, d'altro canto, sono visualizzati direttamente nel documento, proprio come gli elementi di disegno.
- È disponibile un Dialog Editor per la creazione delle finestre di dialogo disponibile nell'ambiente di sviluppo StarOffice Basic. I formulari sono creati utilizzando la **barra dei simboli Funzioni formulario** direttamente all'interno del documento.
- Le funzioni dialog sono disponibili in tutti i documenti di StarOffice, ma la gamma completa delle funzioni formulario è disponibile solo per testi e fogli elettronici.
- Gli elementi di controllo di un formulario possono essere collegati a una tabella di database esterna. Questa funzione non è disponibile nelle finestre di dialogo.
- Gli elementi di controllo di finestre di dialogo e formulari differiscono per diversi aspetti.

Agli utenti che desiderano inserire nei propri formulari metodi personalizzati di gestione degli eventi si consiglia la consultazione del capitolo 11, Finestre di dialogo. I meccanismi descritti di seguito sono identici a quelli per i formulari.

Uso dei formulari

I formulari di StarOffice possono contenere campi di testo, caselle di riepilogo, pulsanti di scelta e una serie di altri elementi di controllo, che vengono inseriti direttamente in un testo o in un foglio elettronico. Per la modifica dei formulari viene utilizzata la **barra dei simboli Funzioni formulario**.

Un formulario di StarOffice può usufruire di una delle due modalità seguenti: bozza e visualizzazione. Nel primo caso, la posizione degli elementi di controllo può essere modificata e le relative proprietà modificate utilizzando una finestra delle proprietà.

Per passare da una modalità all'altra potete utilizzare anche la **barra dei simboli Funzioni formulario**.

Determinazione degli oggetti formulario

StarOffice posiziona gli elementi di controllo di un formulario al livello degli oggetti di disegno. Si accede all'oggetto formulario dall'elenco Forms al livello disegno. Nei documenti di testo si accede agli oggetti nel modo seguente:

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Il metodo `GetByIndex` restituisce il formulario con il numero d'indice 0.

Quando si lavora con i fogli elettronici, è necessaria una fase intermedia tramite l'elenco Sheets perché i livelli di disegno non sono situati direttamente nel documento, ma nei singoli fogli:

```
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)
```

Come suggerito dal nome del metodo `GetByIndex`, un documento può contenere diversi formulari. Ciò risulta utile, ad esempio, se all'interno di un unico documento vengono visualizzati i contenuti di diversi database o se è visualizzata in un formulario una relazione di database 1:n. A tal fine è disponibile anche l'opzione di creare formulari secondari.

I tre aspetti degli elementi di controllo di un formulario

Gli elementi di controllo di un formulario sono composti da tre aspetti:

- In primo luogo, c'è il *modello* dell'elemento di controllo. Si tratta dell'oggetto chiave per il programmatore in StarOffice Basic che opera con gli elementi di controllo.
- La sua controparte è la *vista* dell'elemento di controllo, che gestisce le informazioni da visualizzare.
- Dato che gli elementi di controllo nei documenti vengono gestiti come un elemento di disegno di tipo speciale, esiste anche un *oggetto forma* che riflette le proprietà specifiche dell'elemento disegno dell'elemento di controllo (in particolare, la sua posizione e le dimensioni).

Accesso al modello degli elementi di controllo

I modelli degli elementi di controllo di un formulario sono disponibili tramite il metodo `GetByName` dell'Object form:

```
Dim Doc As Object
Dim Form As Object
Dim Ctl As Object

Doc = StarDesktop.CurrentComponent
Form = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Form.getByName("MyListBox")
```

L'esempio determina il modello dell'elemento di controllo `MyListBox`, situato nel primo formulario del documento di testo attualmente aperto.

Se non si è certi dell'elemento di controllo, utilizzate l'opzione di ricerca in tutti i formulari per individuare l'elemento di controllo richiesto:

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      Exit Function
   End If
Next I
```

L'esempio utilizza il metodo `HasByName` per controllare tutti i formulari di un documento di testo per determinare se contengono un modello di elemento di controllo denominato `MyListBox`. Se viene individuato un modello corrispondente, il riferimento a questo modello viene salvato nella variabile `Ctl` e la ricerca è terminata.

Accesso alla vista degli elementi di controllo

Per accedere alla vista degli elementi di controllo, è necessario prima disporre del modello associato. La vista dell'elemento di controllo può essere determinata con l'ausilio del modello e utilizzando il controller del documento.

```
Dim Doc As Object
Dim DocCrl As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim CtlView As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
DocCrl = Doc.getCurrentControler()
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      CtlView = DocCrl.GetControl(Ctl)
      Exit Function
   End If
Next I
```

Il codice riportato nell'esempio è molto simile al codice dell'esempio precedente per determinare un modello di elemento di controllo. Utilizza non solo l'oggetto documento `Doc` ma anche l'oggetto controller `DocCrl` del documento che fa riferimento alla finestra del documento corrente. Con l'ausilio di questo oggetto controller e del modello dell'elemento di controllo, utilizza il metodo `GetControl` per determinare la vista (variabile `CtlView`) degli elementi di controllo.

Accesso all'oggetto shape degli elementi di controllo

Il metodo per accedere agli oggetti shape di un elemento di controllo si avvale anche del livello disegno corrispondente del documento. Per determinare uno speciale elemento di controllo, occorre eseguire una ricerca tra tutti gli elementi del livello disegno.

```
Dim Doc As Object
Dim Shape as Object
Dim I as integer

Doc = StarDesktop.CurrentComponent

For i = 0 to Doc.DrawPage.Count - 1
   Shape = Doc.DrawPage(i)
```

```
   If HasUnoInterfaces(Shape, _
      "com.sun.star.drawing.XControlShape") Then
      If Shape.Control.Name = "MyListBox" Then
            Exit Function
      End If
   End If
Next
```

L'esempio controlla tutti gli elementi di disegno per determinare se supportano l'interfaccia `com.sun.star.drawing.XControlShape` necessaria per gli elementi di controllo. In caso positivo, la proprietà `Control.Name` controlla se il nome dell'elemento di controllo è `MyListBox`. Se sì, la funzione termina la ricerca.

Determinazione delle dimensioni e della posizione degli elementi di controllo

Come menzionato in precedenza, è possibile determinare le dimensioni e la posizione degli elementi di controllo utilizzando l'oggetto `shape`. La forma dell'elemento di controllo, come molti altri oggetti `shape`, fornisce a tal fine le proprietà `Size` e `Position`:

- **Size (struct)** – dimensioni dell'elemento di controllo (struttura di dati `com.sun.star.awt.Size`).
- **Position (struct)** – posizione dell'elemento di controllo (struttura di dati `com.sun.star.awt.Point`).

L'esempio seguente mostra come impostare la posizione e le dimensioni di un elemento di controllo utilizzando l'oggetto shape associato:

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

Perché il codice funzioni, l'oggetto `shape` dell'elemento di controllo deve già essere noto. In caso contrario, dovrete determinarlo utilizzando il codice di cui sopra.

Informazioni dettagliate sugli elementi di controllo disponibili nei formulari

Gli elementi di controllo disponibili nei formulari sono simili a quelli delle finestre di dialogo. La selezione va dai semplici campi di testo, alle caselle di riepilogo e combinate fino a diversi pulsanti.

Di seguito viene riportato un elenco delle più importanti proprietà per questi elementi di controllo. Tutte le proprietà fanno parte degli oggetti modello associati.

Oltre agli elementi di controllo standard, per i formulari è disponibile anche un elemento di controllo delle tabelle, che permette di incorporare tabelle di database. Questo argomento è descritto nella sezione "Formulari basati su database" a pagina 225 nel Capitolo 12.

Pulsanti

L'oggetto modello di un pulsante di formulario fornisce le seguenti proprietà:

- **BackgroundColor (long)** – colore di sfondo.
- **DefaultButton (Boolean)** – il pulsante funge da valore predefinito. In questo caso, risponde anche al pulsante di immissione se non è attivato.
- **Enabled (Boolean)** – l'elemento di controllo può essere attivato.
- **Tabstop (Boolean)** – l'elemento di controllo può essere raggiunto tramite il pulsante di tabulazione.
- **TabIndex (Long)** – posizione dell'elemento di controllo nella sequenza di attivazione.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **Tag (String)** – stringa contenente informazioni addizionali, che possono essere salvate nel pulsante per l'accesso controllato dal programma.
- **TargetURL (String)** – URL di destinazione per i pulsanti del tipo URL.
- **TargetFrame (String)** – nome della finestra (o cornice) in cui aprire `TargetURL` all'attivazione del pulsante (per i pulsanti del tipo `URL`).
- **Label (String)** – etichetta del pulsante.
- **TextColor (Long)** – colore del testo dell'elemento di controllo.
- **HelpText (String)** – testo della guida in linea visualizzato automaticamente se il cursore del mouse si trova sull'elemento di controllo.
- **HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente.

- **ButtonType (Enum)** – azione collegata al pulsante (valore predefinito di `com.sun.star.form.FormButtonType`).

Tramite la proprietà `ButtonType` si ha l'opportunità di definire un'azione che viene eseguita automaticamente alla pressione del pulsante. Il gruppo di costanti `com.sun.star.form.FormButtonType` fornisce i seguenti valori:

- **PUSH** – pulsante standard.
- **SUBMIT** – fine dell'immissione del formulario (di particolare rilevanza per i formulari HTML).
- **RESET** – ripristina sui valori originali tutti i valori nel formulario.
- **URL** – chiamata dell'URL definito in `TargetURL` (è aperta nella finestra specificata tramite `TargetFrame`).

I tipi di pulsanti **OK** e **Annulla** forniti nelle finestre di dialogo non sono supportati nei formulari.

Pulsanti di scelta

Le seguenti proprietà dei pulsanti di scelta sono disponibili tramite l'oggetto del modello:

- **Enabled (Boolean)** – l'elemento di controllo può essere attivato.
- **Tabstop (Boolean)** – l'elemento di controllo può essere raggiunto tramite il tasto di tabulazione.
- **TabIndex (Long)** – posizione dell'elemento di controllo nella sequenza di attivazione.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **Tag (String)** – stringa contenente informazioni addizionali, che possono essere salvate nel pulsante per l'accesso controllato dal programma.
- **Label (String)** – etichetta del pulsante.
- **Printable (Boolean)** – l'elemento di controllo può essere stampato.
- **State (Short)** – se questa proprietà è uguale a 1, l'opzione è attivata, altrimenti è disattivata.
- **RefValue (String)** – stringa per il salvataggio di informazioni addizionali (ad esempio, per la gestione degli ID dei record di dati).
- **TextColor (Long)** – colore del testo dell'elemento di controllo.
- **HelpText (String)** – testo della guida in linea visualizzato automaticamente se il cursore del mouse si trova sull'elemento di controllo.
- **HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente.

Il meccanismo di raggruppamento dei pulsanti di scelta distingue tra gli elementi di controllo delle finestre di dialogo e quelli dei formulari. Mentre gli elementi di controllo che compaiono uno dopo l'altro nelle finestre di dialogo vengono combinati automaticamente per formare un gruppo, il raggruppamento nei formulari avviene basandosi sui nomi. Per procedere in tal senso, tutti i pulsanti di scelta di un gruppo devono contenere lo stesso nome. StarOffice combina gli elementi di controllo raggruppati in un array in modo che i singoli pulsanti di un programma in StarOffice Basic si possano raggiungere secondo le modalità utilizzate in precedenza.

L'esempio seguente mostra come determinare il modello di un gruppo di elementi di controllo.

```
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("Opzioni") Then
      Ctl = Form. GetGroupbyName("Opzioni")
      Exit Function
   End If
Next I
```

Il codice corrisponde all'esempio precedente per determinare un semplice modello di elemento di controllo. Esegue una ricerca in tutti i formulari nel documento di testo corrente in un ciclo e utilizza il metodo `HasByName` per controllare se il formulario corrispondente contiene un elemento con il nome ricercato. Qualora lo individui, accedere all'array del modello utilizzando il metodo `GetGroupByName` (al posto del metodo `GetByName` utilizzato per determinare i modelli semplici).

Caselle di controllo

L'oggetto modello di una casella di controllo fornisce le seguenti proprietà:

- **Enabled (Boolean)** – l'elemento di controllo può essere attivato.
- **Tabstop (Boolean)** – l'elemento di controllo può essere raggiunto tramite il tasto di tabulazione.
- **TabIndex (Long)** – posizione dell'elemento di controllo nella sequenza di attivazione.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **Tag (String)** – stringa contenente informazioni addizionali, che possono essere salvate nel pulsante per l'accesso controllato dal programma.

- **Label (String)** – etichetta del pulsante.
- **Printable (Boolean)** – l'elemento di controllo può essere stampato.
- **State (Short)** – se questa proprietà è uguale a 1, l'opzione è attivata, altrimenti è disattivata.
- **RefValue (String)** – stringa per il salvataggio di informazioni addizionali (ad esempio, per la gestione degli ID dei record di dati).
- **TextColor (Long)** – colore del testo dell'elemento di controllo.
- **HelpText (String)** – testo della guida in linea visualizzato automaticamente se il cursore del mouse si trova sull'elemento di controllo.
- **HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente.

Campi di testo

L'oggetto modello di un campo di testo fornisce le seguenti proprietà:

- **Align (short)** – orientamento del testo (0: allineato a sinistra, 1: centrato, 2: allineato a destra)
- **BackgroundColor (Long)** – colore di sfondo dell'elemento di controllo.
- **Border (short)** – tipo di bordo (0: nessun bordo, 1: bordo 3D, 2: bordo semplice)
- **EchoChar (String)** – carattere eco per i campi delle password.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **HardLineBreaks (Boolean)** – interruzioni di riga automatiche inserite in modo permanente nel testo dell'elemento di controllo.
- **HScroll (Boolean)** – il testo ha una barra di scorrimento orizzontale.
- **MaxTextLen (Short)** – lunghezza massima del testo; se è specificato 0, non sono presenti limiti.
- **MultiLine (Boolean)** – consente voci a più righe.
- **Printable (Boolean)** – l'elemento di controllo può essere stampato.
- **ReadOnly (Boolean)** – il contenuto dell'elemento di controllo è in sola lettura.
- **Enabled (Boolean)** – l'elemento di controllo può essere attivato.
- **Tabstop (Boolean)** – l'elemento di controllo può essere raggiunto tramite il tasto di tabulazione.
- **TabIndex (Long)** – posizione dell'elemento di controllo nella sequenza di attivazione.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **Text (String)** – testo dell'elemento di controllo.

- **TextColor (Long)** – colore del testo dell'elemento di controllo.
- **VScroll (Boolean)** – il testo ha una barra di scorrimento verticale.
- **HelpText (String)** – testo della guida in linea visualizzato automaticamente se il cursore del mouse si trova sull'elemento di controllo.
- **HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente.

Caselle di riepilogo

L'oggetto modello delle caselle di riepilogo fornisce le seguenti proprietà:

- **BackgroundColor (Long)** – colore di sfondo dell'elemento di controllo.
- **Border (short)** – tipo di bordo (0: nessun bordo, 1: bordo 3D, 2: bordo semplice)
- **FontDescriptor (struct)** – struttura che specifica i dettagli del carattere utilizzato (in conformità alla struttura `com.sun.star.awt.FontDescriptor`).
- **LineCount (Short)** – numero di righe nell'elemento di controllo.
- **MultiSelection (Boolean)** – consente la selezione multipla delle voci.
- **SelectedItems (Array of Strings)** – elenco delle voci evidenziate.
- **StringItemList (Array of Strings)** – elenco di tutte le voci.
- **ValueItemList (Array of Variant)** – elenco contenente informazioni addizionali per ogni voce (ad esempio, per la gestione degli ID dei record di dati).
- **Printable (Boolean)** – l'elemento di controllo può essere stampato.
- **ReadOnly (Boolean)** – il contenuto dell'elemento di controllo è in sola lettura.
- **Enabled (Boolean)** – l'elemento di controllo può essere attivato.
- **Tabstop (Boolean)** – l'elemento di controllo può essere raggiunto tramite il tasto di tabulazione.
- **TabIndex (Long)** – posizione dell'elemento di controllo nella sequenza di attivazione.
- **FontName (String)** – nome del tipo di carattere.
- **FontHeight (Single)** – altezza del carattere in punti (pt).
- **Tag (String)** – stringa contenente informazioni addizionali, che possono essere salvate nel pulsante per l'accesso controllato dal programma.
- **TextColor (Long)** – colore del testo dell'elemento di controllo.
- **HelpText (String)** – testo della guida in linea visualizzato automaticamente se il cursore del mouse si trova sull'elemento di controllo.
- **HelpURL (String)** – URL della guida in linea per l'elemento di controllo corrispondente.

Nota – Attraverso la proprietà `ValueItemList`, i formulari con caselle di riepilogo forniscono una controparte alla proprietà `ItemData` di VBA attraverso la quale gestire informazioni addizionali per le singole voci dell'elenco.

Inoltre, sono disponibili i metodi seguenti tramite l'oggetto vista della casella di riepilogo:

- **addItem (Item, Pos)** – inserisce la stringa specificata in `Item` nell'elenco, nella posizione `Pos`.
- **addItems (ItemArray, Pos)** – inserisce le voci elencate nel campo dati `ItemArray` della stringa nell'elenco nella posizione `Pos`
- **removeItems (Pos, Count)** – rimuove le voci `Count` della posizione `Pos`.
- **selectItem (Item, SelectMode)** – attiva o disattiva l'evidenziazione dell'elemento specificato nella stringa `Item` in base alla variabile `SelectMode`.
- **makeVisible (Pos)** – scorre il campo in modo che la voce specificata con `Pos` sia visibile.

Formulari basati su database

I formulari di StarOffice possono essere collegati direttamente a un database. I formulari creati in questo modo offrono tutte le funzioni del frontend di un database completo senza richiedere lavoro di programmazione extra.

L'utente ha l'opzione di ricercare le tabelle selezionate e le ricerche nonché di modificare i record di dati e inserirne di nuovi. StarOffice garantisce automaticamente che i dati pertinenti siano recuperati dal database e che le modifiche effettuate siano riscritte nel database.

Un formulario basato su database corrisponde di base a un formulario standard di StarOffice. Oltre alle proprietà standard, le seguenti proprietà specifiche di database devono essere anche impostate nel formulario:

- **DataSourceName (String)** – nome della sorgente di dati (consultare il Capitolo 10; la sorgente dei dati deve essere creata globalmente in StarOffice).
- **Command (String)** – nome di tabella, ricerca, o comando di selezione SQL al quale eseguire un collegamento.
- **CommandType (Const)** – specifica se Command è una tabella, una ricerca o un comando SQL (valore dell'enumerazione `com.sun.star.sdb.CommandType`).

L'enumerazione `com.sun.star.sdb.CommandType` copre i seguenti valori:

- **TABLE** - Tabella
- **QUERY** - Ricerca
- **COMMAND** - Comando SQL

I campi del database sono assegnati ai singoli elementi di controllo tramite questa proprietà:

- **DataField (String)** – nome del campo di database collegato.

Tabelle

È fornito un altro elemento di controllo per lavorare con i database: l'elemento di controllo per le tabelle. Ciò rappresenta il contenuto di una tabella di database completa o ricerca. Nello scenario più semplice, un elemento di controllo per tabelle è collegato a un database utilizzando il formulario pilota automatico, che collega tutte le colonne con i campi pertinenti del database in conformità alle specifiche dell'utente. Poiché la API associata è relativamente complessa, non viene tuttavia fornita in questa fase una descrizione completa della API.

Indice analitico

D

E

F

G

H

I

J

K

L

M

R

S

T

U

V

W

X

Y