StarSuite™ 7 Office Suite

A Sun™ ONE Software Offering

Basic 程式設計手冊

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

部件編號817-3928-10
2003 年 , 第一次修訂

Copyrights and Trademarks

本使用手册内容

第 1 章

介紹

本手冊介紹如何使用 StarSuite Basic 6.0 進行程式設計，並指示使用 StarSuite 中的 StarSuite Basic 可以得到哪些可能的應用程式。若要充分利用本手冊，還應熟悉其他程式設計語言。

本手冊提供的大量示例可協助您快速開發自己的 StarSuite Basic 程式。

本手冊爲 Microsoft Visual Basic 程式設計師或使用過 StarSuite Basic 早期版本的程式設計師提供了許多遷移提示，它們由頁面邊緣處的一個小圖示指示。本手冊的〈附錄〉包含所有遷移提示的索引，便於您快速瀏覽至想要閱讀的提示。

關於 StarSuite Basic

StarSuite Basic 程式設計語言是專門爲 StarSuite 而開發的，並且已嚴密整合至 Office 套裝軟體中。

顧名思義，StarSuite Basic 是一種 Basic 系列的程式設計語言。任何先前使用過其他 Basic 語言 (尤其是 Microsoft 的 Visual Basic 或 Visual Basic for Applications (VBA)) 的人，將會很快習慣使用 StarSuite Basic。StarSuite Basic 大部份基本結構與 Visual Basic 相容。

StarSuite Basic 程式設計語言可分爲四個程式元件：

StarSuite Basic 語言：定義基本語言結構，例如，變數宣告、迴路和函數。

執行期間程式庫：提供與 StarSuite 無直接參照的標準函數，例如，用於編輯數字、字串、日期值和檔案的函數。

StarSuite API (應用程式設計介面**)**：允許存取 StarSuite 文件，並允許建立、儲存、修改和列印這些文件。

對話方塊編輯器：建立個人對話方塊視窗，並可加入控制項元素和事件處理程式。

StarSuite Basic 與 VBA 之間的相容性與 StarSuite Basic 語言和執行期間程式庫有關。StarSuite API 和對話方塊編輯器與 VBA 不相容 (標準化這些介面將無法實現 StarSuite 中提供的許多概念)。

StarSuite Basic 的預期使用者

StarSuite 標準函數無能爲力的場合正是 StarSuite Basic 的作用所在。因此，可以使用 StarSuite Basic 自動執行日常工作、建立至其他程式的捷徑 (例如，至資料庫伺服器的捷徑) 以及按下按鈕執行複雜活動 (使用預先定義的程式檔)。

StarSuite Basic 提供對 StarSuite 所有功能的完全存取、支援所有的函數、修改文件類型以及提供建立個人對話方塊視窗的選項。

StarSuite Basic 的使用

任一 StarSuite 的使用者無需任何其他程式和輔助工具即可使用 StarSuite Basic。
即使在標準安裝中，StarSuite Basic 也具有建立自己的 Basic 巨集所需的所有程式元件，其中包括：

整合式開發環境 (IDE)，它提供一個用於輸入和測試巨集的編輯器。

解譯程式，執行 StarSuite Basic 巨集時需要該程式。

各種 StarSuite 應用程式的介面，這些介面允許直接存取 Office 文件。

本手册的結構

前三章向讀者介紹 StarSuite Basic 中的如下內容：

第 2 章：StarSuite Basic 語言

第 3 章：StarSuite Basic 執行期間程式庫

第 4 章：StarSuite API 介紹

這三章概要介紹了 StarSuite Basic，任何想要撰寫 StarSuite Basic 程式的人都應該 閱讀這三章。

其餘各章更爲詳盡地描述了 StarSuite API 的各個程式元件，讀者可依需要有選擇地閱讀：

第 5 章：使用 StarSuite 文件

第 6 章：文字文件

第 7 章：工作表文件

第 8 章：繪圖和簡報

第 章：圖解 (圖表)

第 10 章：資料庫存取

第 11 章：對話方塊

第 12 章：表單

更多資訊

本手册中討論的 StarSuite API 程式元件是依其對 StarSuite Basic 程式設計師的實用價值而選取的。一般只討論部份介面。如需更詳細的描述，請參閱 Internet 上的 API 參照：

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

《StarSuite 開發者指南》比本手册更爲詳細地描述了 StarSuite API，但該指南主要針對 Java 和 C++ 程式設計師。已經熟悉 StarSuite Basic 程式設計的人可以在《StarSuite 開發者指南》中找到有

關 StarSuite Basic 和 StarSuite 程式設計的其他資訊。您可以從 Internet 上的以下位置下載《StarSuite 開發者指南》：

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

希望直接使用 Java 或 C++ 而不是 StarSuite Basic 的程式設計師應參閱《StarSuite 開發者指南》，而不是本手册。與使用 StarSuite Basic 進行程式設計相比，使用 Java 或 C++ 進行 StarSuite 程式設計是一個相當複雜的程序。

第 2 章

StarSuite Basic 語言

StarSuite Basic 屬於 Basic 語言系列。StarSuite Basic 中的許多部份與 Microsoft Visual Basic for Applications 和 Microsoft Visual Basic 完全相同。任何早已用過這些語言的人均可很快習慣使用 StarSuite Basic。

使用其他語言 (如 Java、C++ 或 Delphi) 的程式設計師也會發現，熟悉 StarSuite Basic 是很容易的。StarSuite Basic 是一種十分先進的程序程式設計語言，不再使用早期的控制結構，如 GoTo 和 GoSub。

由於透過 StarSuite Basic 中的介面可以使用外部物件程式庫，因此還可以利用物件導向程式設計的優點。整個 StarSuite API 就是基於這樣一些介面，稍後將更加詳細地描述這些介面。

本章概述了 StarSuite Basic 語言的關鍵元素和結構，以及 StarSuite Basic 中應用程式和程式庫的框架。

StarSuite Basic 程式概述

StarSuite Basic 是 一種解譯程式語言。與 C++ 或 Turbo Pascal 不同，StarSuite 編譯程式不建立能夠自動執行的可執行檔案或自解壓檔案。而是透過按一下某個按鈕來執行 StarSuite Basic 程式。首先會檢查程式碼，看是否有明顯錯誤，然後逐行執行。

程式行

Basic 解譯程式之行導向的工作方式是 Basic 與其他程式設計語言的主要區別之一。Basic 程式中的每一行均形成了一個自包含單元，而在 Java、C++、Delphi 等程式的來源程式碼中，可在任意位置進行硬換行。函數呼叫、數學表示式以及其他語言元素 (如函數和迴路標頭) 必須 在其起始的同一行中完成。

如果没有足夠空間，或者這樣會導致行過長，則可以透過加入底線 _ 將多個行連結起來。以下示例顯示了如何將一個數學表示式中的四行連結起來：

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

. 底線必須是所連結行的最後一個字元，而且其後不能帶有空格和定位鍵，否則程式碼就會 產生錯誤。

除了連結單行以外，StarSuite Basic 允許程式設計師使用冒號將一行分爲若干區域，這樣，就有足夠空間容納多個表示式。例如，指定

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

能以下列形式撰寫：

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

註解

除了要執行的程式碼以外，StarSuite Basic 程式還可以包含註解，用於解釋程式的各個部份並提供重要資訊，以幫助以後某個時候再現程式碼 (例如作爲疑難排解的一部份)。

StarSuite Basic 提供兩種在程式碼中插入註解的方法：

撇號後面的所有字元都會被 視爲註解：

```
Dim A    ' 這是變數 A 的註解
```

關鍵字 Rem，後跟註解。

```
Rem 此註解由關鍵字 Rem 引入。
```

一條註解通常包括向右直到行末的所有字元。然後，StarSuite Basic 又將後續行解譯爲常規指令。如果註解佔用多個行，則應將每行都標識爲註解：

```
Dim B       ' 變數 B 的此註解相當長
            ' 且跨越多個行。
            ' 因此必須在每行重複使用
            ' 註解字元。
```

記號

一個 StarSuite Basic 程式可以含有數十個、數百個甚至數千個記號 (變數、常數、函數等的名稱)。爲一個記號選取名稱時，採用以下規則：

記號只能含有拉丁字母、數字和底線 (_)。

記號的第一個字元必須是字母或底線。

記號不能包含特殊字元，例如，a a i s 等。

記號的最大長度爲 255 個字元。

不區分字元的大小寫。例如，OneTestVariable 記號既可以定義 onetestVariable 變數，也可以定義 ONETESTVARIABLE 變數。

但是，此規則有一個例外：UNO-API 常數區分字元的大小寫。如果需要有關 UNO 的更多資訊，則請參閱第 章。

StarSuite Basic 中用於建構記號的規則與 VBA 中的不同。例如，與在 VBA 中不同的是，StarSuite Basic 不允許記號中含有特殊字元，因爲特殊字元在國際專案中會造成問題。

下面是正確記號和錯誤記號的幾個示例：

```
Surname              ' 正確
Surname5             ' 正確 (數字 5 不是第一個數位)
First Name           ' 錯誤 (不允許有空格)
DejaVu               ' 錯誤 (不允許有諸如 e、a 之類的字母)
5Surnames            ' 錯誤 (第一個字元不能是數字)
First,Name           ' 錯誤 (不允許有逗號和句點)
```

使用變數

隱含的變數宣告

Basic 語言設計得易於使用。因此，StarSuite Basic 允許透過簡易用法來建立一個變數，而無需進行明確的宣告。換句話說，從將一個變數包含在程式碼中的那一刻起，該變數就已存在。以下代碼段最多宣告了三個新變數 (具體的新變數個數取決於已經存在的變數)：

```
a = b + c
```

隱含宣告變數不是一種好的形式，會導致由於鍵入錯誤等而無意中引入新的變數。對於鍵入錯誤，解譯程式不會生成錯誤訊息，而只是將其初始化成一個值爲 0 的新變數。在程式碼中，這類錯誤很難被發現。

明確的變數宣告

爲防止隱含的變數宣告而導致的錯誤，StarSuite Basic 提供了一個開關，即：

```
Option Explicit
```

該開關必須在每個模組的第一個程式行中列出，才能確保在使用未宣告的變數時發出錯誤訊息。`Option Explicit` 開關應 存在於所有 Basic 模組中。

明確的變數宣告指令之最簡單形式如下：

```
Dim MyVar
```

此示例宣告了一個變數，其名稱爲 `MyVar`，類型爲變體型。變體型變數是一種通用變數，可以記錄所有可能的值，包括字串、整數、浮點數和布林值。以下是變體型變數的幾個示例：

```
MyVar = "Hello World"               ' 指定一個字串
MyVar = 1                           ' 指定一個整數
MyVar = 1.0                         ' 指定一個浮點數
MyVar = True                        ' 指定一個布林值
```

上面宣告的變數甚至可以用於同一個程式中的不同變數類型。盡管這會提供很大的靈活性，但最好將一個變數限於一種變數類型。當 StarSuite Basic 在某個特定上下文中遇到定義錯誤的變數類型時，就會生成一個錯誤訊息。

進行限定類型的變數宣告時，請使用以下樣式：

```
Dim MyVar As Integer    ' 宣告一個整數類型的變數
```

該變數被宣告爲整數類型，可以記錄整數值。也可以使用以下樣式來宣告整數類型的變數：

```
Dim MyVar%              ' 宣告一個整數類型的變數
```

`Dim` 指令可以記錄多個變數宣告：

```
Dim MyVar1, MyVar2
```

如果要將變數指定爲某種永久類型，則必須對每個變數進行分別指定：

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

如果未宣告變數的類型，則 StarSuite Basic 會將該變數指定爲變體類型。例如，在下面的變數宣告中，`MyVar1` 是一個變體類型，而 `MyVar2` 則是一個整數類型：

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

以下幾節列出了 StarSuite Basic 中可以使用的變數類型，並描述了如何使用和宣告這些變數類型。

字串

字串和數字是 StarSuite Basic 中最重要的基本類型。字串由一個連續的單個字元序列組成。電腦在內部將字串儲存爲一個數字序列，每個數字代表一個特定字元。

從 ASCII 字元集到統一碼

字元集將字串中的字元與一個描述電腦螢幕顯示或列印該字串表格中的相應代碼 (數字和字元) 相比對。

ASCII 字元集

ASCII 字元集是一組用單個位元組表示數字、字元、特殊符號的代碼。0 到 127 ASCII 碼對應於字母表和常用符號 (如句點、括號和逗號) 以及一些特殊的螢幕和印表機控制碼。ASCII 字元集常用作電腦之間傳送文字資料的標準格式。

但是，此字元集未包括歐洲使用的所有特殊字元 (如 a、a 和 i)，以及諸如西裡爾語字母表之類的其他字元格式。

ANSI 字元集

Microsoft 的 Windows 產品基於美國國家標準學會 (ANSI) 字元集，這種字元集逐漸擴展爲包括 ASCII 字元集中没有的字元。

代碼頁

ISO 8859 字元集提供了一個姍姍來遲的國際標準。ISO 字元集的前 128 個字元與 ASCII 字元集相對應。ISO 標準引入新的字元集 (代碼頁)，以便正確地顯示更多語言。可結果是，相同的字元值在不同的語言中可表示不同的字元。

統一碼

統一碼將一個字元的長度增加到四個位元組，並將不同的字元集組合起來，以建立一種可描述盡可能多的語言的標準。Unicode 2.0 版現在獲得許多程式的支援，其中包括 StarSuite 和 StarSuite Basic。

字串型變數

StarSuite Basic 用統一碼中的字串型變數儲存字串。一個字串型變數最多可以儲存 65535 個字元。在內部，StarSuite Basic 儲存每個字元關聯的統一碼值。一個字串型變數所需的工作記憶體取決於字串的長度。

字串型變數的宣告示例：

```
Dim Variable As String
```

也可以將此宣告撰寫爲：

```
Dim Variable$
```

移植 VBA 應用程式時，請確保遵守 StarSuite Basic 中允許的最大字串長度 (65535 個字元)。

指定明確的字串

若要將一個明確的字串指定至一個字串型變數，請將該字串放在引號 (") 中。

```
Dim MyString As String
MyString = " This is a test"
```

若要將一個字串分開使其位於兩行上，請在第一行末尾加入一個加號：

```
Dim MyString As String
MyString =    "This string is so long that it" + _
              "has been split over two lines."
```

若要在字串中包含一個引號 (")，請在相應點處輸入兩個引號：

```
Dim MyString As String
MyString = "a ""-quotation mark."   ' 生成一個 " 引號
```

數字

StarSuite Basic 支援五種處理數字的基本類型：

整型

長型整數

單精度型

雙精度型

貨幣型

整型變數

整型變數可以儲存 -32768 至 32767 之間的任何整數。一個整型變數最多可佔用兩個位元組的記憶體。整型變數的類型宣告符號爲 %。使用整型變數的計算速度非常快，這對於迴路計數器尤其有用。如果將一個浮點數指定給一個整型變數，則將該浮點數向上或向下捨入爲接近的整數。

整型變數的宣告示例：

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

長型整數變數

長型整數變數可以儲存 -2147483648 至 2147483647 之間的任何整數。長型整數變數最多可佔用四個位元組的記憶體。長型整數的類型宣告符號爲 &。使用長型整數變數的計算速度非常快，這對於迴路計數器尤其有用。如果將一個浮點數指定給一個長型整數變數，則將該浮點數向上或向下捨入爲接近的整數。

長型整數變數的宣告示例：

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

單精度型變數

單精度型變數可以儲存 3.402823 x 10^{38} 與 1.401298 x 10^{-45} 之間的任何正浮點數或負浮點數。一個單精度型變數最多可佔用四個位元組的記憶體。單精度型變數的類型宣告符號爲 !。

最初，單精度型變數用於減少爲更精確的雙精度型變數所需的計算時間。但是，由於不再考量這些速度問題，因而減小了對單精度型變數的需要。

單精度型變數的宣告示例：

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

雙精度型變數

雙精度型變數可以儲存 1.79769313486232 x 10^{308} 與 4.94065645841247 x 10^{-324} 之間的任何正浮點數或負浮點數。一個雙精度型變數最多可佔用八個位元組的記憶體。雙精度型變數適用於精確計算。其類型宣告符號爲 #。

雙精度型變數的宣告示例：

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

貨幣型變數

貨幣型變數在處理數值的方式上與其他變數類型不同。小數點固定，後面帶有四個小數位。該變數的小數點前最多可以含有 15 個數字。貨幣型變數可以儲存 -922337203685477.5808 至 +922337203685477.5807 之間的任何值，最多可佔用八個位元組的記憶體。貨幣型變數的類型宣告符號爲 @。

貨幣型變數主要用於商業計算，以避免在此類計算中由於使用浮點數而產生的無法預料的捨入誤差。

貨幣型變數的宣告示例：

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

指定明確的數字

有多種方式可用來顯示數字，例如，以十進制格式或以科學計數法，甚至使用十進制系統以外的其他基數。以下規則適用於 StarSuite Basic 中的數字字元：

整數

最簡單的方法是使用整數。這些整數在來源文字中未使用逗號分隔千位數字：

```
Dim A As Integer
```

```
Dim B As Float

A = 1210
B = 2438
```

數字前面可以有一個加號 (+) 或減號 (-) (數字與符號之間可以有空格，也可以沒有空格)：

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

小數

輸入小數時，請使用句點 (.) 作爲小數點。此規則可確保在無需轉換的情形下將一個國家/ 地區的源文字格式轉用到另一個國家/ 地區。

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

A = 1223.53       ' 被捨入
B = - 23446.46    ' 被捨入
C = + 3532.76323
```

小數前面也可以使用加號 (+) 或減號 (-) (同樣可以有空格，也可以沒有空格)。

如果將一個小數指定給一個整型變數，則 StarSuite Basic 會將該數捨入。

指數撰寫樣式

StarSuite Basic 允許以指數撰寫樣式指定數字，例如，可以將數字 1.5×10^{-10} (0.00000000015) 寫爲 1.5e-10。字母「e」可以是小寫，也可以是大寫，其前面可以有一個加號 (+)，也可以沒有。

以下是用指數格式表示數字的幾個正確示例和錯誤示例：

```
Dim A As Double

A = 1.43E2            ' 正確
A = + 1.43E2          ' 正確 (加號和基本數字之間有空格)
A = - 1.43E2          ' 正確 (減號與基本數字之間有空格)
A = 1.43E-2           ' 正確 (負指數)

A = 1.43E -2          ' 錯誤 (數字中不允許有空格)
A = 1,43E-2           ' 錯誤 (不允許將逗號用作小數點)
A = 1.43E2.2          ' 錯誤 (指數必須爲整數)
```

請注意，在第一個和第三個錯誤示例中，盡管變數返回錯誤的數值，但不會生成錯誤訊息。並將表示式

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

解譯爲 1.43 減 2，與值 -0.57 相對應。但是，值 $1.43 * 10^{-2}$ (對應於 0.0143) 是預期的值。對於值

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

StarSuite Basic 忽略指數中小數點後面的部份，並將表示式解譯爲

```
A = 1.43E2
```

十六進制值

十六進制系統 (基數爲 16 的系統) 的優點是，一個兩位數字精確地對應於一個位元組。這樣就可以透過一種更接近於反映機器架構的方式來處理數字。在十六進制系統中，將數字 0 到 9 以及字母 A 到 F 用作數字。字母 A 表示十進制數字 10，而字母 F 表示十進制數字 15。StarSuite Basic 允許使用整數的十六進制值，但整數前面應加上 &H。

```
Dim A As Longer

A = &HFF      ' 十六進制值 FF，對應於十進制值 255
A = &H10      ' 十六進制值 10，對應於十進制值 16
```

八進制值

StarSuite Basic 還識別使用 0 到 7 的數字的八進制系統 (基數爲 8 的系統)，但整數前面應加上 &O。

```
Dim A As Longer

A = &O77      ' 八進制值 77，對應於十進制值 63
```

```
A = &O10       ' 八進制值 10，對應於十進制值 8
```

True 和 False

布林型變數

布林型變數只能含有以下兩個值之一：`True` 和 `False`。它們適用於只能採用已命名狀態之一的二進制規格。布林值在內部儲存爲兩個位元組長的整數值，其中，0 對應於 `False`，而任何其他值對應於 `True`。布林型變數没有類型宣告符號，只能使用增補 *As Boolean* 進行宣告。

布林型變數的宣告示例：

```
Dim Variable As Boolean
```

日期和時間細節

日期型變數

日期型變數可以含有日期值和時間值。儲存日期值時，StarSuite Basic 使用一種內部格式，該格式允許對日期值和時間值進行比較和數學計算。日期型變數没有類型宣告符號，只能使用增補 *As Date* 進行宣告。

日期型變數的宣告示例：

```
Dim Variable As Date
```

資料欄位

除了簡單變數 (數量) 外，StarSuite Basic 還支援資料欄位 (陣列)。資料欄位包含多個變數，這些變數透過一個索引進行存取。

簡單陣列

陣列宣告與簡單變數宣告相似，但與變數宣告不同的是，陣列名稱後面有括號，其中含有元素數目規格。表示式

```
Dim MyArray(3)
```

宣告了一個陣列，其中含有四個變體型資料類型的變數，即 `MyArray(0)`、`MyArray(1)`、`MyArray(2)` 和 `MyArray(3)`。

也可以在一個陣列中宣告特定類型的變數，例如，以下行宣告一個具有四個整型變數的陣列：

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

在上面的示例中，陣列的索引總是以標準起始值 0 開始。另外，也可以對資料欄位宣告指定一個含有起始值和結束值的有效範圍。以下示例宣告一個含有六個整數值的資料欄位，而且該資料欄位可使用索引 5 到 10 進行存取：

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

索引不必爲正值。以下示例顯示的也是一個正確宣告，但具有負的資料欄位界限值。

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

它宣告了一個具有六個值的整數資料欄位，欄位可以使用索引 -10 到 -5 進行存取。

定義資料欄位索引時必須遵守三個限制：

允許的最小索引爲 -32768。

允許的最大索引爲 32767。

元素數目 (在一個資料欄位維內) 最大爲 16368。

VBA 中的資料欄位索引有時採用其他界限值。每一維上可能的最大元素數目也採用同樣的值。在相關的 VBA 文件中可找到對應的有效值。

起始索引的指定值

資料欄位的起始索引通常以數值 0 開始。可以呼叫以下指令將所有資料欄位宣告的起始索引變更爲數值 1：

```
Option Base 1
```

如果要對某個模組中的 所有陣列宣告採用此呼叫，則必須將其包含在該模組的標頭中。但是，此呼叫不影響透過 StarSuite API 定義的 UNO 序列，該序列的索引始終以 0 開始。要使程式更清晰，應避免使用 Option Base 1。

如果使用 Option Base 1，陣列中元素的數目不會受到影響，只有起始索引會發生變更。宣告

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

建立了 4 個整型變數，它們可以用表示式 `MyInteger(1)`、`MyInteger(2)`、`MyInteger(3)` 和 `MyInteger(4)` 進行描述。

與在 VBA 中不同，在 StarSuite Basic 中，表示式 Option Base 1 不影響陣列中的元素數目。只是 StarSuite Basic 中的起始索引發生了變更。在 VBA 中，宣告 `MyInteger(3)` 建立索引爲 1 到 3 的三個整數值；而在 StarSuite Basic 中，該宣告建立索引爲 1 到 4 的四個整數值。

多維資料欄位

除了單維資料欄位之外，StarSuite Basic 還支援使用多維資料欄位。相應的維之間用逗號分隔。示例

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

定義了一個二維整數陣列，每一維都具有 6 個索引 (可透過索引 0 到 5 進行存取)。整個陣列總共可記錄 6 × 6 = 36 個整數值。

盡管可以在 StarSuite Basic 陣列中定義幾百個維，但是，可用的記憶體數量限制了可以擁有的維數。

資料欄位維數的動態變更

上面的示例基於指定維數的資料欄位。也可以定義資料欄位的維數可以動態變更的陣列。例如，可以定義一個陣列，使該陣列包含文字中所有以字母 A 起始的字詞。由於最初不知道這些字詞的數目，因此需要以後能夠變更欄位界限值。爲此，在 StarSuite Basic 中使用以下呼叫：

```
ReDim MyArray(10)
```

與 VBA 中只能使用 Dim MyArray() 定義動態陣列維數不同，在 StarSuite Basic 中，可以使用 ReDim 變更靜態和動態陣列。

以下示例變更了初始陣列的維數，這樣該陣列就可以記錄 11 或 21 個值：

```
Dim MyArray(4) As Integer           ' 帶有五個元素的宣告

' ...

ReDim MyArray(10) As Integer        ' 增加到 11 個元素

' ...

ReDim MyArray(20) As Integer        ' 增加到 21 個元素
```

當重設某個陣列的維數時，可以使用前面幾節中概述的任何選項。這包括宣告多維資料欄位和/ 或指定明確的起始值和結束值。變更資料欄位的維數後，會遺失所有內容。如果要保留原始數值，請使用 `Preserve` 指令：

```
Dim MyArray(10) As Integer                          ' 定義初始
                                                    ' 維數

' ...

ReDim Preserve MyArray(20)   As Integer             ' 資料欄位中的
                                                    ' 維數增加時，
                                                    ' 保留內容
```

使用 `Preserve` 時，請確定維數和變數類型保持相同。

在 VBA 中，使用 `Preserve` 時只能變更一個資料欄位的最後一維的上限；而在 StarSuite Basic 中，使用該指令還可以變更其他維。

如果將 `ReDim` 與 `Preserve` 一起使用，就必須使用原始資料欄位宣告中指定的資料類型。

變數的作用範圍和生存期

StarSuite Basic 中的變數具有有限的生存期和有限的作用範圍。在此期間和範圍內，可在其他程式段中讀取和使用該變數。一個變數的生存期及作用範圍取決於指定它的位置和類型。

局部變數

在函數或程序中宣告的變數稱爲局部變數：

```
Sub Test
      Dim MyInteger As Integer

      ' ...
End Sub
```

局部變數僅在該函數或程序執行時有效，然後會被重設爲 0。每次呼叫該函數時，無法再使用以前產生的值。

若要保留以前的值，必須將變數定義爲靜態：

```
Sub Test
      Static MyInteger As Integer

      ' ...

End Sub
```

與 VBA 不同，StarSuite Basic 會確保不在模組標頭中將一個局部變數的名稱同時用作全域變數和私有變數。將 VBA 應用程式移植入 StarSuite Basic 時，必須變更任何重複的變數名稱。

公用域變數

公用域變數是透過關鍵字 Dim 在一個模組的標頭區域中定義的。這些變數可用於其程式庫中的所有模組：

模組 A：

```
Dim A As Integer

Sub Test
      Flip
      Flop
End Sub

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

模組 B：

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

變數 A 的值未在 Test 函數中變更，而是在 Flip 函數中增加了 1，在 Flop 函數中減少了 1。該變數的這兩種變更均爲全域變更。

也可以使用關鍵字 Public 而不是 Dim 來宣告一個公用域變數：

```
Public A As Integer
```

一個公用域變數僅在關聯的巨集執行時可用，然後會被重設。

全域變數

就其功能而言，全域變數與公用域變數相似，不同的是全域變數的值在執行關聯的巨集之後仍然保留。全域變數使用關鍵字 Global 在一個模組的標頭區塊中宣告：

```
Global A As Integer
```

私有變數

私有 變數僅在定義它們的模組中可用。可以使用關鍵字 `Private` 定義私有變數：

```
Private MyInteger As Integer
```

如果多個模組中含有相同名稱的私有變數，StarSuite Basic 會在該名稱每次出現時建立 一個不同的變數。在以下示例中，模組 A 和模組 B 都具有一個私有變數 C。函數 `Test` 先在模組 A 中設定該私有變數，然後再在模組 B 中設定該私有變數。

模組 A：

```
Private C As Integer

Sub Test
      SetModuleA            ' 在模組 A 中設定變數 C
      SetModuleB            ' 在模組 B 中設定變數 C

      ShowVarA              ' 顯示模組 A 中的變數 C (等於 10)
      ShowVarB              ' 顯示模組 B 中的變數 C (等於 20)
End Sub

Sub SetmoduleeA
      A = 10
End Sub

Sub ShowVarA
      MsgBox C              ' 顯示模組 A 中的變數 C。
End Sub
```

模組 B：

```
Private C As Integer

Sub SetModuleB
     A = 20
End Sub

Sub ShowVarB
      MsgBox C              ' 顯示模組 B 中的變數 C。
End Sub
```

常數

在 StarSuite Basic 中，使用關鍵字 Const 來宣告常數。

```
Const A = 10
```

如果需要，還可以在宣告中指定常數類型：

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

運算符

StarSuite Basic 可識別常用數學運算符、邏輯運算符和比較運算符。

數學運算符

數學運算符可採用至所有數字類型，而 + 運算符還可用於連結字串。

+ 數字或日期值相加以及字串連結

- 數字或日期值相減

* 數字相乘

/ 數字相除

\ 數字相除且結果爲一個整數 (被捨入)

^ 數字的乘冪運算

求餘數 模組運算 (計算除法運算的餘數)

邏輯運算符

邏輯運算符允許您依照布林代數規則連結元素。如果對布林值採用運算符，則該連結將直接提供需要的結果。如果與整數值和長型整數值一起使用，則在位元級進行連結。

AND 和連結

OR 或連結

互斥或 互斥或連結

NOT 否定

等價 相等測試 (兩部份相等 (True) 或 不相等 (False))

IMP 隱含 (如果第一個表示式爲真，則第二個表示式一定也爲真)

比較運算符

比較運算符可採用至所有基本變數類型 (數字、日期細節、字串和布林值)。

=	數字、日期值和字串的相等性
<>	數字、日期值和字串的不相等性
>	大於，用於比較數字、日期值和字串
>=	大於或等於，用於比較數字、日期值和字串
<	小於，用於比較數字、日期值和字串
>=	小於或等於，用於比較數字、日期值和字串

StarSuite Basic 不支援 VBA 的 `Like` 比較運算符。

分支

使用分支陳述限制程式碼區塊的執行，直到滿足某個特定條件爲止。

If...Then...Else

最常用的分支陳述是 `If` 陳述，如以下示例所示：

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

僅當變數 A 的值大於 3 時，`B = 2` 指定才會發生。`If/Else` 子句是 `If` 陳述的變體：

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

在此示例中，當 A 大於 3 時，變數 B 的值被指定爲 2，否則，B 的值被指定爲 0。

如果需要更複雜的陳述，可重疊使用 If 陳述，例如：

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

如果變數 A 的值等於零，就將 B 的值指定爲 0。如果 A 小於 3 且不等於 0，則將 B 的值指定爲 1。在所有其他情況下 (即，如果 A 大於或等於 3)，則將 B 的值指定爲 2。

Select...Case

Select...Case 指令可替代重疊的 If 陳述。當您需要依各種條件來檢查某個值時，可用此指令：

```
Select Case DayOfWeek
Case 1:
      NameOfWeekday = "Sunday"
Case 2:
      NameOfWeekday = "Monday"
Case 3:
      NameOfWeekday = "Tuesday"
Case 4:
      NameOfWeekday = "Wednesday"
Case 5:
      NameOfWeekday = "Thursday"
Case 6:
      NameOfWeekday = "Friday"
Case 7:
      NameOfWeekday = "Saturday"
End Select
```

在此示例中，一個工作日的名稱對應於一個數字，因此，當名稱爲 Sunday 時，將 DayOfWeek 變數的值指定爲 1；當名稱爲 Monday 時，將該變數的值指定爲 2，以此類推。

Select 指令不限於簡單的 1:1 指定，也可以在一個 Case 分支中指定比較運算符或表示式清單。以下示例列出了最重要的語法變體：

```
Select Case Var
Case 1 To 5

      ' ... Var 介於數字 1 和 5 之間

Case 6, 7, 8

      ' ... Var 爲 6、7 或 8

Case Var > 8 And Var < 11

      ' ... Var 大於 8，且小於 11

Case Else

      ' ... 所有其他情形

End Select
```

迴路

迴路依照指定的遍數執行某個程式碼區塊。也可以使用未定義遍數的迴路。

For...Next

For...Next 迴路具有固定的遍數。迴路計數器定義了迴路要執行的次數。在以下示例中，

```
Dim I

For I = 1 To 10

      ' ...  迴路的內部

Next I
```

變數 I 爲迴路計數器，其初始值爲 1。在每遍結束時，計數器遞增 1。當變數 I 等於 10 時，迴路停止。

如果要在每遍結束時使迴路計數器遞增一個非 1 的值，請使用 Step 功能：

```
Dim I

For I = 1 To 10 Step 0.5

      ' ... 迴路的內部

Next I
```

在以上示例中，計數器在每遍結束時遞增 0.5，該迴路執行了 19 次。

也可以使用負的 Step 值：

```
Dim I

For I = 10 To 1 Step -1

      ' ... 迴路的內部

Next I
```

在此示例中，計數器從 10 開始，每遍結束時遞減 1，直到計數器爲 1 爲止。

Exit For 指令用於提前結束 For 迴路。在以下示例中，在執行第五遍時終止迴路：

```
Dim I

For I = 1 To 10

      If I = 5 Then
            Exit For
      End If

      ' ... 迴路的內部

Next I
```

StarSuite Basic 不支援 VBA 中的 For Each...Next 迴路變體。

Do...Loop

Do...Loop 不與一個固定的遍數相連結。相反，Do...Loop 一直執行，直到滿足某個條件爲止。Do...Loop 有四種變體 (在以下示例中，A > 10 表示任意條件)：

1. Do While...Loop 變體

```
  Do While A > 10
      ' ... 迴路內文
  Loop
```

   每遍迴路之前檢查是否仍然滿足條件，只有滿足條件時才執行。

2. Do Until...Loop 變體

```
Do Until A > 10
      ' ... 迴路內文
Loop
```

   執行迴路，直到不再滿足條件爲止。

3. Do...Loop While 變體

```
Do
      ' ... 迴路內文
Loop While A > 10
```

   僅在執行第一遍迴路之後才檢查條件，如果滿足此條件，則終止迴路。

4. Do...Loop Until 變體

```
Do
      ' ... 迴路內文
Loop Until A > 10
```

   也是在第一遍之後檢查迴路的條件，但繼續執行迴路，直到不再滿足條件爲止。

與在 For...Next 迴路中一樣，Do...Loop 也提供一條終止指令。Exit Do 指令可以在迴路內的任意位置結束迴路。

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

      ' ... 迴路內文
While A > 10
```

程式設計示例：用內嵌式迴路排序

迴路有很多種用途，例如，搜尋清單、傳回值或執行複雜的數學工作。以下示例是用迴路依名稱排序清單的演算法。

```
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
```

將值成對交換數次，直到最終以向上順序排序。像氣泡一樣，變數逐漸遷移至適合的位置。基於此，該算法也稱爲氣泡排序。

程序和函數

程序和函數形成一個程式結構中的樞軸點。它們提供將一個複雜問題分爲各種子工作的框架。

程序

程序執行動作時無需提供明確的值。其語法是

```
Sub Test

      ' ... 此處是程序的實際程式碼

End Sub
```

該示例定義了一個名爲 Test 的程序，可從程式中的任意位置存取該程序包含的程式碼。在程式的相關位置輸入該程序名即可呼叫該程序：

```
Test
```

函數

函數像程序一樣，將要執行的程式區塊組合爲一個邏輯單元。但是，與程序不同的是，函數提供傳回值。

```
Function Test

      ' ... 此處是函數的實際程式碼

      Test = 123
End Function
```

傳回值是使用簡單指定來指定的。該指定不必位於函數末尾，可以在函數的任何位置指定。

可以在程式中呼叫上述函數，如下所示：

```
Dim A

A = Test
```

該程式碼定義了一個變數 A，並將 Test 函數的結果指定給該變數。

在函數內可以多次覆寫傳回值。就像傳統型的變數指定一樣，此示例中的函數傳回最後一次指定的函數值。

```
Function Test

      Test = 12

      ' ...

      Test = 123

End Function
```

在此示例中，函數的傳回值爲 123。

如果指定被終止，則函數將傳回零值 (對於數字值是數字 0，而對於字串則是空白)。

一個函數的傳回值可以是任意類型，類型的宣告方法與變數宣告相同：

```
Function Test As Integer

      ' ... 此處是函數的實際程式碼

End Function
```

如果明確值指定被終止，則傳回值的類型將被指定爲變體型。

提前終止程序和函數

在 StarSuite Basic 中，可以使用 `Exit Sub` 和 `Exit Function` 指令提前終止程序或函數以執行某種作業 (例如錯誤處理)。這些指令可終止程序或函數，並使程式返回到呼叫該程序和/ 或函數的位置。

以下示例顯示了當
`ErrorOccured` 變數的值爲 `True` 時，終止執行該程序。

```
Sub Test
      Dim ErrorOccured As Boolean

      ' ...

      If ErrorOccured Then
            Exit Sub
      End If

      ' ...

End Sub
```

傳送參數

函數和程序可以接收一個或多個參數。必要的參數須置於函數名或程序名後面的括號中。該示例

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

定義了一個程序，該程序需要一個整數值 A 和一個字串 B 作爲參數。

在 StarSuite Basic 中，通常透過參照來傳送參數。結束程序或函數時，會保留對變數所做的變更：

```
Sub Test
      Dim A As Integer
      A = 10
      ChangeValue(A)
      ' 參數 A 現在的值爲 20
End Sub
Sub ChangeValue(TheValue As Integer)
       TheValue = 20
End Sub
```

在此示例中，Test 函數中定義的值 A 被作爲參數傳送給 ChangeValue 函數。然後該值會變更爲 20 並被傳送給 TheValue， 結束函數時，將保留該值。

如果您不希望隨後對參數的變更影響原來傳送的值，還可以將參數作爲值進行傳送。若要指定一個參數作爲值傳送，請確保在函數標頭的變數宣告前加上 ByVal 關鍵字。

在以上示例中，如果用以下函數代替 ChangeValue 函數

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

，則上級變數 A 不會受到此變更的影響。呼叫 ChangeValue 函數之後，變數 A 的值仍保持爲 10。

在 StarSuite Basic 中，將參數傳送到程序和函數的方法與 VBA 中的基本相同。依標準，參數透過參照傳送。若要將參數作爲值傳送，請使用 ByVal 關鍵字。在 VBA 中，您也可以使用關鍵字 ByRef 强制透過參照傳送參數。StarSuite Basic 不支援此關鍵字，因爲在 StarSuite Basic 中，這已經是標準程序。

通常，StarSuite Basic 中的函數和程序均爲公用。StarSuite Basic 不支援 VBA 中使用的 Public 和 Private 關鍵字。

可選參數

僅當傳送了所有呼叫時必需的參數後，才能呼叫函數和程序。

在 StarSuite Basic 中，可將參數定義爲可選，即，如果呼叫中不包括相應的值，則 StarSuite Basic 將傳送一個空參數。在示例

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

End Sub
```

中，參數 A 是必需的，而參數 B 則是可選的。

IsMissing 函數用於檢查某個參數已被傳送，還是被省略了。

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

      ' 檢查 B 參數是否實際存在
      If Not IsMissing (B) Then
             B_Local = B            ' B 參數存在
      Else
             B_Local = 0            ' 缺少 B 參數 -> 標準值 0
      End If

      ' ... 啓動實際函數

End Sub
```

該示例首先測試是否已傳送了 B 參數。如果已傳送，則將該參數傳送給內部的 B_Local 變數。如果相應的參數不存在，則將標準值 (在此實例中，標準值爲 0) 傳送給 B_ Local，而不是所傳送的參數。

StarSuite Basic 不支援 VBA 中爲定義可選參數的標準值而提供的選項。

StarSuite Basic 不支援 VBA 中的 ParamArray 關鍵字。

遞迴

現在 StarSuite Basic 支援遞迴。遞迴程序或函數能夠一直呼叫其本身，直到偵測到已經滿足某個基本條件。用基本條件呼叫遞迴函數時，會傳回一個結果。

以下示例使用遞迴函數來計算數字 42、-42 和 3.14 的階乘。

```
Sub Main
 Msgbox CalculateFactorial ( 42 )   ' 顯示 1,40500611775288E+51
 Msgbox CalculateFactorial ( -42 )  ' 顯示「對階乘運算無效的數字！」
 Msgbox CalculateFactorial ( 3.14 ) ' 顯示「對階乘運算無效的數字！」
End Sub

Function CalculateFactorial ( Number )
 If Number < 0 Or Number <> Int ( Number ) Then
 CalculateFactorial = "對階乘運算無效的數字！"
 ElseIf Number = 0 Then
 CalculateFactorial = 1
 Else
 ' 這是遞迴呼叫：
 CalculateFactorial = Number * CalculateFactorial ( Number - 1 )
 Endif
End Function
```

在該示例中，透過呼叫函數 CalculateFactorial 並直到滿足了基本條件 0! = 1，傳回了數字 42 的階乘。

請注意，目前 StarSuite Basic 中的遞迴層級限於 500。

錯誤處理

更正錯誤是程式設計中最耗時的工作之一。StarSuite Basic 提供了一系列簡化錯誤處理的工具。

On Error 指令

On Error 指令是任何錯誤處理的關鍵：

```
Sub Test
      On Error Goto ErrorHandler

      ' ... 執行工作，在其間可能會發生錯誤

Exit Sub

ErrorHandler:

      ' ... 用於進行錯誤處理的個別程式碼

End Sub
```

`On Error Goto ErrorHandler` 行定義了發生錯誤時 StarSuite Basic 如何繼續。`Goto ErrorHandler` 可確保 StarSuite Basic 結束目前程式行，然後執行 `ErrorHandler`: 程式碼。

Resume 指令

`Resume Next` 指令用於在執行錯誤處理程式中的程式碼之後，從程式中發生錯誤的位置的下一行開始繼續執行程式：

```
ErrorHandler:

      ' ... 用於進行錯誤處理的個別程式碼

      Resume Next
```

使用 `Resume Proceed` 指令指定一個跳換點，用於在錯誤處理之後繼續執行程式：

```
ErrorHandler:

      ' ... 用於進行錯誤處理的個別程式碼
       Resume Proceed

Proceed:

      ' ... 程式在錯誤處理之後從此處繼續執行
```

若要在發生錯誤時繼續執行程式，且不顯示錯誤訊息，請使用以下格式：

```
Sub Test
      On Error Resume Next

      ' ... 執行工作，在其間可能會發生錯誤

End Sub
```

請小心使用 `On Error Resume Next` 指令，因爲其影響是全域性的。如需更多資訊，請參閱有關結構化錯誤處理的提示。

有關錯誤資訊的查詢

在錯誤處理中，如果有錯誤描述並知道發生錯誤的位置和原因，會很有用：

`Err` 變數包含已發生錯誤的編號。

`Error$` 變數包含錯誤描述。

`Erl` 變數包含已發生的錯誤所在的行編號。

呼叫

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

顯示如何在一個訊息視窗中顯示錯誤資訊。

鑑於 VBA 在一個名爲 Err 的統計物件中總結錯誤訊息，StarSuite Basic 提供了 Err、Error$ 和 Erl 變數。

狀態資訊保持有效，直到程式遇到 Resume 或 On Error 指令，之後將重設資訊。

在 VBA 中，Err 物件的 Err.Clear 方法在錯誤發生後將重設錯誤狀態。而在 StarSuite Basic 中，此事件透過 On Error 或 Resume 指令執行。

有關結構化錯誤處理的提示

定義指令 On Error 和返回指令 Resume 都是 Goto 結構的變體。

如果要清晰地構造程式碼，以防止在使用此結構時產生錯誤，則不應在無監視的情形下使用跳換指令。

應謹慎使用 On Error Resume Next 指令，因爲這會關閉所有開啓的錯誤訊息。

最好的解決方案是在一個程式中僅使用一種錯誤處理方法 - 將錯誤處理與實際程式碼分隔開，且錯誤發生後不跳換回原始程式碼。

以下是錯誤處理程序的一個示例：

```
Sub Example

	' 在函數起始處定義錯誤處理程式
	On Error Goto ErrorHandler

	' ... 此處爲實際程式碼

	' 關閉錯誤處理
	On Error Goto 0

	' 結束常規程式的執行
Exit Sub

' 錯誤處理的起始位置
ErrorHandler:

	' 檢查是否存在預期的錯誤
	If Err = ExpectedErrorNo Then
		' ... 處理錯誤
	Else
		' ... 警告存在未預期錯誤
	End If

	On Error Goto 0' 關閉錯誤處理
End Sub
```

此程序的起始處定義了一個錯誤處理程式，隨後是實際程式碼。在程式碼結尾處，呼叫 `On Error Goto 0` 來關閉錯誤處理，並透過 `Exit Sub` 指令 (不要與 `End Sub` 混淆) 來結束程式的執行。

該示例首先檢查錯誤編號是否與預期的編號 (儲存在假想的 `ExpectedErrorNo` 常數中) 相對應，然後相應地處理錯誤。如果發生了另一個錯誤，系統會輸出一個警告。檢查錯誤編號很重要，因爲這樣 可以偵測到非預期的錯誤。

程式碼結尾處 `On Error Goto 0` 的呼叫重設錯誤的狀態資訊 (`Err` 系統變數中的錯誤程式碼)，這樣即可清楚地識別日後發生的錯誤。

第 3 章

StarSuite Basic 執行期間程式庫

以下各節介紹執行期間程式庫的主要函數。

轉換函數

在許多情形下，需要將變數從一種類型變更爲另一種類型。

隱含的和明確的類型轉換

將變數從一種類型變更爲另一種類型的最簡單方法是使用指定。

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

在此示例中，變數 A 是一個字串，而變數 B 是一個整數。StarSuite Basic 可確保在將變數 B 指定到變數 A 的過程中，將其轉換爲一個字串。此轉換比看起來要複雜得多：整數 B 以一個兩位元組長的數字形式駐留在工作記憶體中。而 A 是一個字串，電腦將每個字元 (每個數字) 儲存爲一個一位元組長或兩位元組長的值。因此，在將 B 的內容複製到 A 之前，必須將 B 轉換爲 A 的內部格式。

與大多數其他程式設計語言不同，Basic 自動執行類型轉換。但是，這可能會導致嚴重的後果。以下程式碼序列

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

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

乍看起來非常簡單，但經由更仔細地檢驗，最後會發現其中存在陷阱。Basic 解譯程式首先計算加法程序的結果，然後將此結果轉換爲一個字串，結果就會生成字串 2。
另一種情形，如果 Basic 解譯程式首先將起始值 B 和 C 轉換爲一個字串，並對結果採用加號運算符，則會生成字串 11。

使用變體型變數時也是如此：

```
Dim A
Dim B
Dim C

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

由於變體型變數可以包含數字和字串，因此不能確定是將數字 2 還是字串 11 指定給變數 A。

若要避免因隱含的類型轉換而著稱的錯誤來源，必須以一種嚴謹的方式撰寫程式，例如，不使用變體資料類型 (我們再次建議這樣做)。

爲了避免由隱含的類型轉換而導致的其他錯誤，StarSuite Basic 提供了一系列轉換函數，用於定義應在何時轉換運算的資料類型：

CStr(Var) - 將任意資料類型轉換爲一個字串。

CInt(Var) - 將任意資料類型轉換爲整型值。

CLng(Var) - 將任意資料類型轉換爲長型值。

CSng(Var) - 將任意資料類型轉換爲單精度型值。

CDbl(Var) - 將任意資料類型轉換爲雙精度型值。

CBool(Var) - 將任意資料類型轉換爲布林型值。

CDate(Var) - 將任意資料類型轉換爲日期型值。

您可以使用這些轉換函數定義 StarSuite Basic 如何執行這些類型轉換作業：

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

B = 1
C = 1

A = CInt(B + C)              ' 首先將 B 和 C 加起來，然後轉換
                               (生成數字 2)
A = CStr(B) + Cstr(C)        ' 將 B 和 C 轉換爲一個字串，然後
                             ' 組合起來 (生成字串「11」)
```

在該示例的第一個加法中，StarSuite Basic 首先將整型變數相加，然後將結果轉換爲一串字元。將字串 2 指定給 A。在第二個實例中，首先將 整型變數轉換爲兩個字串，然後透過指定將兩者連結起來。這樣，就將字串 11 指定給了 A。

CSng 和 CDbl 數值轉換函數也接受小數。必須將在國家/ 地區的特定相應設定中定義的符號用作小數點符號。相反地，CStr 方法在格式化數字、日期和時間細節時，則使用目前選取的國家/ 地區的特定設定。

Val 函數不同於 Csng、Cdbl 和 Cstr 方法。它將一個字串轉換爲一個數字，但卻總是需要用句點作爲小數點符號。

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)        '   在不考量國家/地區的特定設定的情況下已正確轉換
```

檢查變數的內容

在有些場合，無法轉換日期：

```
Dim A As String
Dim B As Date

A = "test"
B = A                    ' 建立錯誤訊息
```

在上面顯示的示例中，將字串 `test` 指定給日期型變數沒有任何意義，因此，Basic 解譯程式會報告一個錯誤。當嘗試將字串指定給布林型變數時，也是如此：

```
Dim A As String
Dim B As Boolean

A = "test"
B = A                    ' 建立錯誤訊息
```

同樣，Basic 解譯程式會報告一個錯誤。

透過在指定之前檢查程式，以確定要指定的變數內容是否與目標變數的類型相符，可以避免這些錯誤訊息。

StarSuite Basic 爲此提供以下測試函數：

`IsNumeric(Value)` - 檢查某個值是否爲數字。

`IsDate(Value)` - 檢查某個值是否爲日期。

`IsArray(Value)` - 檢查某個值是否爲陣列。

當查詢使用者輸入時，這些函數尤其有用。例如，可以檢查使用者是否輸入了有效的數字或日期。

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

在上面的示例中，如果 `UserInput` 變數包含一個有效的數值型值，就將此值指定給 `ValidInput` 變數。如果 `UserInput` 未包含有效的數字，則將 `ValidInput` 的值指定爲 `0`，並傳回錯誤訊息。

雖然 Basic 中存在用於檢查數字、日期細節和陣列的測試函數，但卻没有用於檢查布林型值的相應函數。不過，可以使用 `IsBoolean` 函數來模擬該功能：

```
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
```

`IsBoolean` 函數定義一個布林型的內部輔助變數 `Dummy`，並嘗試將其指定給傳送的變數。如果指定成功，則函數傳回 `True`。如果指定失敗，則會生成一個執行期間錯誤，這將中斷該測試函數以傳回錯誤。

如果 StarSuite Basic 中的字串包含一個非數值型值，而且將此值指定給一個數字，StarSuite Basic 不會生成錯誤訊息，但會將值 `0` 傳送給變數。此程序不同於 VBA。在 VBA 中，如果執行相應指定，就會觸發一個錯誤，並且終止執行程式。

字串

使用字元集

管理字串時，StarSuite Basic 使用統一碼字元集。Asc 和 Chr 函數可以確定屬於一個字元的統一碼值，並且/ 或者爲統一碼值找到相應的字元。以下表示式將各種統一碼值指定給 code 變數：

```
Code = Asc("A")              ' 拉丁字母 A (統一碼值 65)
Code = Asc("€")              ' 歐元字元 (統一碼值 8364)
Code = Asc("ë")              ' 西裡爾字母 л (統一碼值 1083)
```

相反地，表示式

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

可確保用數值 13 (表示硬換行) 初始化 MyString 字串。

Basic 語言中經常使用 Chr 指令在一個字串中插入控制項字元。
因此，指定

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

可確保文字前面有一個定位鍵字元 (統一碼值 9) 並在文字後面加入一個硬換行 (統一碼值 13)。

存取字串中的部份字元

StarSuite Basic 提供四個傳回部份字串的函數：

Left(MyString, Length) - 傳回 MyString 的前 Length 個字元 。

Right(MyString, Length) - 傳回 MyString 的後 Length 個字元 。

Mid(MyString, Start, Length) - 自 Start 處起，傳回 MyString 的前 Length 個字元。

Len(MyString) - 傳回 MyString 中的字元數。

以下是這些已命名函數的幾個呼叫示例：

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

MyString = " This is a small test"

MyResult = Left(MyString,5)          ' 提供字串「This i」
MyResult = Right(MyString, 5)        ' 提供字串「test」
MyResult = Mid(MyString, 8, 5)       ' 提供字串「a sma」
MyLength = Len(MyString)             ' 提供值 4
```

搜尋和代替

StarSuite Basic 提供 InStr 函數，用於在一個字串中搜尋另一個字串的部份內容：

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

SearchString 參數指定要在 MyString 中搜尋的字串。該函數傳回一個數字，該數字表示 SearchString 在 MyString 中首次出現的位置。如果要搜尋該字串的其他符合項，還可用該函數指定一個可選的起始位置，StarSuite Basic 將由此位置開始搜尋。此情形下，該函數的語法爲：

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

在上面的示例中，InStr 忽略字元的大小寫。若要變更搜尋以使 InStr 區分大小寫，請加入參數 0，如以下示例所示：

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

使用上面用於編輯字串的函數，程式設計師可以在一個字串中搜尋或代替另一個字串：

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

  Result = ""
  StartPos = 1
  CurrentPos = 1

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

該函數在原始的術語 Source 中,用 InStr 透過迴路搜尋傳送的 Search 術語。如果找到了搜尋術語，則擷取表示式之前的部份並將其寫入 Result 傳回緩衝區。並在搜尋術語 Search 所在位置加入 NewPart 字串。如果找不到搜尋術語的更多相符項，則函數將將保留該字串的剩餘部份，並將其加入傳回緩衝區中。函數傳回以此種方式生成的字串作爲替代程序的結果。

由於對字元序列的某個部份進行代替是最常用的功能之一，因此延伸了 StarSuite Basic 中的 Mid 函數，這樣即可自動執行此工作。以下示例

```
Dim MyString As String

MyString  = "This was my text"
Mid(MyString, 6, 3, "is")
```

從 MyString 字串的第六個位置開始，用字串 is 代替三個字元。

格式化字串

Format 函數將數字格式化爲字串。爲此，該函數需要指定一個 Format 表示式，然後將該表示式用作格式化數字的樣式。此樣式內的每個萬用字元會確保在輸出值中相應地格式化此項目。文件樣式中最重要的五個萬用字元是零 (0)、井字號 (#)、句點 (.)、逗號 (,) 和 美元符號 ($)。

文件樣式中的零字元會確保總是將數字放置在相應位置。如果没有提供數字，則會在對應的位置顯示 0。

句點表示在國家/ 地區的特定設定中由作業系統定義的小數點符號。

下面的示例顯示了零和句點字元如何定義一個表示式中小數點後的數位：

```
MyFormat = "0.00"

MyString = Format(-1579.8, MyFormat)      ' 提供「-1579,80」
MyString = Format(1579.8, MyFormat)       ' 提供「1579,80」
MyString = Format(0.4, MyFormat)          ' 提供「0,40」
MyString = Format(0.434, MyFormat)        ' 提供「0,43」
```

同樣，可以在一個數字前面加入零，以達到所需的長度：

```
MyFormat = "0000.00"

MyString = Format(-1579.8, MyFormat)      ' 提供「-1579,80」
MyString = Format(1579.8, MyFormat)       ' 提供「1579,80」
MyString = Format(0.4, MyFormat)          ' 提供「0000,40」
MyString = Format(0.434, MyFormat)        ' 提供「0000,43」
```

逗號表示作業系統使用的千位元分隔符，而井字號表示一個數位或位置，僅在輸入字串需要時才會顯示。

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

MyString = Format(-1579.8, MyFormat)      ' 提供「-1.579,80」
MyString = Format(1579.8, MyFormat)       ' 提供「1.579,80」
MyString = Format(0.4, MyFormat)          ' 提供「0,40」
MyString = Format(0.434, MyFormat)        ' 提供「0,43」
```

Format 函數用系統定義的相關貨幣符號代替美元符號萬用字元進行顯示：

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

MyString = Format(-1579.8, MyFormat)      ' 提供「-1.579,80 €」
MyString = Format(1579.8, MyFormat)       ' 提供「1.579,80 €」
MyString = Format(0.4, MyFormat)          ' 提供「0,40 €」
MyString = Format(0.434, MyFormat)        ' 提供「0,43 €」
```

StarSuite Basic 不支援在 VBA 中用於格式化日期和時間細節的 Format 指令。

日期和時間

StarSuite Basic 提供了日期型資料類型，它以二進制格式儲存日期和時間細節。

程式碼中的日期和時間細節的規格

透過一個簡單字串的指定，可以將某個日期指定給一個日期型變數：

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

此指定可以正常進行，因爲 StarSuite Basic 會自動將以字串定義的日期值轉換爲一個日期型變數。但是，此種指定可能會導致錯誤， 日期值和時間值在不同國家/ 地區有不同的定義和顯示方式。

由於 StarSuite Basic 在將字串轉換爲日期值時使用國家/ 地區的特定作業系統設定，因此只有當國家/ 地區的特定設定與字串表示式相符時，上面顯示的表示式才能正常執行。

爲避免此問題，應使用 `DateSerial` 函數將某個固定值指定給一個日期型變數：

```
Dim MyVar As Date

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

該函數參數的順序必須爲：年、月、日。該函數可確保將實際變數指定爲正確的值，而不考量國家/地區的特定設定。

TimeSerial 函數格式化時間細節的方式與 DateSerial 函數格式化日期的方式相同：

```
Dim MyVar As Date

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

其參數應依照以下順序指定：小時、分鐘、秒。

擷取日期和時間細節

以下函數與 DateSerial 和 TimeSerial 函數互爲補充：

Day(MyDate) - 從 MyDate 傳回該日期中的日

Month (MyDate) - 從 MyDate 傳回該日期中的月

Year (MyDate) - 從 MyDate 傳回該日期中的年

Weekday(MyDate) - 從 MyDate 傳回該日期爲星期幾

Hour(MyTime) - 從 MyTime 傳回該時間中的小時數

Minute(MyTime) - 從 MyTime 傳回該時間中的分鐘數

Second(MyTime) - 從 MyTime 傳回該時間中的秒數

這些函數從指定的日期型變數中擷取日期和/ 或時間部份。示例

```
Dim MyDate As Date

' ... 初始化 MyDate

If Year(MyDate) = 2003 Then

' ... 指定的日期中的年份爲 2003
End If
```

檢查 MyDate 中所儲存日期的年份是否爲 2003。同樣，示例

```
Dim MyTime As Date

' ... 初始化 MyTime

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

' ... 指定的日期中的年份爲 2003
End If
```

檢查 MyTime 是否介於 12 小時和 14 小時之間。

Weekday 函數傳回了所傳送的日期是星期幾：

```
Dim MyDate As Date
Dim MyWeekday As String

' ... 初始化 MyDate

Select Case WeekDay(MyDate)
case 1
      MyWeekday = "Sunday"
case 2
      MyWeekday = "Monday"
case 3
      MyWeekday = "Tuesday"
case 4
      MyWeekday = "Wednesday"
case 5
      MyWeekday = "Thursday"
case 6
      MyWeekday = "Friday"
case 7
      MyWeekday = "Saturday"
End Select
```

備註：將星期日視爲一個星期的第一天。

擷取系統日期和時間

在 StarSuite Basic 中，可以使用以下函數擷取系統時間和系統日期：

Date - 傳回目前日期

Time - 傳回目前時間

Now - 傳回目前的時間點 (日期和時間的組合值)

檔案和目錄

使用檔案是一個應用程式的基本工作之一。StarSuite API 爲您提供了一整套物件，用於建立、開啓和修改 Office 文件。第 4 章中詳細介紹了這些物件。盡管如此，在有些場合，仍需要直接存取檔案系統、搜尋目錄或編輯文字檔案。StarSuite Basic 執行期間程式庫爲這些工作提供了數個基本函數。

StarSuite 6.x 中不再提供某些 DOS 特有的檔案函數和目錄函數，或者其功能非常有限。例如，不提供對 ChDir、ChDrive 和 CurDir 函數的支援。需要檔案屬性作爲參數的函數中不再使用某些 DOS 特有的屬性 (例如，將隱藏檔案與系統檔案區分開來)。爲確保 StarSuite 的平台獨立性達到盡可能高的層次，此變更是必需的。

管理檔案

搜尋目錄

StarSuite Basic 中的 Dir 函數負責從目錄中搜尋檔案和子目錄。初次請求時，必須將一個包含所要搜尋目錄的路徑的字串指定給 Dir，作爲其第一個參數。Dir 的第二個參數指定所要搜尋的檔案或目錄。StarSuite Basic 傳回搜尋到的第一個目錄條目的名稱。爲了擷取下一個條目，應在不含參數的情形下請求 Dir 函數。如果 Dir 函數没有搜尋到條目，則傳回一個空字串。

以下示例顯示了如何使用 Dir 函數來請求位於一個目錄中的所有檔案。該程序將個別檔案名稱儲存在 AllFiles 變數中，然後將此變數顯示在一個訊息方塊中。

```
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
```

用 0 作爲 Dir 函數中的第二個參數可以確保 Dir 僅傳回檔案名稱而忽略目錄。此處可以指定以下參數：

0：傳回一般檔案

16：傳回子目錄

以下示例幾乎與上面的示例相同，只是 Dir 函數將值 16 作爲一個參數傳送，該函數傳回資料夾的子目錄，而不是檔案名稱。

```
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
```

與在 VBA 中的情形不同，當在 StarSuite Basic 中請求 `Dir` 時，使用參數 `16` 的 `Dir` 函數僅傳回某個資料夾的子目錄。(在 VBA 中，該函數還傳回標準檔案的名稱，這樣，需要進一步檢查才能僅擷取目錄。)

StarSuite Basic 中不存在 VBA 中提供的用於在目錄中專門搜尋具有 隱藏、系統檔案、歸檔和卷次名稱屬性的選項，因爲相應的檔案系統函數並非在所有的作業系統上都可用。

在 VBA 和 StarSuite Basic 中，`Dir` 中列出的路徑規格都可以使用 `*` 和 `?` 萬用字元。但是，在 StarSuite Basic 中，`*` 萬用字元可能僅是檔案名稱和/或檔案副檔名的最後一個字元，而在 VBA 中卻並非如此。

建立和刪除目錄

StarSuite Basic 提供了用於建立目錄的 `MkDir` 函數。

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

此函數用於建立目錄和子目錄。如果需要，還可建立階層結構中需要的所有目錄。例如，如果僅存在 `C:\SubDir1` 目錄，則呼叫

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

將同時建立 `C:\SubDir1\SubDir2` 目錄和 `C:\SubDir1\SubDir2\SubDir3` 目錄。

`RmDir` 函數用於刪除目錄。

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

如果目錄中包含子目錄或檔案，則也將其刪除。因此，使用 `RmDir` 時應謹慎。

在 VBA 中，`MkDir` 和 `RmDir` 函數僅與目前目錄相關。而在 StarSuite Basic 中，`MkDir` 和 `RmDir` 可用於建立或刪除多層級目錄。

在 VBA 中，如果目錄中包含檔案，則 `RmDir` 會生成一個錯誤訊息。而在 StarSuite Basic 中，則將目錄及其所有檔案全部刪除。

複製、重新命名、刪除檔案以及檢查檔案是否存在

呼叫

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

以 Destination 的名稱建立 Source 檔案的副本。

借助該函數，

```
Name OldName As NewName
```

可將 OldName 檔案重新命名爲 NewName。As 關鍵字語法以及不使用逗號這一事實可追溯到 Basic 語言的起源。

呼叫

```
Kill(Filename)
```

刪除 Filename 檔案。如果要刪除目錄 (含括其檔案)，則請使用 RmDir 函數。

FileExists 函數可用於檢查某個檔案是否存在：

```
If FileExists(Filename) Then
      MsgBox "file exists."
End If
```

讀取和變更檔案屬性

使用檔案時，能夠確定檔案屬性、上次變更檔案的時間以及檔案長度有時很重要。

呼叫

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

將傳回檔案的某些屬性。傳回值是作爲一個位元遮罩提供的，其中可能包含以下值：

1 : 唯讀檔案

16 : 目錄名稱

示例

```
Dim FileMask As Integer
Dim FileDescription As String

FileMask = GetAttr("test.txt")

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

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

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

MsgBox FileDescription
```

確定 `test.txt` 檔案的位元遮罩並檢查檔案是否爲唯讀以及是否是一個目錄。如果都不是，就將字串「normal」指定給 `FileDescription`。

因爲 VBA 中用於查詢 隱藏、系統檔案、歸檔和卷次名稱等檔案屬性的旗標是 Windows 所特有的，而在其他作業系統中不可用或者只是部份可用，所以 StarSuite Basic 不支援這些旗標。

`SetAttr` 函數允許變更檔案的屬性。因此，呼叫

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

可用於爲檔案提供唯讀狀態。可以透過以下呼叫來刪除現有的唯讀狀態：

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

`FileDateTime` 函數提供了上次變更檔案時的日期和時間。此處，依照作業系統上使用的國家/ 地區的特定設定來格式化日期。

```
FileDateTime("test.txt")      ' 提供上次變更檔案時的日期和時間。
```

`FileLen` 函數以位元組爲單位確定一個檔案的長度 (傳回值爲長型整數值)。

```
FileLen("test.txt")           ' 以位元組爲單位提供檔案長度
```

寫入和讀取文字檔案

StarSuite Basic 提供了一整套讀取和寫入檔案的方法。以下說明與使用文字檔案 (不是文字文件) 有關。

寫入文字檔案

存取一個文字檔案之前，必須先開啓該文字檔案。爲此，需要一個自由的檔案標示元，它可以清楚地標識檔案，以便隨後存取。

`FreeFile` 函數用於建立一個自由的檔案標示元。標示元用作 `Open` 指令的一個參數，該指令可用以開啓檔案。若要開啓一個檔案以便將其指定爲文字檔案，則 `Open` 呼叫爲：

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

`Filename` 是一個包含檔案名稱的字串。`FileNo` 是由 `FreeFile` 函數建立的標示元。

一旦開啓該檔案，即可逐行描述 `Print` 指令：

```
Print #FileNo, "This is a test line."
```

`FileNo` 在此還代表檔案標示元。第二個參數指定要儲存爲文字檔案之一行的文字。

一旦完成寫入程序，就必須使用一個 `Close` 呼叫關閉檔案：

```
Close #FileNo
```

此處同樣應該指定檔案標示元。

以下示例顯示如何開啓、描述和關閉文字檔案：

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

Filename = "c:\data.txt"                              ' 定義檔案名稱
FileNo = Freefile                                     ' 建立自由的檔案標示元

Open Filename For Output As #FileNo                   ' 開啓檔案 (寫入模式)
Print #FileNo, "This is a line of text"               ' 儲存行
Print #FileNo, "This is another line of text"         ' 儲存行
Close #FileNo                                         ' 關閉檔案
```

讀取文字檔案

文字檔案的讀取方式與其寫入方式相同。用來開啓該檔案的 `Open` 指令包含 `For Input` 表示式 (用來取代 `For Outut` 表示式)，並應該使用 Line Input 指令來讀取資料，而不是用於寫入資料的 `Print` 指令

最後，當呼叫一個文字檔案時，指令

```
eof(FileNo)
```

用於檢查是否已到達檔案末尾。

以下示例顯示如何讀取文字檔案：

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

' 定義檔案名稱
Filename = "c:\data.txt"

' 建立自由的檔案標示元
FileNo = Freefile

' 開啓檔案 (讀取模式)
Open Filename For Input As FileNo

' 檢查是否到達檔案末尾
Do While not eof(FileNo)

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

Loop

' 關閉檔案
Close #FileNo

Msgbox Msg
```

在 `Do While` 迴路中擷取的各行儲存在 `Msg` 變數中，並顯示在一個訊息方塊的末端。

訊息方塊和輸入方塊

StarOffice Basic 提供了 `MsgBox` 和 `InputBox` 函數，用於簡單的使用者通訊。

輸出訊息

`MsgBox` 顯示一個簡單的資訊方塊，該方塊可包含一個或多個按鈕。在其最簡單的變體

```
MsgBox "This is a piece of information!"
```

中，`MsgBox` 僅包含文字和一個[確定]按鈕。

可以使用一個參數來變更資訊方塊的外觀。該參數提供用於新增其他按鈕、定義預先指定的按鈕以及加入資訊符號的選項。用於選取按鈕的值爲：

0 - [確定]按鈕

1 - [確定]和[取消]按鈕

2 - [取消]和[重複]按鈕

3 - [是]、[否]和[取消]按鈕

4 - [是]和[否]按鈕

5 - [重複]和[取消]按鈕

若要將一個按鈕設定爲標準按鈕，請用以下值之一加上按鈕選擇清單中的參數值。例如，要建立[是]、[否]和[取消]按鈕 (值 3)，並使[取消]按鈕爲標準按鈕 (值 512)，則參數值將爲 3 + 512 = 515。

0 - 第一個按鈕爲標準值

256 - 第二個按鈕爲標準值

512 - 第三個按鈕爲標準值

最後，可以使用以下資訊符號，還可透過加入相關參數值來顯示這些資訊符號：

16 - 停止符號

32 - 問號

48 - 感嘆號

64 - 提示圖示

呼叫

```
MsgBox "Do you want to continue?",  292
```

顯示一個包含[是]和[否]按鈕 (值 4) 的資訊方塊，其中，第二個按鈕 ([否]) 被設定爲標準值 (值 256)，而且該資訊方塊還包含一個問號 (值 32)，因此參數值爲 4+256+32=292。

如果一個資訊方塊包含多個按鈕，則應查詢傳回值，以確定按了哪個按鈕。在此情形下，可以使用以下傳回值：

1 - [確定]

2 - [取消]

4 - [重複]

5 - [忽略]

6 - [是]

7 - [否]

在前面的示例中，可以按以下方式檢查傳回值：

```
If MsgBox ("Do you want to continue?",  292) = 6 Then
       ' 按下[是]按鈕
Else
       ' 按下[否]按鈕
End IF
```

除了資訊文字和用於排序資訊方塊的參數之外，MsgBox 還允許有第三個參數，該參數定義方塊標題的文字：

```
MsgBox "Do you want to continue?",  292, "Fenstertitel"
```

如果不指定方塊標題，則標準值是「soffice」。

用於查詢簡單字串的 Input Box

InputBox 函數可查詢使用者輸入的簡單字串。因此，它是調整對話方塊的一種簡單的替代方法。InputBox 接受三個標準參數：

一個資訊文字，

一個方塊標題，

一個可在輸入區域中加入的標準值。

```
InputVal = InputBox("Please enter value:", "Test", "default value")
```

InputBox 將使用者輸入的字串作爲傳回值。

其他函數

Beep

Beep 函數使系統播放一種聲音，該聲音可用於警告使用者執行了錯誤的動作。Beep 不含任何參數：

```
Beep   ' 建立一種資訊音調
```

Shell

使用 Shell 函數可以啓動外部程式。

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

Pathname 定義要執行的程式的路徑。Windowstyle 定義用於啓動程式的視窗。可能採取的值如下：

0 - 程式接收到焦點，並在一個隱藏視窗中啓動。

1 - 程式接收到焦點，並在一個一般大小的視窗中啓動。

2 - 程式接收到焦點，並在一個最小化視窗中啓動。

3 - 程式接收到焦點，並在一個最大化視窗中啓動。

4 - 程式在一般大小的視窗中啓動，但没有接收到焦點。

6 - 程式在一個最小化視窗中啓動，焦點保留在目前視窗中。

10 - 程式以全螢幕模式啓動。

第三個參數 Param 允許將指令行參數傳送到要啓動的程式。

Wait

Wait 函數將程式執行終止一段時間。等待時間以毫秒爲單位指定。指令

```
Wait 2000
```

指定一個 2 秒 (2000 毫秒) 的中斷。

Environ

Environ 函數傳回作業系統的環境變數。此處可以儲存各種類型的資料，具體情形取決於系統和調整。呼叫

```
Dim TempDir

TempDir=Environ ("TEMP")
```

確定作業系統暫存目錄的環境變數。

第 4 章

StarSuite API 介紹

StarSuite API 是用於存取 StarSuite 的通用程式設計介面。使用 StarSuite API 可以建立、開啓、修改和列印輸出 StarSuite 文件。它透過個人巨集提供了擴展 StarSuite 功能範圍的選項並允許寫入個人對話方塊。

StarSuite API 不僅可以與 StarSuite Basic 一起使用，還可與其他程式設計語言 (例如 Java 和 C++) 一起使用。這要歸因於一種名爲通用網路物件 (UNO) 的技術，該技術爲各種程式設計語言提供了一個介面。

本章主要介紹如何借助 UNO 在 StarSuite Basic 中使用 StarSuite，並從一個 StarSuite Basic 程式設計師的觀點對 UNO 的主要概念進行說明。有關 StarSuite API 各部份的使用細節之介紹可在後續的章節中找到。

通用網路物件 (UNO)

StarSuite 以通用網路物件 (UNO) 的形式提供程式設計介面。這是一個物件導向的程式設計介面，StarSuite 又將其細分爲各種物件，從而確保可透過程式來控制對 Office 套裝軟體的存取。

由於 StarSuite Basic 是一種程序程式設計語言，因此必須要在其中加入數種語言結構才能使用 UNO。

爲了在 StarSuite Basic 中使用通用網路物件，需要對關聯的物件進行變數宣告。進行宣告時使用 `Dim` 指令 (請參閱第 2 章)。應使用 `Object` 類型指定來宣告物件型變數：

```
Dim Obj As Object
```

該呼叫宣告了一個名爲 `Obj` 的物件型變數。

然後必須初始化所建立的物件型變數，初始化後才可使用該變數。此作業可使用 `createUnoService` 函數來完成：

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

該呼叫將新建立的物件的參照指定給 `Obj` 變數。 `com.sun.star.frame.Desktop` 類似一個物件類型，但在 UNO 術語中，它被稱作「服務」而不是類型。依照 UNO 的原理，`Obj` 可描述爲對支援 `com.sun.star.frame.Desktop` 服務之物件的一個參照。因此，StarSuite Basic 中使用的「服務」術語，與其他程式設計語言中使用的類型或類別術語相對應。

但其中存在一個主要區別：一個通用網路物件可同時支援多種服務。而有些 UNO 服務又支援其他服務，這樣，透過一個物件就可以提供一整套的服務。例如，上述基於 `com.sun.star.frame.Desktop` 服務的物件還可以含括用於載入文件和結束程式的其他服務。

在 VBA 中，物件的結構是透過其所屬的類別來定義的，而在 StarSuite Basic 中，物件的結構是透過其所支援的服務來定義的。VBA 物件總是被嚴格指定給一個單一類別，而 StarSuite Basic 物件可以支援多種服務。

屬性和方法

StarSuite Basic 中的物件提供了一系列的屬性和方法，可以透過該物件來呼叫這些屬性和方法。

屬性

此處的屬性與物件具有的屬性類似，例如 `Document` 物件的 `Filename` 和 `Title`。

屬性可以透過簡單的指定來設定：

```
Document.Title = "Programmierhandbuch StarOffice 6.0"
Document.Filename = "progman.sxv"
```

屬性像常規變數一樣具有類型，該類型定義了該屬性所能記錄的值的類型。
上面的 `Filename` 和 `Title` 屬性是字串型的。

真實屬性和模擬屬性

在 StarSuite Basic 中，物件的大多數屬性都依照服務的 UNO 描述來定義。除了這些「真實」屬性之外，在 StarSuite Basic 中，還有一些屬性包括 UNO 層級的兩種方法。其中的一種方法用於查詢屬性的值，另一種用於設定屬性的值 (`get` 和 `set` 方法)。該屬性實際上是從兩種方法模擬而成。例如，UNO 中的字元物件提供了 `getPosition` 和 `setPosition` 兩種方法，透過這些方法可以呼叫和變更關聯的關鍵點。StarSuite Basic 程式設計師可以透過 `Position` 屬性來存取這些值。除了此屬性以外，原來的方法也可用 (此示例中爲 `getPosition` 和 `setPosition`)。

方法

可以將方法理解爲與物件直接相關的函數，並可以透過該函數呼叫此物件。例如，上面的 Document 物件可以提供 Save 方法，該方法的呼叫方式如下:

```
Document.Save()
```

方法就像函數一樣，可以含有參數並傳回值。這種方法呼叫的語法導向的是傳統型的函數。呼叫

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

在請求 Save 方法時，還爲文件物件指定 True 參數。
一旦方法完成，Save 就會將傳回值儲存在 Ok 變數中。

模組、服務和介面

StarSuite 提供了數百種服務。爲對這些服務做個簡單的概述，已將它們組合成了模組。對於 StarSuite Basic 程式設計師來說，這些模組並未提供重要的功能。只有在指定服務名稱時，模組名稱才是重要的，這是因爲指定的名稱中也必須要列出模組名稱。完整的服務名稱包括 com.sun.star 表示式，該表示式指定它爲 StarSuite 服務；接著是模組名稱，例如 frame；最後是實際的服務名稱，例如 Desktop。在談到的示例中，其完整名稱如下所示：

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

除了模組和服務這兩個術語之外，UNO 還引入了術語「介面」。雖然 Java 程式設計師可能比較熟悉這個術語，但在 Basic 中並未使用它。

一個介面可以組合多種方法。嚴格來說，UNO 中的服務不是支援方法，而是支援介面，而介面反過來又提供了不同的方法。換句話說，是將方法 (作爲組合) 以介面的形式指定到服務中的。Java 或 C++ 程式設計師可能會對這個細節特別感興趣，因爲在這兩種語言中請求方法時需要使用介面。但在 StarSuite Basic 中，方法與介面無關，而是透過相關的物件來直接呼叫方法。

但是，爲方便對 API 的理解，將方法指定到便於操作的各種介面非常有用，因爲不同的服務中都要使用許多介面。如果您對某個介面比較熟悉，則可以將您的知識運用到其他服務，從而舉一反三。

有些主要介面 (由不同服務觸發) 使用非常頻繁，因此在本章結尾處將再次列示這些介面。

使用 UNO 時所需的工具

現在的問題是哪些物件 (或服務，如果繼續使用 UNO 術語的話) 支援哪些屬性、方法和介面，及如何確定它們。除本手冊外，可以從以下來源獲得有關物件的更多資訊： `supportsService` 方法、除錯方法、《StarSuite 開發者指南》和 API 參照。

supportsService 方法

許多 UNO 物件都支援 `supportsService` 方法，可以使用該方法來確定物件是否支援特定服務。例如，呼叫

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

確定 `TextElement` 物件是否支援 `com.sun.star.text.Paragraph` 服務。

除錯屬性

StarSuite Basic 中的每個 UNO 物件都知道已經包含哪些屬性、方法和介面，並提供了以清單的形式傳回這些資訊的屬性。相應的屬性有：

`DBG_properties` - 傳回一個包含物件所有屬性的字串

`DBG_methods` - 傳回一個包含物件所有方法的字串

`DBG_supportetInterfaces` - 傳回一個包含支援物件的所有介面的字串。

以下程式碼顯示了如何在實際應用程式中使用 `DBG_properties` 和 `DBG_methods`。首先建立 `com.sun.star.frame.Desktop` 服務，然後在訊息方塊中顯示所支援的屬性和方法。

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

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

在使用 `DBG_properties` 時請注意，該函數將傳回一個特定服務在理論上可以支援的所有屬性，但並不能確保所討論的物件也可以使用這些屬性。因此，在呼叫屬性之前，必須使用 `IsEmpty` 函數來檢查物件是否真的可用。

API 參照

如需有關可用的服務及其介面、方法和屬性的更多資訊，請參閱 StarSuite API 的 API 參照。 此參照可以在 www.openoffice.org 上找到：

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

幾個重要介面的概述

有些 StarSuite 的介面存在於 StarSuite API 的許多部份中。它們定義了用於抽像工作的一系列方法，可將這些方法採用至各種問題。在此，將對這些介面中最常用的部份做一簡單介紹。

這些物件的由來將在本手册的後面章節中進行說明。在此，僅介紹物件的一些抽像的內容，StarSuite API 爲這些物件提供了一些重要介面。

建立上下文相關的物件

StarSuite API 提供了兩種建立物件的方法。其中的一種方法是本章開始處提到的 `createUnoService` 函數。 `createUnoService` 可以建立通用的物件。這類物件和服務也被稱爲上下文不相關服務。

除了上下文不相關服務以外，還存在上下文相關服務，此種服務的物件僅在與其他物件一起使用時才有用。例如，某個工作表文件中的繪圖物件僅在這一文件中存在。

com.sun.star.lang.XMultiServiceFactory 介面

上下文相關物件通常是透過該物件所依賴的某種物件方法而建立的。在 `XMultiServiceFactory` 介面中定義的 `createInstance` 方法尤其用於文件物件。

例如，可以使用工作表物件來建立上述繪圖物件，方法如下：

```
Dim RectangleShape As Object

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

文字文件中的段落樣式也可以用同樣的方式建立，方法如下：

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

用名稱存取下級物件

XNameAccess 和 XNameContainer 介面用在含有下級物件的物件中，用自然語言名稱就可以存取這些下級物件。

XNamedAccess 允許存取個別物件，XNameContainer 可以插入、修改和刪除元素。

com.sun.star.container.XNameAccess 介面

工作表中的工作表物件提供了 XNameAccess 的使用示例。該物件組合了工作表中的所有頁。可以使用 XNameAccess 中的 getByName 方法來存取個別頁：

```
Dim Sheets As Object
Dim Sheet As Object

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

getElementNames 方法提供所有元素名稱的摘要。作爲結果，它傳回一個含有這些名稱的資料欄位。以下示例顯示了如何在一個迴路中確定並顯示某個工作表中的所有元素名稱：

```
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
```

XNameAccess 介面的 hasByName 方法顯示基本物件中是否存在具有特定名稱的下級物件。因此，以下示例中會顯示一條訊息，告知使用者 Spreadsheet 物件中是否包含名爲 Sheet1 的頁。

```
Dim Sheets As Object

Sheets = Spreadsheet.Sheets
If Sheets.HasByName("Sheet1") Then
      MsgBox " Sheet1 available"
Else
      MsgBox "Sheet1 not available"
End If
```

com.sun.star.container.XNameContainer 介面

XNameContainer 介面用於插入、刪除和修改基本物件的下級元素。負責實現這些功能的函數爲：insertByName、removeByName 和 replaceByName。

以下是此介面的一個實際示例。 該示例呼叫一個文字文件，該文件包含一個 StyleFamilies 物件，使用此物件又可以處理文件的段落樣式 (ParagraphStyles)。

```
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")
```

`insertByName` 行使用 `ParagraphStyles` 物件中的同名名稱插入 `NewStyle` 樣式。`replaceByName` 行將 `ChangingStyle` 後的物件變更爲 `NewStyle`。最後，`removeByName` 呼叫將 `OldStyle` 後的物件從 `ParagraphStyles` 中刪除。

基於索引存取下級物件

`XIndexAccess` 和 `XIndexContainer` 介面用於含有下級物件的物件和可以使用索引存取的物件。

`XIndexAccess` 提供存取個別物件的方法。
`XIndexContainer` 提供插入和移除元素的方法。

com.sun.star.container.XIndexAccess 介面

`XIndexAccess` 提供 `getByIndex` 和 `getCount` 兩種呼叫下級物件的方法。`getByIndex` 提供具有特定索引的物件。`getCount` 傳回可用物件的數目。

```
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)
      ' 編輯工作表
Next I
```

該示例顯示了一個遍歷所有工作表元素的迴路，並用 `Sheet` 物件型變數儲存每個元素的參照。使用索引時請注意，`getCount` 傳回元素數目，不過，`getByIndex` 中的元素將從 `0` 開始編號，因此，迴路的計數變數是從 `0` 到 `getCount()-1`。

com.sun.star.container.XIndexContainer 介面

`XIndexContainer` 介面提供了 `insertByIndex` 和 `removeByIndex` 函數。其參數的結構與 `XNameContainer` 中的相應函數相同。

反覆存取下級物件

在有些情形下，物件可能含有許多個下級物件，但無法透過名稱或索引存取這些下級物件。這時，使用 XEnumeration 和 XenumerationAccess 介面比較適合。這是因爲它們提供了一種機制來遍歷物件的所有下級元素，而無需使用直接存取。

com.sun.star.container.XEnumeration 和 XenumerationAccess 介面

基本物件必須提供 XEnumerationAccess 介面，該介面僅包含 createEnumeration 方法。此方法傳回一個輔助物件，而這個物件反過來又提供具有 hasMoreElements 和 nextElement 方法的 XEnumeration 介面。這樣，就可以存取下級物件了。

以下示例遍歷一篇文字的所有段落：

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

該示例首先建立了一個 ParagraphEnumeration 輔助物件。然後在一個迴路中利用此物件逐步傳回文字的各個段落。一旦 hasMoreElements 方法傳回 False 值 (表示已經到達文字的末尾)，就會終止迴路。

第 **5** 章

使用 StarSuite 文件

StarSuite API 是一種結構化的 API，因而其絕大部份均可普遍應用於不同的工作。其中包括用於建立、開啓、儲存、轉換和列印文件以及進行文件樣式管理的各種介面和服務。由於這些功能區域可用於各類文件，因此本章將首先對其進行介紹。

StarDesktop

使用文件時，最常使用以下兩項服務：

`com.sun.star.frame.Desktop` 服務，此服務與 StarSuite 的核心服務類似。它提供了 StarSuite 的訊框物件功能，此服務中的所有文件視窗都已分類。也可以使用此服務建立、開啓和匯入文件。

`com.sun.star.document.OfficeDocument` 服務提供個別文件物件的基本功能。此服務提供了儲存、匯出和列印文件的方法。

啓動 StarSuite 時，`com.sun.star.frame.Desktop` 服務將自動開啓。爲此，StarSuite 建立了一個可透過全域名稱 `StarDesktop` 存取的物件。

`StarDesktop` 最重要的介面爲 `com.sun.star.frame.XComponentLoader`。該介面基本涵蓋 `loadComponentFromURL` 方法，此方法 負責建立、匯入和開啓文件。

`StarDesktop` 物件的名稱可回溯至 StarSuite 5，在此版本中，所有文件視窗都內嵌於一個名爲 StarDesktop 的共用應用程式中。在 StarSuite 的目前版本中，已不再使用可視的 StarDesktop。然而，由於此名稱可清楚地指示這是整個應用程式的一個基本物件，所以爲 StarSuite 的訊框物件保留了名稱 StarDesktop。

`StarDesktop` 物件取代了先前作爲根物件採用的 StarSuite 5 的 `Application` 物件。但與舊的 `Application` 物件不同，它主要負責開啓新文件。不再使用舊版 `Application` 物件中控制 StarSuite 螢幕描述的常駐功能 (例如，`FullScreen`、`FunctionBarVisible`、`Height`、`Width`、`Top`、`Visible`)。

雖然在 Word 中，使用中的文件透過 `Application.ActiveDocument` 存取，在 Excel 中，使用中的文件透過 `Application.ActiveWorkbook` 存取，但在 StarOffice 中，則由 `StarDesktop` 負責此項工作。在 StarSuite 6 中，使用中的文件物件是透過 `StarDesktop.CurrentComponent` 屬性存取的。

有關 StarSuite 中文件的基本資訊

使用 StarSuite 文件時，討論一下 StarSuite 中某些文件管理的基本問題很有用。這些問題包括了 StarSuite 文件的名稱構成方式以及這些檔案的儲存格式。

採用 URL 表示法表示的檔案名稱

由於 StarSuite 是作爲一種不依賴於平台的應用程式設計的，因此它使用 URL 表示法 (此表示法不依賴於任何作業系統)，正如在《Internet Standard RFC 1738》中對檔案名稱所做的定義那樣。使用此系統的標準檔案名稱都以前綴

```
file:///
```

開頭，後跟本機路徑。如果檔案名稱中包含子目錄，則這些子目錄使用 單一正斜線進行分隔，而不是 Windows 通常使用的反斜線。以下路徑參照磁碟機 `C:` 上的 `doc` 目錄中的 `test.sxw` 檔案。

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

爲了將本機檔案名稱轉換爲 URL，StarSuite 中提供了 `ConvertToUrl` 函數。
爲了將 URL 轉換爲本機檔案名稱，StarSuite 提供了 `ConvertFromUrl` 函數：

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

MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
            ' 提供 (在 Windows 下) c:\doc\test.sxw
```

本示例將本機檔案名稱轉換爲一個 URL，並將其顯示在訊息方塊中。然後再將 URL 轉換爲一個檔案名稱，並且也將其顯示出來。

此表示法所依據的《Internet Standard RFC 1738》允許使用 `0-9`、`a-z` 以及 `A-Z` 等字元。其他所有字元都會作爲移植碼插入 URL 中。爲此，這些字元會轉換成相應的 ISO 8859-1 (ISO-Latin) 字元集的十六進制值，且前面跟有一個百分號。例如，本機檔案名稱中的空格因此將成爲 URL 中的 `%20`。

XML 檔案格式

從 6.0 版起，StarSuite 提供了一種基於 XML 的檔案格式。透過使用 XML，使用者可以選擇在其他程式中開啓和編輯這些檔案。

檔案壓縮

由於 XML 基於標準文字檔案，所以生成的檔案通常都非常大。因此，StarSuite 會壓縮這些檔案，並將其儲存爲 ZIP 檔案。

透過使用 `storeAsURL` 方法選項，使用者可以直接儲存原來的 XML 檔案。請參閱 第 83 頁中的 storeAsURL 方法選項。

建立、開啓和匯入文件

開啓、匯入和建立文件將使用方法

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

loadComponentFromURL 中的第一個參數指定關聯檔案的 URL。

至於第二個參數，loadComponentFromURL 需要 StarSuite 爲進行管理而在其內部建立的視窗的訊框物件的名稱。通常在此處指定預先定義的名稱 _blank，這樣即可確保 StarSuite 建立一個新視窗。或者，也可以指定 _hidden，這樣即可確保載入相應文件，但仍不顯示。

使用這些參數，使用者就可以開啓 StarSuite 文件，因爲最後兩個參數可指定爲萬用字元 (虛擬值)：

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

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

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

以上呼叫會開啓 text.sxw 檔案，並在一個新視窗中顯示此檔案。

可以在 StarSuite Basic 中用此方法開啓無數個文件，然後再使用傳回的文件物件進行編輯。

StarDesktop.loadComponentFromURL 取代了舊版 StarSuite API 中的 Documents.Add 和 Documents.Open 方法。

代替文件視窗的內容

Frame 參數的名爲 _blank 和 _hidden 的值可確保 StarSuite 爲來自 loadComponentFromURL 的各個呼叫建立一個新的視窗。在某些情形下，代替現有視窗的內容是很有用的。在此種情形下，此視窗的訊框物件應含有一個明確的名稱。請注意，此名稱不得以底線開頭。而且，必須設定 SearchFlags 參數，以便建立相應的框架 (如果此框架尚未存在)。用於 SearchFlags 的相應常數爲：

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

以下示例顯示了如何借助訊框參數和 SearchFlags 代替開啓的視窗的內容：

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

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

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

MsgBox "Press OK to display the second document."

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

本示例首先在具有訊框名 MyFrame 的新視窗中開啓 test.sxw 檔案。一旦確認了訊息方塊，此訊息方塊就用 test2.sxw 檔案代替視窗的內容。

loadComponentFromURL 方法選項

loadComponentFromURL 函數的第四個參數爲一個 PropertyValue 資料欄位， 此欄位用於爲 StarSuite 提供多種開啓和建立文件的選項。此資料欄位必須爲每個選項提供一個 PropertyValue 結構。在此結構中，將選項名稱儲存字串，並儲存了關聯的值。

loadComponentFromURL 支援以下選項：

AsTemplate (布林型**)** - 如果爲 True，則從給定的 URL 中載入一個未命名的新文件。如果爲 False，就載入要編輯的文件樣式檔案。

CharacterSet (字串型**)** - 定義文件基於哪個字元集。

FilterName (字串型**)** - 爲 loadComponentFromURL 函數指定一個特殊篩選。可用的篩選名定義在 \share\config\registry\instance\org\openoffice\office\TypeDetection.xml 檔案中。

FilterOptions (字串型**)** - 用於定義篩選的其他選項。

JumpMark (字串型**)** - 一旦文件開啓，則跳換到在 JumpMark 中所定義的位置。

Password (字串型**)** - 傳送受保護檔案的密碼。

ReadOnly (布林型**)** - 載入唯讀文件。

以下示例顯示如何使用 FilterName 選項開啓在 StarSuite Calc 中以逗號分隔的文字檔案。

```
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())
```

`FileProperties` 資料欄位只包含一個值，因爲它記錄的是一個選項。
`Filtername` 屬性定義 StarSuite 是否使用 StarSuite Calc 文字篩選開啓檔案。

建立新文件

如果 URL 中指定的文件爲一種文件樣式，StarSuite 將自動建立一個新文件。

此外，如果只需要一個未經任何改寫的空白文件，則可以指定一個 `private:factory`-URL：

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

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

本呼叫建立了一個空白的 StarSuite Writer 文件。

文件物件

上一節中介紹的 `loadComponentFromURL` 函數會傳回一個文件物件。此文件物件支援 `com.sun.star.document.OfficeDocument` 服務，該服務反過來提供兩個重要介面：

- `com.sun.star.frame.XStorable` 介面 (此介面負責儲存文件) 以及
- `com.sun.star.view.XPrintable` 介面 (此介面包含列印文件的方法)。

變更爲 StarSuite 6 後，您會發現文件物件的功能範圍大多都未發生變更。例如，文件物件仍然會提供儲存和列印文件的方法。但是，這些方法的名稱和參數卻發生了變更。

儲存和匯出文件

StarSuite 文件是直接透過文件物件儲存的。爲此，可以使用 `com.sun.star.frame.XStorable` 介面的 `store` 方法：

```
Doc.store()
```

如果已爲文件指定了記憶體空間，則此呼叫就會正常運行。對於新文件來說，情形則有所不同。在此情形下，使用 `storeAsURL` 方法。該方法也是在 `com.sun.star.frame.XStorable` 中定義的，可將其用於定義文件的位置：

```
Dim URL As String
Dim Dummy()

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

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

除了上面的方法之外，`com.sun.star.frame.XStorable` 也提供了一些在儲存文件時很有用的輔助方法。它們是：

`hasLocation()` - 指定是否已爲文件指定了 URL。

`isReadonly()` - 指定文件是否具有唯讀保護。

`isModified()` - 指定文件自上次儲存以來是否做過修改。

用於儲存檔案的程式碼可以透過這些選項進行延伸，這樣，僅當確實對物件進行了修改時，系統才會將其儲存，而且也只有在實際需要時才查詢檔案名稱：

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

該示例首先檢查自上次儲存以來是否修改過相關文件。僅在已做過修改的情形下才會繼續儲存程序。如果文件已有 URL，且不是唯讀唯讀文件，則系統會將其儲存在現有的 URL 下。如果文件没有 URL 或是以唯讀狀態開啓，則以新的 URL 下將其儲存。

storeAsURL 方法選項

當用於 loadComponentFromURL 方法時，也可以使用 storeAsURL 方法以 PropertyValue 資料欄位形式對某些選項進行指定。這將確定儲存文件時 StarSuite 使用的程序。storeAsURL 提供以下選項：

CharacterSet (字串型**)** - 定義文件基於哪個字元集。

FilterName (字串型**)** - 爲 loadComponentFromURL 函數指定一個特殊篩選。可用的篩選名定義在 \share\config\registry\instance\org\openoffice\office\TypeDetection.xml檔案中。

FilterOptions (字串型**)** - 用於定義篩選的其他選項。

Overwrite (布林型**)** - 無需查詢即可覆寫已存在的檔案。

Password (字串型**)** - 傳送受保護檔案的密碼。

Unpacked (布林型**)** - 在子目錄中儲存文件 (未壓縮)。

以下示例顯示如何將 Overwrite 選項與 storeAsURL 一起使用：

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

' ... 初始化 Doc

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

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

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

此如果此名稱下已有檔案，本示例則會將 Doc 儲存在指定的檔案名稱下。

列印文件

與儲存文件類似，列印文件也是直接透過文件物件進行的。爲此，系統提供了 com.sun.star.view.Xprintable 介面的 Print 方法。
print 呼叫的最簡單形式爲：

```
Dim Dummy()

Doc.print(Dummy())
```

與 loadComponentFromURL 方法的情形相同，Dummy 參數是一個 PropertyValue 資料欄位，透過此欄位，StarSuite 可以指定多個列印選項。

Print 方法的選項

`print` 方法需要一個 `PropertyValue` 資料欄位作爲其參數，此欄位反映了 StarSuite 的[列印]對話方塊的設定：

`CopyCount` (整型) - 指定要列印的副本數。.

`FileName` (字串型) - 列印指定檔案中的文件。

`Collate` (布林型) - 建議印表機合併各頁副本。

`Sort` (布林型) - 列印多頁時對頁面進行排序 (`CopyCount` > 1)。

`Pages` (字串型) - 包含要列印的頁面清單 (在列印對話方塊中指定的語法)。

以下示例顯示如何使用 `Pages` 選項列印一個文件的多個頁面：

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

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

Doc.print(PrintProperties())
```

印表機選擇和設定

`com.sun.star.view.XPrintable` 介面提供了 `Printer` 屬性，此屬性用於選取印表機。此屬性會收到一個具有以下設定的 `PropertyValue` 資料欄位：

`Name` (字串型) - 指定印表機的名稱。

`PaperOrientation` (列舉型) - 指定紙張方向 (`com.sun.star.view.PaperOrientation.PORTRAIT` 值用於縱向格式, `com.sun.star.view.PaperOrientation.LANDSCAPE` 則用於橫向格式)。

`PaperFormat` (列舉型) - 指定紙張格式 (例如，用於 DIN A4 的 com.sun.star.view.PaperFormat.A4 或用於 US 字母的 `com.sun.star.view.PaperFormat.Letter`)。

`PaperSize` (`Size` 結構型) - 以百分之一公釐爲單位指定紙張大小。

以下示例顯示如何借助 Printer 屬性變更印表機和設定紙張的大小。

```
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      ' 相當於 20 cm
PaperSize.Height = 20000     ' 相當於 20 cm

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

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

Doc.Printer = PrinterProperties()
```

本示例定義了一個名爲 PaperSize 的物件，此物件具有 com.sun.star.awt.Size 類型。此爲指定紙張大小時所需的物件。另外，本示例還爲兩個 PropertyValue 條目建立了一個資料欄位，名爲 PrinterProperties。然後使用要設定並指定給 Printer 屬性的值初始化此資料欄位。依 UNO 的觀點，印表機不是一個真實屬性，而是一個模擬屬性。

文件樣式

文件樣式是含有格式化屬性的已命名清單。它們貫穿 StarSuite 的所有應用程式，協助其有效地簡化格式。如果使用者變更了一個文件樣式的其中一個屬性，則 StarSuite 會依此屬性自動調整所有文件區域。例如，使用者可以因此透過對文件進行一次集中修改，而變更所有一級標頭的字型類型。依相關的文件類型，StarSuite 能夠分別識別一整套不同類型的文件樣式。

StarSuite Writer 支援

字元樣式、

段落樣式、

框樣式、

頁面樣式

編號樣式

StarSuite Calc 支援

儲存格樣式

頁面樣式

StarSuite Impress 支援

字元元素樣式

簡報樣式

在 StarSuite 術語中，不同的文件樣式類型稱爲 `StyleFamilies`，這與文件樣式類型所基於的 `com.sun.star.style.StyleFamily` 服務相一致。
`StyleFamilies` 是透過文件物件存取的：

```
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")
```

本示例使用工作表文件的 `StyleFamilies` 屬性建立了一個含有所有可用儲存格樣式的清單。

單一文件樣式可透過索引直接存取：

```
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
```

與上一個示例相比，本示例中新增了在一個訊息方塊內逐個顯示所有儲存格樣式名稱的迴路。

有關各種格式化選項的細節

每種文件樣式類型都提供一整套個別的格式化屬性。以下簡要介紹了最重要的格式化屬性以及這些屬性的要點：

字元屬性，第 6 章文字文件
`com.sun.star.style.CharacterProperties` 服務

段落屬性，第 6 章文字文件
`com.sun.star.text.Paragraph` 服務

儲存格屬性，第 7 章工作表文件
`com.sun.star.table.CellProperties` 服務

頁面屬性，第 7 章工作表文件
`com.sun.star.style.PageStyle` 服務

字元元素屬性，第 7 章工作表文件
各種服務

這些格式屬性決不僅限於它們所說明的應用程式，而是可以通用。例如，第 7 章中所述的的多數頁面屬性因此不僅可以用於 StarSuite Calc，而且還可以用於 StarSuite Writer。

如需有關使用文件樣式的更多資訊，請參閱第 6 章文字文件的字元屬性和段落屬性的標準值一節。

第 6 章

文字文件

除了純字串以外，文字文件中也包含格式資訊。這些資訊可以出現在文字中的任意位置。如果使用表格，結構將更加複雜。這些表格不但包含一維字串，而且還包含二維欄位。目前，大部份字處理程式最終都提供了將繪圖物件、文字方塊和其他物件插入文字中所需的選項。這些物件可以位於換行和分頁之外，也可以位於頁面中的任意位置。

本章將介紹文字文件的主要介面和服務。第一節介紹文字文件的結構，並藉由一個 StarSuite 文件來集中說明如何使用 StarSuite Basic 程式執行遍歷作業的步驟。這一節重點介紹段落、段落部份及其格式。

第二節重點介紹如何有效率地使用文字文件。爲此，StarSuite 提供了數個輔助物件，例如 `TextCursor` 物件，這些物件是對第一節中所指定物件的補充。

第三節介紹一些文字以外的內容，集中說明了表格、文字方塊、文字欄位、內文標籤、內容目錄等。

有關如何建立、開啓、儲存和列印文件的資訊在第 5 章中介紹，因爲這些資訊不僅可以用於文字文件，還可以用於其他類型的文件。

文字文件的結構

文字文件實質上可以包含四種類型的資訊：

實際文字

用於格式化字元、段落和頁面的文件樣式

非文字元素，如表格、圖形和繪圖物件

文字文件的全域設定

本節主要說明了文字選項以及相關聯的格式選項。

StarSuite Writer 的 StarSuite 6.x API 之設計與前一版本有所不同。在舊版 API 中，主要使用 `Selection` 物件，它是針對一般使用者的使用者介面而設計的，因此舊版 API 著重於使用滑鼠控制來反白顯示工作。

StarSuite 6.x API 已經代替了使用者介面和程式設計師介面之間的這些連結。因此，程式設計師可以平行存取應用程式的所有部份，還可以同時使用文件中不同子區域內的物件。舊的 `Selection` 物件不再可用。

段落和段落部份

文字文件的核心由一序列段落組成。由於這些段落既沒有命名，也沒有建立索引，因此無法直接存取個別段落。但是，可以借助第 4 章中介紹的 Enumeration 物件來循序地遍歷這些段落。這樣就可以對這些段落進行編輯了。

但是，在使用 Enumeration 物件時，需要注意：
它不僅傳回段落，而且還傳回表格 (嚴格地講，在 StarSuite Writer 中，表格是一種特殊類型的段落)。因此，在存取傳回的物件之前，您應該檢查傳回的物件是支援用於段落的 com.sun.star.text.Paragraph 服務，還是支援用於表格的 com.sun.star.text.TextTable 服務。

以下示例在迴路中遍歷文字文件的內容，並在每個實例中使用一條訊息來通知使用者，考量的物件是段落還是表格。

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

' 建立文件物件
Doc = StarDesktop.CurrentComponent

' 建立列舉型物件
Enum = Doc.Text.createEnumeration

' 迴路遍歷全部文字元素
While Enum.hasMoreElements
	TextElement = Enum.nextElement

	If TextElement.supportsService("com.sun.star.text.TextTable") Then
		MsgBox "The current block contains a table."
	End If

	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		MsgBox "The current block contains a paragraph."
	End If
Wend
```

此示例建立了參照目前 StarSuite 文件的 Doc 文件物件。借助 Doc，此示例隨後建立了一個 Enumeration 物件，此物件遍歷 文字 (段落和表格) 的各個部份，並將目前元素指定給 TextElement 物件。此示例使用了 supportsService 方法來檢查 TextElement 是段落還是表格，以傳送 相應的訊息。

段落

com.sun.star.text.Paragraph 服務可用於存取段落內容。使用 String 屬性可以擷取和修改段落中的文字：

```
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
```

此示例開啓目前的文字文件，並借助 Enumeration 物件遍歷此文件。它在全部段落中使用 TextElement.String 屬性來存取相關的段落，並使用 U、2 和 4 這三個字元來代替 you、too 和 for 這三個字串。
用於代替的 Replace 函數不屬於 StarSuite Basic 的標準語言範圍。這就是第 3 章搜尋和代替一節中所介紹的示例函數的一個實例。

此處介紹的用於存取文字段落的程序內容相當於 VBA 中所使用的 Paragraphs 清單，此清單由 VBA 中可用的 Range 物件和 Document 物件提供。在 VBA 中，段落是透過其編號 (例如，呼叫 Paragraph(1)) 來存取的，而在 StarSuite Basic 中，應該使用上述的 Enumeration 物件來存取。

在 StarSuite Basic 中，沒有與 VBA 中所提供的 Characters、Sentences 和 Words 等清單相對應的內容。但是，您可以選擇切換至 TextCursor，使用它可在字元、句子和字詞等層級中瀏覽 (請參閱 *TextCursor*)。

段落部份

前一個 示例可以依要求變更文字，但是有時也可能銷毀格式。

這是因爲段落也是由個別子物件組成的。這些子物件都包含自己的格式資訊。例如，如果段落中部包含以粗體列印的字詞，則在 StarSuite 中，此段落要由三個段落部份來表示：粗體類型前面的部份，然後是粗體字詞，最後是粗體類型後面的部份 (又變爲正常字型)。

現在，如果使用段落的 String 屬性變更段落的文字，那麼 StarSuite 將首先刪除舊的段落部份，然後插入新的段落部份。由此，先前段落部份的格式就遺失了。

爲了防止發生這種情況，使用者可以存取相關聯的段落部份，而不要存取整個段落。爲此，段落都提供了自己的 Enumeration 物件。以下示例顯示了一個雙迴路，此迴路遍歷文字文件的所有段落以及其中所包含的段落部份，並採用前一個示例中的替代程序：

```
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

' 迴路遍歷全部段落
While Enum1.hasMoreElements
	TextElement = Enum1.nextElement
	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = TextElement.createEnumeration

		' 迴路遍歷全部子段落
		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
```

此示例在文字文件中以雙迴路形式執行。外部迴路針對文字的段落；內部迴路處理這些段落中的段落部份。此示例程式碼使用字串的 `String` 屬性修改了這些段落部份的內容，所做修改與前一個示例中對段落的修改相同。由於可以直接編輯段落部份，因此在代替字串時，仍保留這些段落部份的格式資訊。

格式化

格式化文字有多種方法。 最簡單的方法是將格式屬性直接指定給文字序列，這稱爲直接格式化。直接格式化主要用在較短的文件中，因爲使用者可以使用滑鼠來指定格式。例如，可以使用粗體類型來反白顯示文字中的某個字詞，或置中一行。

除了直接格式化以外，還可以使用文件樣式來格式化文字，這稱爲間接格式化。使用間接格式化，使用者可以將預先定義的文件樣式指定給相關的文字部份。日後如果要變更文字的版式，只需變更文件樣式即可。StarSuite 將變更使用此文件樣式的所有文字部份的描述方式。

在 VBA 中，物件的格式屬性通常會延伸到子物件區域 (例如，`Range.Font`、`Range.Borders`、`Range.Shading` 和 `Range.ParagraphFormat`)。這些屬性是透過重疊表示式 (例如，`Range.Font.AllCaps`) 來存取的。另一方面，在 StarSuite Basic 中，使用相關的物件 (`TextCursor`、`Paragraph` 等) 時，也可以直接使用格式屬性。在以下兩節中，您將找到 StarSuite 中可用的字元屬性和段落屬性之摘要。

在舊的 StarSuite API 中，主要使用 Selection 物件及其下級物件 (例如，Selection.Font、Selection.Paragraph 和 Selection.Border) 格式化文字。在新的 API 中，每個物件都具有格式化屬性 (Paragraph、TextCursor 等)，這些屬性可以直接採用。在下面的段落中，可以找到可用的字元屬性和段落屬性之清單。

字元屬性

針對個別字元的那些格式屬性稱爲字元屬性。其中包括粗體類型和字型類型。允許設定字元屬性的物件必須支援 com.sun.star.style.CharacterProperties 服務。StarSuite 可以識別支援此服務的一整套服務，其中包括上述用於段落的 com.sun.star.text.Paragraph 服務以及用於段落部份的 com.sun.star.text.TextPortion 服務。

com.sun.star.style.CharacterProperties 服務不提供任何介面，但提供了一系列屬性，透過這些屬性可以定義和呼叫字元屬性。在 StarSuite API 參照中，包含了所有字元屬性的完整清單。以下清單介紹了最重要的屬性：

CharFontName (字串型**) -** 所選字型類型的名稱。

CharColor (長型**) -** 文字顏色。

CharHeight (浮點型**) -** 字元高度，以點 (pt) 爲單位。

CharUnderline (常數型群組**) -** 底線的類型 (常數與 com.sun.star.awt.FontUnderline 一致)。

CharWeight (常數型群組**) -** 字型粗細 (常數與 com.sun.star.awt.FontWeight 一致)。

CharBackColor (長型**) -** 背景顏色。

CharKeepTogether (布林型**) -** 禁止自動換行。

CharStyleName (字串型**) -** 字元文件樣式的名稱。

段落屬性

不針對個別字元而針對整個段落的格式資訊視爲段落屬性，其中包括段落與頁面邊緣的間隔以及段落的行距。可以透過 com.sun.star.style.ParagraphProperties 服務來使用段落屬性。

甚至可以在各種物件中使用段落屬性。支援 com.sun.star.text.Paragraph 服務的所有物件也支援 com.sun.star.style.ParagraphProperties 中的段落屬性。

在 StarSuite API 參照中，包含了段落屬性的完整清單。最常用的段落屬性爲：

ParaAdjust (列舉型**) -** 垂直書寫方向 (常數與 com.sun.star.style.ParagraphAdjust 一致)。

ParaLineSpacing (結構型**) -** 行距 (結構與 com.sun.star.style.LineSpacing 一致)。

ParaBackColor (長型**) -** 背景顏色。

ParaLeftMargin (長型**) -** 左邊距，以百分之一公釐爲單位。

ParaRightMargin (長型**) -** 右邊距，以百分之一公釐爲單位。

ParaTopMargin (長型**) -** 上邊距，以百分之一公釐爲單位。

ParaBottomMargin (長型**) -** 下邊距，以百分之一公釐爲單位。

ParaTabStops (結構型陣列**) -** 定位鍵的類型和位置 (具有 `com.sun.star.style.TabStop` 結構的陣列)。

ParaStyleName (字串型**) -** 段落文件樣式的名稱。

示例：簡單的 HTML 匯出

以下示例說明了如何使用格式資訊。此示例將遍歷整個文字文件，並建立簡單的 HTML 檔案。爲此，該文件以自己的 HTML 元素 <P> 記錄每個段落。匯出時，使用 HTML 元素 <B> 標記以粗體類型顯示的段落部份。

```
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

' 迴路遍歷全部段落
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration
              CurLine = "<P>"

              ' 迴路遍歷全部段落部份
              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

              ' 匯出此行
              CurLine = CurLine & "</P>"
              Print #FileNo, CurLine

       End If
Wend

' 寫入 HTML 頁尾
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

此示例的基本結構是爲了便於示例遍歷前面所討論的段落部份。其中加入了寫入 HTML 檔案所需的函數，還加入了測試程式碼，該程式碼可檢查相應文字部份字型粗細，並爲粗體類型的段落部份提供了對應的 HTML 標記。

字元屬性和段落屬性的標準值

直接格式化總是優先於間接格式化。換句話說，在文字中，使用文件樣式的格式化比直接格式化的優先度要低。

要確定文件的某個部份採用的是直接格式化還是間接格式化並不容易。StarSuite 所提供的圖示列顯示了常用的文字屬性，例如字型類型、粗細和大小。但是，文字中相應的設定是基於文件樣式，還是基於直接格式化仍然不清楚。

StarSuite Basic 提供了 `getPropertyState` 方法，程式設計師可以使用此方法來檢查某個屬性是如何格式化的。該方法以屬性名稱作爲參數，傳回一個常數，此常數可提供有關格式來源的資訊。可能出現在 `com.sun.star.beans.PropertyState` 列舉中定義的以下回應：

`com.sun.star.beans.PropertyState.DIRECT_VALUE`- 在文字中直接定義的屬性 (直接格式化)，

`com.sun.star.beans.PropertyState.DEFAULT_VALUE`- 透過文件樣式定義的屬性 (間接格式化)

`com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE`- 屬性不清楚。
例如，一個段落中既有粗體字型的字詞，又有正常字型的字詞，在查詢該段落的粗體類型屬性時，就會出現這種狀態。

以下示例顯示了如何在 StarSuite 中編輯格式屬性。此示例將搜尋文字中那些直接格式化爲粗體類型的段落部份。如果遇到相應的段落部份，將使用 `setPropertyToDefault` 方法刪除直接格式化，並對這些段落部份指定 `MyBold` 字元文件樣式。

```
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

' 迴路遍歷全部段落
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration

              ' 迴路遍歷全部段落部份
              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
```

編輯文字文件

前一節已經討論了用於編輯文字文件的一整套選項， 重點介紹了 `com.sun.star.text.TextPortion` 服務和 `com.sun.star.text.Paragraph` 服務，這些服務都可用於存取段落部份和段落。這些服務適用於那些要在一次迴路中對文字內容進行編輯的應用程式。但是，對於很多問題而言，這樣是不夠的。StarSuite 提供了 `com.sun.star.text.TextCursor` 服務，用以處理更複雜的工作，其中包括向後瀏覽文件，或者基於句子和字詞而不是 `TextPortions` 進行瀏覽。

TextCursor

StarSuite API 中的 `TextCursor` 相當於 StarSuite 文件中使用的可見游標。它在文字文件中標記出某一點，透過各種指令向不同的方向瀏覽。但是，不要混淆可見游標和 StarSuite Basic 中使用的 `TextCursor` 物件。它們完全是兩碼事。

> 警告：與 VBA 中所使用的術語不同：就功能範圍而言，VBA 的 `Range` 物件相當於 StarSuite 中的 `TextCursor` 物件，而不是 StarSuite 中的 `Range` 物件 (儘管它們的名稱相同)。
>
> 例如，StarSuite 中的 `TextCursor` 物件提供了瀏覽和變更文字的方法，VBA 中的 `Range` 物件也提供了這些方法 (例如，`MoveStart`、`MoveEnd`、`InsertBefore` 和 `InsertAfter`)。以下各節將介紹 StarSuite 中 `TextCursor` 物件的相關內容。

在文字中瀏覽

StarSuite Basic 中的 `TextCursor` 物件與文字文件中的可見游標是相互獨立作用的。在任何情況下，`TextCursor` 物件的程序設計控制位置之變更都不會影響到可見游標。您甚至可以爲同一文件開啓數個 `TextCursor` 物件，並可用於各個位置，這些物件之間相互獨立。

`TextCursor` 物件透過 `createTextCursor` 呼叫來建立：

```
Dim Doc As Object
Dim Cursor As Object

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

以這種方法建立的 `Cursor` 物件支援 `com.sun.star.text.TextCursor` 服務，因此，可提供一整套瀏覽文字文件的方法。以下示例首先將 `TextCursor` 向左移動十個字元，然後再向右移動三個字元：

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

`TextCursor` 可以反白顯示一個完整的區域。這相當於使用滑鼠來反白顯示文字中的某一點。以上函數呼叫中的 `False` 參數指定是否反白顯示游標移動所掠過的區域。例如，以下示例中的 `TextCursor`

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

首先向右移動十個字元但不反白顯示所掠過的區域，然後再向左移動三個字元並反白顯示此區域。因此，由 TextCursor 反白顯示的區域從文字中的第七個字元開始，到第十個字元爲止。

以下是 com.sun.star.text.TextCursor 服務所提供的用於瀏覽的主要方法：

goLeft (Count, Expand) - 向左跳過 Count 個字元。

goRight (Count, Expand) - 向右跳過 Count 個字元。

gotoStart (Expand) - 跳換到文字文件的開頭。

gotoEnd (Expand) - 跳換到文字文件的結尾。

gotoRange (TextRange, Expand) - 跳換到指定的 TextRange-Objekt。

gotoStartOfWord (Expand) - 跳換到目前字詞的開頭。

gotoEndOfWord (Expand) - 跳換到目前字詞的結尾。

gotoNextWord (Expand) - 跳換到下一個字詞的開頭。

gotoPreviousWord (Expand) - 跳換到前一個字詞的開頭。

isStartOfWord () - 如果 TextCursor 位於字詞的開頭，則傳回 True。

isEndOfWord () - 如果 TextCursor 位於字詞的結尾，則傳回 True。

gotoStartOfSentence (Expand) - 跳換到目前句子的開頭。

gotoEndOfSentence (Expand) - 跳換到目前句子的結尾。

gotoNextSentence (Expand) - 跳換到下一句的開頭。

gotoPreviousSentence (Expand) - 跳換到前一句的開頭。

isStartOfSentence () - 如果 TextCursor 位於句子的開頭，則傳回 True。

isEndOfSentence () - 如果 TextCursor 位於句子的結尾，則傳回 True。

gotoStartOfParagraph (Expand) - 跳換到目前段落的開頭。

gotoEndOfParagraph (Expand) - 跳換到目前段落的結尾。

gotoNextParagraph (Expand) - 跳換到下一個段落的開頭。

gotoPreviousParagraph (Expand) - 跳換到前一個段落的開頭。

isStartOfParagraph () - 如果 TextCursor 位於段落的開頭，則傳回 True。

isEndOfParagraph () - 如果 TextCursor 位於段落的結尾，則傳回 True。

文字依句子符號劃分成句。例如，句點被解譯爲指示句子結尾的符號。

Expand 參數是一個布林型值，用來指定是否要反白顯示瀏覽期間掠過的區域。另外，所有瀏覽方法都將傳回一個參數，以指定瀏覽是否成功，或指定此動作是否由於文字不足而終止。

以下清單中的數種方法，可用於編輯使用 TextCursor 反白顯示的區域，這些方法也支援 com.sun.star.text.TextCursor 服務：

collapseToStart () - 重設反白顯示，並將 TextCursor 定位在先前反白顯示的區域之開頭。

collapseToEnd () - 重設反白顯示，並將 TextCursor 定位在先前反白顯示的區域之結尾。

isCollapsed () - 如果 TextCursor 目前未覆蓋任何反白顯示的區域，則傳回 True。

使用 TextCursor 格式化文字

com.sun.star.text.TextCursor 服務支援本章開頭所提到的所有字元屬性和段落屬性。

以下示例顯示了如何將這些屬性與 TextCursor 配合使用。
此示例將遍歷整個文件，並以粗體類型格式化每句的第一個字詞。

```
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
```

此示例首先爲剛開啓的文字建立了一個文件物件，然後，逐句遍歷全部文字，反白顯示各句的第一個字詞，並以粗體格式化這些字詞。

擷取和修改文字內容

如果 TextCursor 包含反白顯示的區域，則可透過 TextCursor 物件的 String 屬性使用這些文字。以下示例將使用 String 屬性在訊息方塊中顯示句子的第一個字詞：

```
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
```

可以使用 String 屬性以同樣的方式修改每句的第一個字詞：

```
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
```

如果 TextCursor 包含反白顯示的區域，則指定的 String 屬性將使用新文字來代替此區域。如果未包含反白顯示的區域，則此文字會插入到 TextCursor 目前的位置。

插入控制碼

有時，需要修改的不是文件的實際文字，而是文件的結構。爲此，StarSuite 提供了控制碼。在文字中插入這些程式碼，會影響到文字的結構。控制碼是在常數的 `com.sun.star.text.ControlCharacter` 群組中定義的。在 StarSuite 中，以下控制碼可用：

`PARAGRAPH_BREAK` - 段落換行。

`LINE_BREAK` - 段落中換行。

`SOFT_HYPHEN` - 可能的分音節點。

`HARD_HYPHEN` - 必需的分音節點。

`HARD_SPACE` - 在左右對齊的文字中，未展開或壓縮的受保護的空格。

若要插入控制碼，您不僅需要游標，還需要相關聯的文字文件物件。以下示例將在文字的第 20 個字元後插入段落：

```
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)
```

呼叫 `insertControlCharacter` 方法時所用到的參數 `False`，確保了在插入作業完成後，目前由 `TextCursor` 反白顯示的區域會保留下來。如果此處傳送的是參數 `True`，那麼 `insertControlCharacter` 將代替目前的文字。

搜尋文字部份

在許多情況下，需要在文字中搜尋特定的術語，並編輯相應的位置。爲此，所有的 StarSuite 文件都提供了特殊的介面，此介面總是依照同一原則起作用：通常，在開始搜尋前必須先建立一個所謂的 `SearchDescriptor`，它定義了 StarSuite 在文件中要搜尋的內容。`SearchDescriptor` 是一個物件，支援 `com.sun.star.util.SearchDescriptor` 服務，並可透過文件的 `createSearchDescriptor` 方法來建立：

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

`SearchDescriptor` 建立後，會接收要搜尋的文字：

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

就其功能而言，`SearchDescriptor` 最接近於 StarSuite 的搜尋對話方塊。與搜尋視窗類似，在 `SearchDescriptor` 物件中可以設定搜尋所需的設定。

com.sun.star.util.SearchDescriptor 服務提供的屬性包括：

SearchBackwards (布林型**)** - 向後搜尋文字，而不是向前搜尋。

SearchCaseSensitive (布林型**)** - 搜尋期間考量字元的大小寫。

SearchRegularExpression (布林型**)** - 處理搜尋表示式的方式與處理常規表示式的方式相同。

SearchStyles (布林型**)** - 在文字中搜尋指定的段落文件樣式。

SearchWords (布林型**)** - 僅搜尋完整的字詞。

在 StarSuite Basic 中，也可以使用 StarSuite SearchSimilarity (或「模糊符合」) 功能。使用此功能，StarSuite 將搜尋與搜尋表示式類似但不完全相同的表示式。可以爲這些表示式分別定義要其他的、刪除的或修改的字元數。以下是與 com.sun.star.util.SearchDescriptor 服務相關聯的屬性：

SearchSimilarity (布林型**)** - 執行相似字搜尋。

SearchSimilarityAdd (短型**)** - 在相似字搜尋中可以新增的字元數。

SearchSimilarityExchange (短型**)** - 在相似字搜尋中可以代替的字元數。

SearchSimilarityRemove (短型**)** - 在相似字搜尋中可以移除的字元數。

SearchSimilarityRelax (布林型**)** - 對於搜尋表示式，同時考量所有的變化規則。

依要求準備好 SearchDescriptor 後，便可將其採用至文字文件。爲此，StarSuite 文件提供了 findFirst 方法和 findNext 方法：

```
Found = Doc.findFirst (SearchDesc)

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

此示例透過一次迴路找到所有符合的項目，並傳回 TextRange 物件，此物件參照找到的文字訊息。

示例：相似字搜尋

此示例顯示了如何在文字中搜尋字詞「turnover」，並以粗體類型格式化搜尋結果。使用相似字搜尋，不僅可以找到字詞「turnover」，還可以找到其複數形式「turnovers」以及詞尾發生變化的字詞，例如「turnover's」。找到的表示式與搜尋表示式最多有兩個字母不同：

```
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
```

在 StarSuite 中，搜尋和代替的基本思想與 VBA 中的幾乎相同。這兩種介面都爲您提供了一個物件，透過此物件可以定義用於搜尋和代替的屬性。然後可將該物件採用至所需的文字區域以執行此動作。在 VBA 中，可以透過 `Range` 物件的 `Find` 屬性得到重要的輔助物件，而在 StarSuite Basic 中，則要透過文件物件的 `createSearchDescriptor` 呼叫或 `createReplaceDescriptor` 呼叫來建立此輔助物件。甚至，可用的搜尋屬性和方法也有所不同。

與 StarSuite 的舊 API 一樣，在新的 API 中，也是使用文件物件來執行搜尋和代替文字的。鑒於以前有一個名爲 `SearchSettings` 的物件專門用於定義搜尋選項，現在，使用 `SearchDescriptor` 物件來執行物件搜尋，或使用 `ReplaceDescriptor` 物件來自動代替文字。這些物件不僅包含選項，而且包含目前的搜尋文字，如果需要，還可以包含相關聯的文字替代。描述元物件使用文件物件建立，依照相關的請求完成，然後作爲搜尋方法的參數傳回文件物件中。

代替文字部份

在 StarSuite Basic 中，除了使用 StarSuite 的搜尋功能，還使用 StarSuite 的替代功能。這兩種功能的處理方法相同。替代程序首先也需要一個特殊物件，用於記錄程序的參數。此物件稱爲 `ReplaceDescriptor`，它支援 `com.sun.star.util.ReplaceDescriptor` 服務。`ReplaceDescriptor` 也支援以上段落中所介紹的 `SearchDescriptor` 的所有屬性。例如，在替代程序中，可以啓動或關閉區分大小寫功能，還可以執行相似字搜尋。

以下示例說明了如何使用 `ReplaceDescriptors` 在 StarSuite 文件中搜尋。

```
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
```

搜尋和代替所使用的表示式是透過 `ReplaceDescriptors` 的 `SearchString` 屬性和 `ReplaceString` 屬性設定的。實際的替代程序最終使用文件物件的 `replaceAll` 方法實現，此方法將代替搜尋表示式的所有搜尋結果。

示例：使用常規表示式搜尋和代替文字

StarSuite 的替代功能在與常規表示式一同使用時特別有效。這樣可提供一個選項，即使用萬用字元和特殊字元來定義變數搜尋表示式，而不使用固定值。

在 StarSuite 的線上說明章節中，會詳細介紹 StarSuite 所支援的常規表示式。以下是幾個示例：

搜尋表示式中的句點代表任意字元。因此，搜尋表示式 `sh.rt` 既可以代表 `shirt`，又可以代表 `short`。

字元 `^` 用於標記段落的開頭。因此，要搜尋所有出現在段落開頭的名稱 `Peter`，可以使用搜尋表示式 `^Peter`。

字元 `$` 用於標記段落的結尾。因此，要搜尋所有出現在段落結尾的名字 `Peter`，可以使用搜尋表示式 `Peter$`。

`*` 表示在此符號前面可以有若干字元。它可以與句點組合使用，作爲代表任意字元的萬用字元。例如，表示式 `temper.*e` 可以代表表示式 `temperance` 和 `temperature`。

以下示例顯示了如何借助常規表示式 `^$` 來移除文字文件中的所有空行。

```
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)
```

文字文件：文字以外的內容

到目前爲止，本章僅討論了文字段落及其段落部份。但是，文字文件中還可能包含其他物件，例如表格、繪圖、文字欄位和目錄等。這些物件可以鎖定到文字中的任意位置。

由於有了這些常用功能，StarSuite 中的所有這些物件都支援一個名爲 `com.sun.star.text.TextContent` 的常用基本服務。此服務提供以下屬性：

`AnchorType` **(**列舉型**)** - 確定 `TextContent` 物件的鎖定類型 (標準值與 `com.sun.star.text.TextContentAnchorType` 列舉一致)。

`AnchorTypes` **(**列舉序列**)** - 支援特殊 `TextContent` 物件的所有 `AnchorTypes` 的列舉。

`TextWrap` **(**列舉型**)** - 確定 `TextContent` 物件周圍文字換行的類型 (標準值與 `com.sun.star.text.WrapTextMode` 列舉一致)。

這些 `TextContent` 物件還共用一些方法，尤其是建立、插入和刪除物件的方法。

使用文件物件的 `createInstance` 方法可建立新的 `TextContent` 物件。

使用文字物件的 `insertTextContent` 方法可插入物件。

使用 `removeTextContent` 方法可刪除 `TextContent` 物件。

下面各節將介紹一系列使用這些方法的示例。

表格

以下示例將借助前面所介紹的 `createInstance` 方法建立表格。

```
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)
```

建立表格後，使用 `initialize` 呼叫將表格設定爲要求的列數和欄數，然後使用 `insertTextContent` 將其插入文字文件中。

正如示例所示，`insertTextContent` 方法不僅需要將要插入 `Content` 物件，還需要其他兩個參數：

`Cursor` 物件，用於確定插入位置

布林型變數，用於指定是用 `Content` 物件代替游標目前選擇的內容 (`True` 值)，還是在目前選擇的文字前插入 `Content` 物件 (`False` 值)

在文字文件中建立和插入表格時，StarSuite Basic 使用的物件與 VBA 中使用的物件相似：文件物件和 StarSuite Basic 中的 `TextCursor` 物件 (在 VBA 中是 `Range` 物件)。在 VBA 中，`Document.Tables.Add` 方法負責建立和設定表格，而在 StarSuite Basic 中，從上面的示例可以看出，表格要使用 `createInstance` 建立，並要透過 `insertTextContent` 初始化並插入文件中。

可以使用一個簡單的迴路來確定插入到文字文件中的表格。爲此，可使用文字文件物件的 `getTextTables()` 方法：

```
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)

      ' 編輯表格
Next I
```

在 StarSuite 6 中，可透過文件物件的 `TextTables` 清單使用文字表格。此清單取代了以前由 `Selection` 物件提供的表格清單。以上的示例顯示了建立文字表格的方法。存取文字表格所需的選項將在下一節介紹。

編輯表格

表格是由各個列組成的，而列是由各個儲存格組成的。嚴格地講，StarSuite 中並沒有表格欄。表格欄是由相鄰的列 (一列接一列) 隱含地組合而成。但是，爲了簡化對表格的存取，StarSuite 提供了一些使用欄進行操作的方法。如果表格中未合併儲存格，則這些方法是有用的。

讓我們先瞭解一下表格自身的屬性。這些屬性是在 `com.sun.star.text.TextTable` 服務中定義的。以下清單包含了表格物件最重要的屬性：

`BackColor (`長型**`)`** - 表格的背景顏色。

`BottomMargin (`長型**`)`** - 下邊距，以百分之一公釐爲單位。

`LeftMargin (`長型**`)`** - 左邊距，以百分之一公釐爲單位。

`RightMargin (`長型**`)`** - 右邊距，以百分之一公釐爲單位。

`TopMargin (`長型**`)`** - 上邊距，以百分之一公釐爲單位。

`RepeatHeadline (`布林型**`)`** - 在每個頁面上重複表格標題。

`Width (`長型**`)`** - 表格的絕對寬度，以百分之一公釐爲單位。

列

表格由一個列的清單組成。以下示例顯示了擷取和格式化表格列的方法。

```
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
```

此示例首先使用 `Table.getRows` 呼叫建立了一個包含所有列的清單。使用 `getCount` 方法和 `getByIndex` 方法可以進一步處理清單，這兩個方法都屬於 `com.sun.star.table.XtableRows` 介面。`getByIndex` 方法傳回一個列物件，此物件支援 `com.sun.star.text.TextTableRow` 服務。

以下是 `com.sun.star.table.XtableRows` 介面的主要方法：

`getByIndex(Integer)` - 依指定的索引傳回一個列物件。

getCount() - 傳回列物件的數目。

insertByIndex(Index, Count) - 在表格中在 Index 的位置處插入 Count 個列。

removeByIndex (Index, Count) - 在表格中在 Index 的位置處刪除 Count 個列。

getByIndex 方法和 getCount 方法適用於所有表格，而 insertByIndex 方法和 removeByIndex 方法僅適用於不包含合併的儲存格的表格。

com.sun.star.text.TextTableRow 服務提供了以下屬性：

BackColor (長型**)** - 列的背景顏色。

Height (長型**)** - 行高，以百分之一公釐爲單位。

IsAutoHeight (布林型**)** - 動態調整表格高度，使之與內容相符。

VertOrient (常數型**)** - 文字方塊的垂直方向，即有關表格中文字垂直方向的細節 (值與 com.sun.star.text.VertOrientation 一致)

欄

存取欄的方法與列相同，即對 Column 物件使用 getByIndex、getCount、insertByIndex 和 removeByIndex 等方法，此物件可透過 getColumns 取得。但是，這些方法僅適用於不包含合併的儲存格的表格。在 StarSuite Basic 中，無法依欄格式化儲存格，而必須使用格式化個別表格儲存格的方法來進行。

儲存格

在 StarSuite 文件中，每個儲存格都有唯一的名稱。當 StarSuite 的游標位於儲存格中時，此儲存格的名稱會顯示在狀態列上。通常，左上角的儲存格稱爲 A1，右下角的儲存格稱爲 Xn，其中 X 代表首欄的字母，n 代表最後一列的編號。透過表格物件的 getCellByName() 方法，可以使用儲存格物件。以下示例顯示了一個迴路，它遍歷表格的所有儲存格，並在各儲存格中輸入相應的列號和欄號。

```
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
```

表格儲存格相當於標準文字，它支援用於建立相關聯的 TextCursor 物件之 createTextCursor 介面。

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

因此，針對個別字元和段落的所有格式選項都可用於表格儲存格。

以下示例將對文字文件中的所有表格進行搜尋，並透過相應的段落屬性對包含數值型值的所有儲存格採用向右對齊的格式。

```
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
```

此示例建立了一個 `TextTables` 清單，其中包含在一個迴路中遍歷的所有文字表格。然後，StarSuite 爲這些表格中的每一個建立了相關聯儲存格名稱的清單，又在另一個迴路中依次遍歷這些清單。如果儲存格中包含數值型值，此示例就會相應地變更格式。爲達此目的，此示例先建立了一個 `TextCursor` 物件，用它來參照表格儲存格的內容，然後調整表格儲存格的段落屬性。

文字方塊

文字方塊和表格、圖形一樣，也被視爲 `TextContent` 物件。文字方塊主要由標準文字組成，可以放置到頁面上的任一位置，但在換行和分頁中並不包括文字方塊。

與所有 `TextContent` 物件一樣，文件中實際建立的文字方塊和插入的文字方塊是有區別的。

```
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)
```

建立文字方塊要使用文件物件的 `createInstance` 方法。以此方法建立的文字方塊，可以使用 `Text` 物件的 `insertTextContent` 方法插入到文件中。爲此，應該指定適當的 `com.sun.star.text.TextFrame` 服務之名稱。

文字方塊的插入位置由 `Cursor` 物件確定，而且在插入時也會執行此物件。

StarSuite 中的文字方塊與 Word 中的位置方塊相對應。而 VBA 使用 `Document.Frames.Add` 方法來建立位置方塊，在 VBA 中執行建立作業時，要借助上

述程序中的 TextCursor 方法和文件物件的 createInstance 方法。

文字方塊物件提供了一系列可以影響方塊之位置和事件的屬性。這些屬性大部份是在 com.sun.star.text.BaseFrameProperties 服務中定義的，每個 TextFrame 服務都支援這些屬性。主要屬性包括：

BackColor (長型**)** - 文字方塊的背景顏色。

BottomMargin (長型**)** - 下邊距，以百分之一公釐爲單位。

LeftMargin (長型**)** - 左邊距，以百分之一公釐爲單位。

RightMargin (長型**)** - 右邊距，以百分之一公釐爲單位。

TopMargin (長型**)** - 上邊距，以百分之一公釐爲單位。

Height (長型**)** - 文字方塊的高度，以百分之一公釐爲單位。

Width (長型**)** - 文字方塊的寬度，以百分之一公釐爲單位。

HoriOrient (常數型**)** - 文字方塊的水平方向 (與 com.sun.star.text.HoriOrientation 一致)。

VertOrient (常數型**)** - 文字方塊的垂直方向 (與 com.sun.star.text.VertOrientation 一致)。

以下示例將使用以上屬性建立文字方塊：

```
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)
```

此示例建立了 `TextCursor` 作爲文字方塊的插入標記。它位於文字的第一個字詞和第二個字詞之間。文字方塊是使用 `Doc.createInstance` 建立的。文字方塊的屬性已設定爲所需的起始值。

此處應該注意 `AnchorType` 屬性 (來源於 `TextContent` 服務) 和 `VertOrient` 屬性 (來源於 `BaseFrameProperties` 服務) 之間的互動關係。`AnchorType` 接收 `AS_CHARACTER` 的值，因此，文字方塊會直接插入到換行和分頁中，其行爲與字元一樣。例如，如果遇到換行，文字方塊會移到下一行。`VertOrient` 屬性的 `LINE_TOP` 值可確保文字方塊的上邊緣與字元的上邊緣高度相同。

完成初始化，最後，呼叫 `insertTextContent` 將文字方塊插入文字文件中。

爲了編輯文字方塊的內容，使用者應該使用前面已經再三提及的 TextCursor，它也可用於文字方塊。

```
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 = "This is a small Test!"
```

此示例建立了文字方塊，並將其插入目前的文件中，還爲此文字方塊開啓了一個 TextCursor。此游標用於將文字方塊的字型設定爲粗體類型，並將段落的方向設定爲置中。最後，爲文字方塊指定了「This is a small test!」字串。

文字欄位

文字欄位是 TextContent 物件，因爲它提供了超越純文字的其他邏輯。如果要在文字文件中插入文字欄位，可以使用與處理其他 TextContent 物件相同的方法：

```
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)
```

此示例將包含目前日期的文字欄位插入到目前文字文件的開頭。IsDate 屬性的 True 值只顯示日期而不顯示時間。IsFixed 的 False 值確保在開啓文件時自動更新日期。

在 VBA 中，欄位類型由 Document.Fields.Add 方法的參數指定，而在 StarSuite Basic 中，欄位類型由相應的服務之名稱定義。

以往，文字欄位是透過 StarSuite 中舊的 `Selection` 物件所提供的一整套方法 (例如 `InsertField`、`DELUserField` 和 `SetCurField`) 來存取的。

在 StarSuite 6 中，使用以物件爲導向的概念來管理欄位。若要建立文字欄位，應先使用需要的屬性建立所需類型的文字欄位，並將其初始化。然後，使用 `insertTextContent` 方法將文字欄位插入文件中。在前一個示例中，可以看到對應的來源文字。下面各節將介紹最重要的欄位類型及其屬性。

除了插入文字欄位以外，在文件中搜尋欄位也是一項重要工作。以下示例顯示了如何透過迴路來遍歷文字文件中所有的文字欄位，並檢查它們的相關類型。

```
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 "Date/time"
	ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
		MsgBox "Annotation"
	Else
		MsgBox "unknown"
	End If

Wend
```

若要建立文字欄位，首先需要文件物件的 `TextFields` 清單。此示例依此清單建立了一個 `Enumeration` 物件，透過此物件，可以在一次迴路中依次查詢所有的文字欄位。對於找到的文字欄位，使用 `supportsService` 方法檢查其支援的服務。如果證實此欄位是日期/ 時間欄位或是備註，則在資訊方塊中顯示相應的欄位類型。另一個方面，如果此示例找到其他欄位，則顯示的資訊爲「不明」。

下面，將爲您介紹最重要的文字欄位及其相關聯屬性的清單。`com.sun.star.text.TextField` 模組中的 API 參照提供了所有文字欄位的完整清單。(在 StarSuite Basic 中，列出文字欄位的服務名稱時，應該使用大寫字元和小寫字元，如前例所示。)

頁數、字數和字元數

文字欄位

`com.sun.star.text.TextField.PageCount`

`com.sun.star.text.TextField.WordCount`

`com.sun.star.text.TextField.CharacterCount`

可以傳回文字的頁數、字數和字元數。它們支援以下屬性：

`NumberingType (`常數型**`)`** - 編號格式 (依據 `com.sun.star.style.NumberingType` 中的常數準則)。

目前頁

可以使用 `com.sun.star.text.TextField.PageNumber` 文字欄位將目前頁的編號插入文件中。可以指定以下屬性：

`NumberingType (`常數型**`)`** - 編號格式 (依據 `com.sun.star.style.NumberingType` 中的常數準則)。

`Offset (`短型**`)`** - 爲頁碼加入的偏移 (可以指定負數)。

以下示例顯示了將頁數插入到文件頁尾的方法。

```
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)
```

此示例首先建立了一個支援 `com.sun.star.text.TextField.PageNumber` 服務的文字欄位。由於頁首和頁尾的行被定義爲 StarSuite 頁面文件樣式的一部份，因此首先使用所有 `PageStyles` 的清單來建立頁面樣式。

爲了確保頁尾行可見，將 `FooterIsOn` 屬性設定爲 `True`。然後，使用左側頁尾行的相關聯文字物件將文字欄位插入文件中。

備註

備註欄位 (com.sun.star.text.TextField.Annotation) 在文字中以黃色小圖示來表示。按一下此圖示，可開啓文字欄位，其中記錄了有關文字中目前位置的註解。備註欄位具有以下屬性。

Author (字串型**)** - 作者的姓名。

Content (字串型**)** - 註解文字。

Date (日期型**)** - 寫入備註的日期。

日期/ 時間

日期/ 時間欄位 (com.sun.star.text.TextField.DateTime) 表示目前日期或目前時間。它支援以下屬性：

IsFixed (布林型**)** - 如果爲 True，則保持插入時的時間細節不變；如果爲 False，則在每次開啓文件時進行更新。

IsDate (布林型**)** - 如果爲 True，則此欄位顯示目前的日期，否則顯示目前的時間。

DateTimeValue (結構型**)** - 欄位的目前內容 (com.sun.star.util.DateTime 結構)

NumberFormat (常數型**)** - 描述時間或日期所使用的格式。

章節名稱/ 章節編號

目前章節的名稱可以透過 com.sun.star.text.TextField.Chapter 類型的文字欄位取得。可以使用兩個屬性來定義此類型。

ChapterFormat (常數型**)** - 確定是否描述章節名稱或章節編號 (與 com.sun.star.text.ChapterFormat 一致)

Level (整數型**)** - 確定要顯示的章節名稱和/ 或章節編號的章節級。數值 0 代表可用的最高層級。

內文標籤

內文標籤 (服務 com.sun.star.text.Bookmark) 是 TextContent 物件。使用前面介紹的概念即可建立和插入內文標籤：

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

Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor()

Bookmark = Doc.createInstance("com.sun.star.text.Bookmark")
Bookmark.Name = "My bookmarks"
Doc.Text.insertTextContent(Cursor, Bookmark, True)
```

此示例建立了一個標記內文標籤插入位置的 Cursor 以及實際的內文標籤物件 (Bookmark)。接著，爲內文標籤指定了名稱，並透過 insertTextContent 將其插入到文件中游標所在的位置。

文字的內文標籤是透過名爲 Bookmarks 的清單來存取的，也可以透過其編號或名稱來存取。

以下示例顯示了如何在文字中尋找內文標籤，並在內文標籤所在的位置插入文字。

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

Doc = StarDesktop.CurrentComponent

Bookmark = Doc.Bookmarks.getByName("My bookmarks")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Here is the bookmark"
```

在此示例中，使用了 getByName 方法透過內文標籤的名稱尋找所需的內文標籤。createTextCursorByRange 呼叫隨後建立了一個 Cursor，它位於內文標籤的鎖定位置。然後，游標將所需文字插入此處。

第 7 章

工作表文件

StarSuite Basic 爲程式設計控制的建立和編輯工作表提供了豐富的介面。本章介紹如何控制工作表文件的相關服務、方法和屬性。

第一節介紹工作表文件的基本結構，說明如何存取和編輯個別儲存格的內容。

第二節重點介紹儲存格區域以及用於搜尋和代替儲存格內容的選項，從而集中說明如何有效率地編輯工作表。

Range 物件允許您定位任一表格區域，目前此物件已包含在新的 API 中。

基於表格的文件 (工作表) 之結構

工作表的文件物件是基於 `com.sun.star.sheet.SpreadsheetDocument` 服務的。這樣的每一個文件中都可能包含數個工作表。在本手册中，基於表格的文件或工作表文件是指整個文件，而工作表 (spreadsheet，簡寫爲 *sheet*) 是指文件中的工作表或表格。

對於工作表及其內容，在 VBA 和 SatrOffice Basic 中分別使用了不同的術語。文件物件在 VBA 中稱爲 *Workbook*，其包含的個別頁稱爲 *Worksheets*，而在 StarSuite Basic 中，這兩者分別稱爲 *SpreadsheetDocument* 和 *Sheet*。

工作表

您可以透過 `Sheets` 清單來存取工作表文件的個別工作表。

以下示例顯示了如何透過工作表編號或工作表名稱來存取工作表。

示例 **1**：透過編號存取 **(**編號從 **0** 開始**)**

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet  = Doc. Sheets (0)
```

示例 **2**：透過名稱存取

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
```

第一個示例是透過編號 (從 0 開始計數) 存取工作表的。第二個示例是透過工作表名稱和 `getByName` 方法存取工作表的。

透過 `getByName` 方法得到的 `Sheet` 物件支援 `com.sun.star.sheet.Spreadsheet` 服務。此服務除了提供數個用於編輯內容的介面以外，還提供以下屬性：

`IsVisible (`布林型`)` - 工作表是可見的。

`PageStyle (`字串型`)` - 工作表頁面文件樣式的名稱。

建立、刪除和重新命名工作表

還可以使用工作表文件的 `Sheets` 清單來建立、刪除和重新命名個別工作表。以下示例使用 `hasByName` 方法檢查名爲 *MySheet* 的工作表是否存在。如果存在，則此方法就使用 `getByName` 方法確定相應的物件參照，然後將參照儲存到 `Sheet` 內的變數中。如果相應的工作表不存在，則呼叫 `createInstance` 來建立它，並透過 `insertByName` 方法將其插入工作表文件中。

```
Dim Doc As Object
Dim Sheet As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)

If Doc.Sheets.hasByName("MySheet") Then
      Sheet = Doc.Sheets.getByName("MySheet")
Else
      Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
      Doc.Sheets.insertByName("MySheet", Sheet)
End If
```

`getByName` 方法和 `insertByName` 方法來源於 `com.sun.star.container.XnameContainer` 介面，第 4 章中已介紹過此介面。

列和欄

每個工作表都包含列和欄的清單，這些列和欄可以透過工作表物件的 Rows 和 Columns 屬性取得。列和欄支援 com.sun.star.table.TableColumns 服務和/ 或 com.sun.star.table.TableRows 服務。

以下示例將建立兩個物件 (這兩個物件分別參照工作表的第一列和第一欄)，並將參照儲存在 FirstCol 和 FirstRow 物件變數中。

```
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)
```

欄物件支援 com.sun.star.table.TableColumn 服務，此服務具有以下屬性：

Width (長型**)** - 欄寬，以百分之一公釐爲單位。

OptimalWidth (布林型**)** - 將欄設定爲最合適的寬度。

IsVisible (布林型**)** - 顯示欄。

IsStartOfNewPage (布林型**)** - 列印時，在欄前建立換頁。

僅當將 OptimalWidth 屬性設定爲 True 時，才能最適化欄寬。這時，如果變更個別儲存格的寬度，此儲存格所在欄的寬度不變。就功能而言，OptimalWidth 更像是一個方法而不是屬性。

列物件基於 com.sun.star.table.RowColumn 服務，此服務具有以下屬性：

Height (長型**)** - 列高，以百分之一公釐爲單位。

OptimalHeight (布林型**)** - 將列設定爲最合適的高度。

IsVisible (布林型**)** - 顯示列。

IsStartOfNewPage (布林型**)** - 列印時，在列前建立換頁。

如果將列的 OptimalHeight 屬性設定爲 True，則列中儲存格的高度變更時，列高將自動變更。除非透過 Height 屬性指定列的絕對高度，否則將一直採用自動最適化高度。

以下示例將對工作表前五列採用自動最適化高度，並使第二欄不顯示。

```
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
```

在 StarSuite Basic 中，可以透過索引存取 Rows 清單和 Columns 清單。與在 VBA 中不同的是，第一欄的索引是 0 而不是 1。

插入和刪除列與欄

工作表的 Rows 物件和 Columns 物件可以存取現有的列和欄，也可以將它們插入或刪除。

```
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)
```

此示例使用 insertByIndex 方法在工作表第四欄的位置 (索引 3，編號從 0 開始) 插入一個新欄。第二個參數指定要插入的欄數 (在此示例中爲一欄)。

removeByIndex 方法刪除了第六欄 (索引 5)。同樣，第二個參數指定要刪除的欄數。

用於插入和刪除列的方法使用 Rows 物件功能的方式，與此處所示的用於編輯欄的方法使用 Columns 物件的方式相同。

儲存格

工作表由包含儲存格的二維清單組成。每個儲存格的 X 位置和 Y 位置相對於左上角儲存格的位置 (0,0) 來定義，由此也定義了儲存格。

以下示例將建立一個參照左上角儲存格的物件，並在此儲存格中插入文字：

```
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"
```

除了數值座標以外，工作表中的每個儲存格都有名稱，例如，工作表左上角的儲存格 (0,0) 被稱爲 `A1`。字母 `A` 代表欄，數字 `1` 代表列。請勿混淆儲存格的名稱和位置，因爲名稱的列計數從 1 開始，而位置的計數從 0 開始。

在 StarSuite 中，表格儲存格可以爲空，也可以包含文字、數字或公式。儲存格的類型不由儲存在儲存格中的內容來確定，而由用於其所輸入內容的物件屬性來確定。使用 `Value` 屬性可以插入並呼叫數字，而對於文字和公式，則可分別使用 `String` 屬性和 `Formula` 屬性。

```
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"
```

此示例在欄位 A1 到 A3 中分別插入了一個數字、一個文字和一個公式。

`Value`、`String` 和 `Formula` 等屬性取代了用於設定表格儲存格之值的 `PutCell` 方法。

對於使用 `String` 屬性輸入的儲存格內容，StarSuite 會將其視爲文字，即使輸入的是數字也如此。用這種方法輸入的數字會在儲存格中向左對齊，而不是向右對齊。使用公式時，還要注意文字和數字的差別：

```
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
```

儘管儲存格 A1 中包含的數值是 100，儲存格 A2 中包含的數值是 1000，但是公式 A1+A2 卻傳回數值 100。這是因爲儲存格 A2 中的內容是作爲字串而不是作爲數字輸入的。

若要檢查儲存格的內容是數字還是字串，請使用 Type 屬性：

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
Cell = Sheet.getCellByPosition(1,1)

Cell.Value = 1000

Select Case Cell.Type
Case com.sun.star.table.CellContentType.EMPTY
      MsgBox "Content: Empty"
Case com.sun.star.table.CellContentType.VALUE
      MsgBox "Content: Value"
Case com.sun.star.table.CellContentType.TEXT
      MsgBox "Content: Text"
Case com.sun.star.table.CellContentType.FORMULA
      MsgBox "Content: Formula"
End Select
```

`Cell.Type` 屬性傳回 `com.sun.star.table.CellContentType` 列舉的值，此值可以識別儲存格的內容類型。可能的值包括：

EMPTY - 無值

VALUE - 數字

TEXT - 字串

FORMULA - 公式

插入、刪除、複製和移動儲存格

除了直接修改儲存格內容以外，StarSuite Calc 還提供了一個介面，用於插入、刪除、複製或合併儲存格。此介面 (com.sun.star.sheet.XRangeMovement) 可以透過工作表物件來使用，它提供了四種修改儲存格內容的方法。

insertCell 方法用於向工作表中插入儲存格。

```
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)
```

此示例在工作表文件中第一個工作表 (編號 0) 的第二欄和第二列的交叉處 (列、欄編號都爲 1) 插入一個大小爲兩列兩欄的儲存格範圍。指定的儲存格範圍中所有現有的值都移至此範圍下方。

若要定義插入的儲存格範圍，請使用 com.sun.star.table.CellRangeAddress 結構。此結構包括以下值：

Sheet (短型**)** - 工作表編號 (編號從 0 開始)。

StartColumn (長型**)** - 儲存格範圍中的第一欄 (編號從 0 開始)。

StartRow (長型**)** - 儲存格範圍中的第一列 (編號從 0 開始)。

EndColumn (長型**)** - 儲存格範圍中的最後一欄 (編號從 0 開始)。

EndRow (長型**)** - 儲存格範圍中的最後一列 (編號從 0 開始)。

必須將完整的 CellRangeAddress 結構作爲 insertCells 方法的第一個參數來傳送。insertCells 的第二個參數包含 com.sun.star.sheet.CellInsertMode 列舉的值，用於定義如何處理位於插入位置之前的值。CellInsertMode 列舉可識別以下值：

NONE - 目前值保留在其目前位置。

DOWN - 插入位置處和插入位置下方的儲存格向下移動。

RIGHT - 插入位置處和插入位置右側的儲存格向右移動。

ROWS - 插入位置之後的列向下移動。

COLUMNS - 插入位置之後的欄向右移動。

removeRange 方法與 insertCells 方法相對應，它可以從工作表中刪除在 CellRangeAddress 結構中定義的範圍。

```
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)
```

此示例從工作表中移除了儲存格範圍 B2:C3，然後將下面的儲存格向上移動 兩列。移動類型由以下 com.sun.star.sheet.CellDELMode 列舉的其中一個值來定義：

NONE - 目前值保留在其目前位置。

UP - 插入位置處和插入位置下方的儲存格向上移動。

LEFT - 插入位置處和插入位置右側的儲存格向左移動。

ROWS - 插入位置之後的列向上移動。

COLUMNS - 插入位置之後的欄向左移動。

XRangeMovement 介面提供了其他兩種方法，用於移動 (moveRange) 和複製 (copyRange) 儲存格範圍。以下示例將移動 B2:C3 範圍，使此範圍從 A6 開始：

```
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)
```

除了 CellRangeAdress 結構以外，moveRange 方法還需要使用 com.sun.star.table.CellAddress 結構來定義移動作業的目標範圍之原點。 CellAddress 方法提供了以下值：

Sheet (短型**)** - 工作表編號 (編號從 0 開始)。

Column (長型**)** - 已定位的欄之編號 (編號從 0 開始)。

Row (長型**)** - 已定位的列之編號 (編號從 0 開始)。

目標範圍中的儲存格內容總是透過 moveRange 方法來覆寫。
與 InsertCells 方法不同的是，removeRange 方法未提供用於執行自動移動的參數。

copyRange 方法所起的作用與 moveRange 方法相同，只是 copyRange 插入儲存格範圍的副本，而不是移動此範圍。

就其功能而言，StarSuite Basic 中的 insertCell、removeRange 和 copyRange 等方法相當於 VBA 中的 Range.Insert、Range.Delete 和 Range.Copy 等方法。在 VBA 中，這些方法適用於相應的 Range 物件，而在 StarSuite Basic 中，則適用於相關聯的 Sheet 物件。

格式化

工作表文件提供了格式化儲存格和頁面所需的屬性與方法。

儲存格屬性

用於格式化儲存格的選項有很多，例如指定文字的字型類型和大小。每個儲存格都支援 `com.sun.star.style.CharacterProperties` 服務和 `com.sun.star.style.ParagraphProperties` 服務，這兩種服務的主要屬性在第 6 章 (文字文件) 中已有過介紹。特殊的儲存格格式由 `com.sun.star.table.CellProperties` 服務來處理。此服務的主要屬性將在以下各節中介紹。

所有的已命名屬性既可用於個別儲存格，也可用於儲存格範圍。

> StarSuite API 中的 `CellProperties` 物件相當於 VBA 中的 `Interior` 物件，也用於定義特定於儲存格的屬性。

背景顏色和陰影

`com.sun.star.table.CellProperties` 服務提供了定義背景顏色和陰影所需的以下屬性：

`CellBackColor` (長型) - 表格儲存格的背景顏色。

`IsCellBackgroundTransparent` (布林型) - 將背景顏色設定爲透明。

`ShadowFormat` (結構型) - 爲儲存格指定陰影 (結構與 `com.sun.star.table.ShadowFormat` 一致)。

用於儲存格陰影的 `com.sun.star.table.ShadowFormat` 結構和詳細規格具有以下結構：

`Location` (列舉型) - 陰影的位置 (其值來源於 `com.sun.star.table.ShadowLocation` 結構)。

`ShadowWidth` (短型) - 陰影的大小，以百分之一公釐爲單位。

`IsTransparent` (布林型) - 將陰影設定爲透明。

`Color` (長型) - 陰影的顏色。

以下示例會在 B2 儲存格中寫入數字 1000，並使用 `CellBackColor` 屬性將背景顏色變更爲紅色，然後爲此儲存格建立一個向左下方偏移 1 公釐的淺灰色陰影。

```
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
```

對齊

StarSuite 爲您提供了變更表格儲存格中文字之對齊的各種功能。

以下屬性定義了文字水平方向和垂直方向的對齊：

HoriJustify (列舉型**)** - 文字水平方向的對齊 (其值來源於 `com.sun.star.table.CellHoriJustify`)

VertJustify (列舉型**)** - 文字垂直方向的對齊 (其值來源於 `com.sun.star.table.CellVertJustify`)

Orientation (列舉型**)** - 文字的方向 (其值與 `com.sun.star.table.CellOrientation` 一致)

IsTextWrapped (布林型**)** - 允許在儲存格內自動換行

RotateAngle (長型**)** - 文字的旋轉角度，以百分之一度爲單位

以下示例顯示了如何「重疊」儲存格的內容，以在儲存格的左上角重疊列印個別字元。這些字元不旋轉。

```
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
```

數字、日期和文字的格式

StarSuite 提供了一整套預先定義的日期格式和時間格式。每種格式都具有內部編號，可透過它使用 `NumberFormat` 屬性爲儲存格指定格式。StarSuite 提供了 `queryKey` 方法和 `addNew` 方法，使您可以存取現有的數字格式，也可以建立自己的數字格式。可以透過呼叫下列物件來存取這些方法：

```
NumberFormats = Doc.NumberFormats
```

格式是使用格式字串指定的，結構化格式字串的方式類似於結構化 StarSuite Basic 中格式函數的方式。然而，它們之間有一個主要差別：指令格式需要英文縮寫和作爲千位分隔符的小數點或字元，而 NumberFormats 物件的指令格式結構必須使用指定的國家/ 地區的縮寫。

以下示例將格式化儲存格 B2，以使數字顯示三位小數，並使用逗號作爲千位分隔符。

```
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
```

在 StarSuite Calc 中，**[**格式化儲存格**]**對話方塊提供了用於儲存格的不同格式選項之摘要。

頁面屬性

頁面屬性是一些格式選項，用於在頁面上定位文件內容以及每頁中重複出現的可視元素。這些屬性包括

- 紙張格式
- 頁面邊距

頁首和頁尾。

定義頁面格式的程序不同於定義其他形式的格式。可以直接定義和採用儲存格、段落和字元元素，使用頁面樣式也可以定義頁面格式，但需要間接採用頁面格式。例如，將頁首和頁尾加入到頁面樣式中。

下面各節介紹了用於工作表頁面的主要格式選項。這裡介紹的許多樣式同樣適用於文字文件。對兩種類型的文件都有效的頁面屬性是在 `com.sun.star.style.PageProperties` 服務中定義的。僅適用於工作表文件的頁面屬性則是在 `com.sun.star.sheet.TablePageStyle` 服務中定義的。

Microsoft Office 文件的頁面屬性 (頁面邊距、邊框等) 是透過 `PageSetup` 物件，在 `Worksheet` 物件級 (Excel) 或 `Document` 物件級 (Word) 定義的。
在 StarSuite 中，這些屬性使用頁面樣式來定義，而頁面樣式又與相關聯的文件連結。

頁面背景

`com.sun.star.style.PageProperties` 服務定義了頁面背景的以下屬性：

`BackColor` (長型) - 背景的顏色

`BackGraphicURL` (字串型) - 要使用的背景圖形之 URL

`BackGraphicFilter` (字串型) - 用於解譯背景圖形的篩選之名稱

`BackGraphicLocation` (列舉型) - 背景圖形的位置 (其值由 `com.sun.star.style.GraphicLocation` 列舉決定)

`BackTransparent` (布林型) - 使背景透明

頁面格式

頁面格式是使用 `com.sun.star.style.PageProperties` 服務的以下屬性定義的：

`IsLandscape` (布林型) - 橫向格式

`Width` (長型) - 頁面寬度，以百分之一公釐爲單位

`Height` (長型) - 頁面高度，以百分之一公釐爲單位

`PrinterPaperTray` (字串型) - 要使用的印表機紙張托盤的名稱

以下示例會將「標準」頁面樣式的頁面大小設定爲 DIN A5 橫向格式 (高 14.8 cm，寬 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
```

頁面邊距、邊框和陰影

com.sun.star.style.PageProperties 服務提供了以下屬性，用於調整頁面邊距、邊框和陰影：

LeftMargin (長型**)** - 頁面左邊距的寬度，以百分之一公釐爲單位

RightMargin (長型**)** - 頁面右邊距的寬度，以百分之一公釐爲單位

TopMargin (長型**)** - 頁面上邊距的寬度，以百分之一公釐爲單位

BottomMargin (長型**)** - 頁面下邊距的寬度，以百分之一公釐爲單位

LeftBorder (結構型**)** - 頁面邊框左側線條的規格 (com.sun.star.table.BorderLine 結構)

RightBorder (結構型**)** - 頁面邊框右側線條的規格 (com.sun.star.table.BorderLine 結構)

TopBorder (結構型**)** - 頁面邊框上方線條的規格 (com.sun.star.table.BorderLine 結構)

BottomBorder (結構型**)** - 頁面邊框下方線條的規格 (com.sun.star.table.BorderLine 結構)

LeftBorderDistance (長型**)** - 頁面左邊框與頁面內容之間的間隔，以百分之一公釐爲單位

RightBorderDistance (長型**)** - 頁面右邊框與頁面內容之間的間隔，以百分之一公釐爲單位

TopBorderDistance (長型**)** - 頁面上邊框與頁面內容之間的間隔，以百分之一公釐爲單位

BottomBorderDistance (長型**)** - 頁面下邊框與頁面內容之間的間隔，以百分之一公釐爲單位

ShadowFormat (結構型**)** - 頁面內容區域的陰影規格 (com.sun.star.table.ShadowFormat 結構)

以下示例會將「標準」頁面樣式的左、右邊框設定爲 1 公分。

```
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
```

頁首和頁尾

文件的頁首和頁尾構成頁面屬性的組成部份，是使用 `com.sun.star.style.PageProperties` 服務定義的。用於格式化頁首的屬性包括：

HeaderIsOn (布林型**)** - 啓動頁首

HeaderLeftMargin (長型**)** - 頁首與頁面左邊距之間的間隔，以百分之一公釐爲單位

HeaderRightMargin (長型**)** - 頁首與頁面右邊距之間的間隔，以百分之一公釐爲單位

HeaderBodyDistance (長型**)** - 頁首與文件正文之間的間隔，以百分之一公釐爲單位

HeaderHeight (長型**)** - 頁首的高度，以百分之一公釐爲單位

HeaderIsDynamicHeight (布林型**)** - 自動調整頁首高度，使之與內容相符。

HeaderLeftBorder (結構型**)** - 頁首框的左邊框之細節 (`com.sun.star.table.BorderLine` 結構)

HeaderRightBorder (結構型**)** - 頁首框的右邊框之細節 (`com.sun.star.table.BorderLine` 結構)

HeaderTopBorder (結構型**)** - 頁首框的上邊框之細節 (`com.sun.star.table.BorderLine` 結構)

HeaderBottomBorder (結構型**)** - 頁首框的下邊框之細節 (`com.sun.star.table.BorderLine` 結構)

HeaderLeftBorderDistance (長型**)** - 頁首內容與左邊框之間的間隔，以百分之一公釐爲單位

HeaderRightBorderDistance (長型**)** - 頁首內容與右邊框之間的間隔，以百分之一公釐爲單位

HeaderTopBorderDistance (長型**)** - 頁首內容與上邊框之間的間隔，以百分之一公釐爲單位

HeaderBottomBorderDistance (長型**)** - 頁首內容與下邊框之間的間隔，以百分之一公釐爲單位

HeaderIsShared (布林型**)** - 奇數頁和偶數頁的頁首內容相同 (請參閱 `HeaderText`、`HeaderTextLeft` 和 `HeaderTextRight`)

HeaderBackColor (長型**)** - 頁首的背景顏色

HeaderBackGraphicURL (字串型**)** - 要使用的背景圖形之 URL

HeaderBackGraphicFilter (字串型**)** - 用於解譯頁首背景圖形的篩選之名稱

HeaderBackGraphicLocation (列舉型**)** - 頁首背景圖形的位置 (其值由 `com.sun.star.style.GraphicLocation` 列舉決定)

HeaderBackTransparent (布林型**)** - 將頁首背景顯示爲透明

HeaderShadowFormat (結構型**)** - 頁首陰影的細節 (`com.sun.star.table.ShadowFormat` 結構)

用於格式化頁尾的屬性包括：

FooterIsOn (布林型**)** - 啓動頁尾

FooterLeftMargin (長型**)** - 頁尾與頁面左邊距之間的間隔，以百分之一公釐爲單位

FooterRightMargin (長型**)** - 頁尾與頁面右邊距之間的間隔，以百分之一公釐爲單位

FooterBodyDistance (長型**)** - 頁尾與文件正文之間的間隔，以百分之一公釐爲單位

FooterHeight (長型**)** - 頁尾高度，以百分之一公釐爲單位

FooterIsDynamicHeight (布林型**)** - 自動調整頁尾高度，使之與內容相符。

FooterLeftBorder (結構型**)** - 頁尾框的左邊框之細節 (`com.sun.star.table.BorderLine` 結構)

FooterRightBorder (結構型**)** - 頁尾框的右邊框之細節 (`com.sun.star.table.BorderLine` 結構)

FooterTopBorder (結構型**)** - 頁尾框的上邊框之細節 (`com.sun.star.table.BorderLine` 結構)

FooterBottomBorder (結構型**)** - 頁尾框的下邊框之細節 (`com.sun.star.table.BorderLine` 結構)

FooterLeftBorderDistance (長型**)** - 頁尾內容與左邊框之間的間隔，以百分之一公釐爲單位

FooterRightBorderDistance (長型**)** - 頁尾內容與右邊框之間的間隔，以百分之一公釐爲單位

FooterTopBorderDistance (長型**)** - 頁尾內容與上邊框之間的間隔，以百分之一公釐爲單位

FooterBottomBorderDistance (長型**)** - 頁尾內容與下邊框之間的間隔，以百分之一公釐爲單位

FooterIsShared (布林型**)** - 奇數頁和偶數頁的頁尾內容相同 (請參閱 `FooterText`、`FooterTextLeft` 和 `FooterTextRight`)

FooterBackColor (長型**)** - 頁尾的背景顏色

FooterBackGraphicURL (字串型**)** - 要使用的背景圖形之 URL

FooterBackGraphicFilter (字串型**)** - 用於解譯頁尾背景圖形的篩選之名稱

FooterBackGraphicLocation (列舉型**)** - 頁尾背景圖形的位置 (其值由 `com.sun.star.style.GraphicLocation` 列舉決定)

FooterBackTransparent (布林型**)** - 將頁尾背景顯示爲透明

FooterShadowFormat (結構型**)** - 頁尾陰影的細節 (`com.sun.star.table.ShadowFormat` 結構)

變更頁首和頁尾的文字

工作表中頁首和頁尾的內容可以透過以下屬性存取：

LeftPageHeaderContent (物件型**)** - 偶數頁中頁首的內容 (`com.sun.star.sheet.HeaderFooterContent` 服務)

RightPageHeaderContent (物件型**)** - 奇數頁中頁首的內容 (`com.sun.star.sheet.HeaderFooterContent` 服務)

LeftPageFooterContent (物件型**)** - 偶數頁中頁尾的內容 (`com.sun.star.sheet.HeaderFooterContent` 服務)

RightPageFooterContent (物件型**)** - 奇數頁中頁尾的內容 (`com.sun.star.sheet.HeaderFooterContent` 服務)

如果不需要區分奇數頁和偶數頁的頁首或頁尾 (`FooterIsShared` 屬性爲 `False`)，則請設定奇數頁中頁首和頁尾的屬性。

所有的已命名物件都傳回支援 `com.sun.star.sheet.HeaderFooterContent` 服務的物件。透過 (非真) 屬性 `LeftText`、`CenterText` 和 `RightText`，此服務爲 StarSuite Calc 的頁首和頁尾提供了三種文字元素。

以下示例將在「標準」文件樣式頁首的左側文字欄位中寫入一個值「Just a Test.」。

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object
Dim HContent As Object
Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Just a Test."
DefPage.RightPageHeaderContent = HContent
```

請注意示例的最後一行： 一旦變更了文字，就必須再次將 `TextContent` 物件指定給頁首，這樣變更才能生效。

由於頁首和頁尾是由單個文字區塊組成的，因此，對於文字文件 (StarSuite Writer)，還可以使用另一種機制來變更頁首和頁尾的文字。以下屬性是在 com.sun.star.style.PageProperties 服務中定義的：

HeaderText (物件型**)** - 包含頁首內容的文字物件 (com.sun.star.text.XText 服務)

HeaderTextLeft (物件型**)** - 包含左側頁面中頁首內容的文字物件 (com.sun.star.text.XText 服務)

HeaderTextRight (物件型**)** - 包含右側頁面中頁首內容的文字物件 (com.sun.star.text.XText 服務)

FooterText (物件型**)** - 包含頁尾內容的文字物件 (com.sun.star.text.XText 服務)

FooterTextLeft (物件型**)** - 包含左側頁面中頁尾內容的文字物件 (com.sun.star.text.XText 服務)

FooterTextRight (物件型**)** - 包含右側頁面中頁尾內容的文字物件 (com.sun.star.text.XText 服務)

以下示例將在文字文件的「標準」頁面樣式中建立一個頁首，並在頁首中加入文字「Just a Test」。

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Just a Test."
```

在此實例中，可以透過頁面樣式的 `HeaderText` 屬性直接進行存取，而不必使用 `HeaderFooterContent` 物件。

置中 (僅適用於工作表)

`com.sun.star.sheet.TablePageStyle` 服務僅用在 StarSuite Calc 頁面樣式中，它可將您要列印的儲存格範圍在頁面上置中。此服務提供了以下屬性：

`CenterHorizontally` (布林型) - 水平方向置中表格內容

`CenterVertically` (布林型) - 垂直方向置中表格內容

定義要列印的元素 (僅適用於工作表)

格式化工作表時，您可以定義頁面元素是否可見。爲此，`com.sun.star.sheet.TablePageStyle` 服務提供了以下屬性：

`PrintAnnotations` (布林型) - 列印儲存格的註解

`PrintGrid` (布林型) - 列印儲存格的格線

`PrintHeaders` (布林型) - 列印列和欄的標頭

`PrintCharts` (布林型) - 列印工作表中包含的圖表

`PrintObjects` (布林型) - 列印內嵌的物件

`PrintDrawing` (布林型) - 列印繪圖物件

`PrintDownFirst` (布林型) - 如果工作表的內容延伸數個頁面，則首先依垂直方向從上向下列印，然後列印右側的頁面。

`PrintFormulas` (布林型) - 列印公式，而不列印計算所得的值

`PrintZeroValues` (布林型) - 列印零值

有效率地編輯工作表文件

上一節已經介紹了工作表文件的主要結構，本節將爲您介紹一些服務，使用這些服務，您可以輕鬆地存取個別儲存格或儲存格範圍 。

儲存格範圍

StarSuite 除了提供用於個別儲存格的物件 (`com.sun.star.table.Cell` 服務) 以外，還提供表示儲存格範圍的物件。透過工作表物件的 `getCellRangeByName` 呼叫可以建立這樣的 `CellRange` 物件：

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C15")
```

在工作表文件中使用冒號 (:) 來指定儲存格範圍。例如，A1:C15 表示 A、B、C 三個欄中 1 到 15 列的所有儲存格。

可以使用 `getCellByPosition` 方法確定儲存格範圍中個別儲存格的位置，其中，此範圍中左上角儲存格的座標爲 (0,0)。以下示例將使用此方法建立儲存格 C3 的物件。

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Cell As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

格式化儲存格範圍

與個別儲存格一樣，您可以使用 `com.sun.star.table.CellProperties` 服務來格式化儲存格範圍。如需有關此服務的更多資訊和示例，請參閱格式化一節。

對儲存格範圍進行運算

您可以使用 `computeFunction` 方法對儲存格範圍執行數學運算。`computeFunction` 需要使用一個常數作爲參數，以描述要使用的數學函數。相關聯的常數是在 `com.sun.star.sheet.GeneralFunction` 列舉中定義的。下列值可用：

SUM - 全部數值型值的總和

COUNT - 全部值的總個數 (包括非數值型值)

COUNTNUMS - 全部數值型值的總個數

AVERAGE - 全部數值型值的平均值

MAX - 最大的數值型值

MIN - 最小的數值型值

PRODUCT - 全部數值型值的乘積

STDEV - 標準離差

VAR - 方差

STDEVP - 基於總體的標準離差

VARP - 基於總體的方差

以下示例將計算 A1:C3 範圍的平均值，並在訊息方塊中列印結果：

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

删除儲存格內容

`clearContents` 方法簡化了删除儲存格內容和儲存格範圍的程序，因爲它從儲存格範圍中删除特定類型的內容。

以下示例將從 B2:C3 範圍中移除全部字串和直接格式化資訊。

```
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)
```

`clearContents` 中指定的旗標來源於 `com.sun.star.sheet.CellFlags` 常數清單。此清單提供了以下元素：

VALUE - 未格式化爲日期或時間的數值型值

DATETIME - 已格式化爲日期或時間的數值型值

STRING - 字串

ANNOTATION - 連結至儲存格的註解

FORMULA - 公式

HARDATTR - 儲存格的直接格式化

STYLES - 間接格式化

OBJECTS - 連結至儲存格的繪圖物件

EDITATTR - 僅適用於部份儲存格的字元格式

您還可以將常數相加，以使用 `clearContents` 的呼叫來刪除不同資訊。

搜尋和代替儲存格內容

與文字文件一樣，工作表文件也提供搜尋和代替所需的功能。

在工作表文件中，用於搜尋和代替的描述元物件不是透過文件物件直接建立的，而是透過 `Sheets` 清單建立的。以下示例介紹了搜尋和代替的程序：

```
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
```

此示例使用文件的第一頁建立 `ReplaceDescriptor`，然後透過迴路將其採用至所有的頁面 。

第 **8** 章

繪圖和簡報

本章介紹了如何透過巨集建立和編輯繪圖。第一節介紹了繪圖的結構，其中包括構成繪圖的基本元素。第二節介紹了較爲複雜的編輯功能，例如分組物件、旋轉物件以及調整物件的顯示比例。

如需有關建立、開啓和儲存繪圖的資訊，請參閱第 5 章使用 *StarSuite* 文件。

繪圖的結構

StarSuite 對繪圖文件的頁數沒有限制，您可以單獨設計每一頁。它對加入頁面中的繪圖元素數目也沒有限制。

由於分層的使用，才使得圖片略顯複雜。依標準，每個繪圖文件都包含*[*版式*]*、*[*控制項*]*和*[*定量線*]*等分層，而且所有繪圖元素都會加入到*[*版式*]*分層中。您也可以選擇加入新的分層。如需有關繪圖分層的更多資訊，請參閱《StarSuite 開發者指南》。

頁面

`DrawPages` 清單提供了可用的繪圖文件頁面。您可以透過編號或名稱來存取個別頁面。如果有一個單頁文件，名爲 *Slide 1*，則以下兩個示例相同。

示例 **1**：

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
```

示例 **2**：

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Slide 1")
```

在示例 1 中，使用頁面編號 (從 0 開始計數) 來定位該頁面。而在第二個示例中，使用頁面名稱和 `getByName` 方法來存取該頁面。

```
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
```

以上呼叫會傳回一個支援 `com.sun.star.drawing.DrawPage` 服務的頁面物件。此服務識別以下屬性：

BorderLeft (長型**)** - 左邊框，以百分之一公釐爲單位

BorderRight (長型**)** - 右邊框，以百分之一公釐爲單位

BorderTop (長型**)** - 上邊框，以百分之一公釐爲單位

BorderBottom (長型**)** - 下邊框，以百分之一公釐爲單位

Width (長型**)** - 頁面寬度，以百分之一公釐爲單位

Height (長型**)** - 頁面高度，以百分之一公釐爲單位

Number (短型**)** - 頁數 (從 1 開始編號)，唯讀

Orientation (列舉型**)** - 頁面方向 (與 `com.sun.star.view.PaperOrientation` 列舉一致)

如果變更了這些設定，則會影響文件中的全部頁面。

以下示例會將剛才開啓的繪圖文件之頁面大小設定爲 20 × 20 公分，將頁面邊距設定爲 0.5 公分：

```
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
```

繪圖物件的基本屬性

繪圖物件包括形狀物件 (矩形、圓等)、線條物件和文字物件。所有這些物件共用一些常用功能，並且都支援 `com.sun.star.drawing.Shape` 服務。此服務可定義繪圖物件的 `Size` 屬性和 `Position` 屬性。

StarSuite Basic 還提供其他數種服務，用於修改格式之類的屬性或採用充填。格式選項是否可用，取決於繪圖物件的類型。

以下示例將在繪圖文件中建立並插入矩形：

```
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)
```

此示例使用 `StarDesktop.CurrentComponent` 呼叫，以確定哪一個文件已經開啓。使用這種方法確定的文件物件會透過呼叫 `drawPages(0)`，傳回繪圖的第一頁。

然後，初始化繪圖物件原點 (左角點) 和大小的 `Point` 與 `Size` 結構。長度以百分之一公釐爲單位。

接著，程式碼使用 Doc.createInstance 呼叫來建立 com.sun.star.drawing.RectangleShape 服務所指定的矩形繪圖物件。最後，使用 Page.add 呼叫將繪圖物件指定給頁面。

充填屬性

本節介紹四種服務，而且在每個實例中，示例程式碼都會使用矩形元素組合數種格式類型。充填屬性都組合到 com.sun.star.drawing.FillProperties 服務中。

對於充填區域，StarSuite 能識別四種主要的格式類型。最簡單的變體是單色充填。使用那些定義彩色圖案和陰影線的選項，可建立其他顏色以供使用。 第四個變體是將現有圖形投影到充填區域的選項。

繪圖物件的充填模式由 FillStyle 屬性定義。允許值在 com.sun.star.drawing.FillStyle 中定義。

單色充填

單色充填的主要屬性爲：

FillColor (長型**)** - 區域的充填顏色。

若要使用充填模式，必須將 FillStyle 屬性設定爲 SOLID 充填模式。

以下示例將建立矩形，並且使用紅色進行充填 (RGB 值爲 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)
```

彩色圖案

如果將 `FillStyle` 屬性設定爲 `GRADIENT`，則可對 StarSuite 文件的任一充填區域採用彩色圖案。

如果要採用預先定義的彩色圖案，只需指定相關聯的 `FillTransparenceGradientName` 屬性名稱即可。若要定義自己的彩色圖案，需要建立一個 `com.sun.star.awt.Gradient` 結構，以指定 `FillGradient` 屬性。此屬性提供以下選項：

`Style` **(**列舉型**)** - 彩色圖案類型，例如，線性或徑向 (標準值與 `com.sun.star.awt.GradientStyle` 一致)

StartColor (長型**)** - 彩色圖案的起始顏色

EndColor (長型**)** - 彩色圖案的最終顏色

Angle (短型**)** - 彩色圖案的角度，以十分之一度爲單位

XOffset (短型**)** - 彩色圖案的起點 X 座標，以百分之一公釐爲單位

YOffset (短型**)** - 彩色圖案的起點 Y 座標，以百分之一公釐爲單位

StartIntensity (短型**)** - `StartColor` 的强度，以百分比表示 (在 StarSuite Basic 中，也可以指定大於 100% 的值)

EndIntensity (短型**)** - `EndColor` 的强度，以百分比表示 (在 StarSuite Basic 中，也可以指定大於 100% 的值)

StepCount (短型**)** - StarSuite 用來計算彩色圖案的顏色等級數。

以下示例將借助 `com.sun.star.awt.Gradient` 結構來說明彩色圖案的用法：

```
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)
```

此示例建立了一個線性彩色圖案 (`Style` = `LINEAR`)。此彩色圖案由左上角的紅色 (`StartColor`) 開始，依 45 度角 (`Angle`) 延伸到右下角變爲綠色 (`EndColor`)。起始顏色和最終顏色的强度都爲 150% (`StartIntensity` 和 `EndIntensity`)，所產生的顏色看起來比在 `StartColor` 和 `EndColor` 屬性中指定的值要亮一些。此彩色圖案由一百種漸變的個別顏色 (`StepCount`) 繪製而成。

陰影線

若要建立陰影線充填，必須將 `FillStyle` 屬性設定爲 `HATCH`。用於定義陰影線的程式碼與定義彩色圖案的程式碼十分相似。同樣是輔助結構，此處使用 `com.sun.star.drawing.Hatch` 來定義陰影線的外觀。陰影線的結構具有以下屬性：

`Style` (列舉型) - 陰影線的類型：單線、方格或斜紋方格 (標準值與 `com.sun.star.awt.HatchStyle` 一致)

`Color` (長型) - 線條的顏色

`Distance` (長型) - 線條之間的間隔，以百分之一公釐爲單位

`Angle` (短型) - 陰影線的角度，以十分之一度爲單位

以下示例說明了陰影線結構的用法：

```
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)
```

此程式碼建立了一個單線陰影線結構 (`HatchStyle` = `SINGLE`)，其線條旋轉了 45 度 (`Angle`)。線條爲深灰色 (`Color`)，間距爲 0.2 公釐 (`Distance`)

點陣圖

若要使用點陣圖投影作爲充填，必須將 `FillStyle` 屬性設定爲 `BITMAP`。如果 StarSuite 中已有可用的點陣圖，則只需在 `FillBitMapName` 屬性中指定相應的名稱，並在 `FillBitmapMode` 屬性中指定顯示樣式，例如簡單、平鋪或拉伸 (標準值與 `com.sun.star.drawing.BitmapMode` 一致)。

如果要使用外部點陣圖檔案，可以在 `FillBitmapURL` 屬性中指定其 URL。

以下示例將建立矩形，並平鋪 StarSuite 中提供的 Sky 點陣圖，以充填矩形區域。

```
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)
```

透明

您可以調整所採用的任一充填的透明度。要變更繪圖元素的透明度，最簡單的方法就是使用 `FillTransparence` 屬性

以下示例將建立一個透明度爲 50% 的紅色矩形。

```
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)
```

若要使充填透明，請將 `FillTransparence` 屬性設定爲 100。

除了 `FillTransparence` 屬性以外，`com.sun.star.drawing.FillProperties` 服務還提供 `FillTransparenceGradient` 屬性。此屬性用於定義透明圖案，以指定充填區域的透明度。

線條屬性

所有具有邊框線條的繪圖物件都支援 `com.sun.star.drawing.LineStyle` 服務。以下是此服務提供的一些屬性：

LineStyle (列舉型**)** - 線條類型 (標準值與 `com.sun.star.drawing.LineStyle` 一致)

LineColor (長型**)** - 線條顏色

LineTransparence (短型**)** - 線條透明度

LineWidth (長型**)** - 線條寬度，以百分之一公釐爲單位

LineJoint (列舉型**)** - 與連結點的過渡 (標準值與 `com.sun.star.drawing.LineJoint` 一致)

以下示例將建立一個帶有實邊框 (LineStyle = SOLID) 的矩形，邊框線條寬 5 公釐 (LineWidth)，透明度爲 50%。線條的左右邊緣延伸並相交 (LineJoint = MITER) 成直角。

```
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)
```

除了以上列出的屬性以外，com.sun.star.drawing.LineStyle 服務還提供繪製點線和劃線所需的選項。如需更多資訊，請參閱 StarSuite API 參照。

文字屬性 (繪圖物件)

com.sun.star.style.CharacterProperties 服務和 com.sun.star.style.ParagraphProperties 服務可以格式化繪圖物件中的文字。這些服務與個別字元和段落有關，如需詳細資訊，請參閱第 6 章 (文字文件)。

以下示例將在矩形中插入文字，並格式化字型 (com.sun.star.style.CharacterProperties 服務)。

```
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 = "Das ist ein Test"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

此程式碼使用矩形的 String 屬性來插入文字，並使用 com.sun.star.style.CharacterProperties 服務中的 CharWeight 屬性和 CharFontName 屬性來格式化文字字型。

僅當繪圖物件加入到繪圖頁面中之後，才能插入文字。您還可以使用 com.sun.star.drawing.Text 服務來定位並格式化繪圖物件中的文字。以下爲此服務的一些重要屬性：

TextAutoGrowHeight (布林型) - 依繪圖元素所包含的文字調整該元素的高度

TextAutoGrowWidth (布林型**)** - 依繪圖元素所包含的文字調整採用該元素的寬度

TextHorizontalAdjust (列舉型**)** - 繪圖元素中文字的水平位置 (標準值與 com.sun.star.drawing.TextHorizontalAdjust 一致)

TextVerticalAdjust (列舉型**)** - 繪圖元素中文字的垂直位置 (標準值與 com.sun.star.drawing.TextVerticalAdjust 一致)

TextLeftDistance (長型**)** - 繪圖元素左側與文字左側之間的間隔，以百分之一公釐爲單位

TextRightDistance (長型**)** - 繪圖元素右側與文字右側之間的間隔，以百分之一公釐爲單位

TextUpperDistance (長型) - 繪圖元素頂端與文字頂端之間的間隔，以百分之一公釐爲單位

TextLowerDistance (長型**)** - 繪圖元素底端與文字底端之間的間隔，以百分之一公釐爲單位

以下示例說明了已命名屬性的用法。

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
```

```
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "This is a test"  ' 僅在執行 Page.add 之後執行!

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
```

此程式碼在頁面中插入了繪圖元素，然後使用 `TextVerticalAdjust` 屬性和 `TextHorizontalAdjust` 屬性將文字加入到繪圖物件的左上角。繪圖物件邊緣與文字邊緣之間的最小間隔設定爲三公釐。

陰影屬性

可以使用 `com.sun.star.drawing.ShadowProperties` 服務爲大多數繪圖物件加上陰影。此服務具有以下屬性：

Shadow (布林型**)** - 啓動陰影

ShadowColor (長型**)** - 陰影顏色

ShadowTransparence (短型**)** - 陰影的透明度

ShadowXDistance (長型**)** - 陰影到繪圖物件的垂直距離，以百分之一公釐爲單位

ShadowYDistance (長型**)** - 陰影到繪圖物件的水平距離，以百分之一公釐爲單位

以下示例將建立一個帶有陰影的矩形，陰影在水平方向和垂直方向上相對矩形各偏移 2 公釐。陰影以深灰色描繪，透明度爲 50%。

```
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)
```

各種繪圖物件的摘要

矩形

矩形物件 (`com.sun.star.drawing.RectangleShape`) 支援以下用於格式化物件的服務：

充填屬性 - `com.sun.star.drawing.FillProperties`

線條屬性 - `com.sun.star.drawing.LineProperties`

文字屬性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)

陰影屬性 - `com.sun.star.drawing.ShadowProperties`

CornerRadius (長型**) -** 圓角半徑，以百分之一公釐爲單位

圓和橢圓

服務 `com.sun.star.drawing.EllipseShape` 用於圓和橢圓，它所支援的服務如下：

充填屬性 - `com.sun.star.drawing.FillProperties`

線條屬性 - `com.sun.star.drawing.LineProperties`

文字屬性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)

陰影屬性 - `com.sun.star.drawing.ShadowProperties`

除了以上服務以外，圓和橢圓還提供下列屬性：

CircleKind (列舉型**)** - 圓或橢圓的類型 (標準值與 com.sun.star.drawing.CircleKind 一致)

CircleStartAngle (長型**)** - 起始角度，以十分之一度爲單位 (僅適用於圓弧段或橢圓弧段)

CircleEndAngle (長型**)** - 結束角度，以十分之一度爲單位 (僅適用於圓弧段或橢圓弧段)

CircleKind 屬性確定物件是完整的圓、圓缺還是部份圓。下列值可用：

com.sun.star.drawing.CircleKind.FULL - 完整的圓或橢圓

com.sun.star.drawing.CircleKind.CUT - 部份圓 (其接合部位直接相互連結的部份圓)

com.sun.star.drawing.CircleKind.SECTION - 圓缺

com.sun.star.drawing.CircleKind.ARC - 角度 (不包括圓形線條)

以下示例將建立 70 度的圓缺 (起始角度爲 20 度，結束角度爲 90 度)

```
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)
```

線條

StarSuite 爲線條物件提供了 `com.sun.star.drawing.LineShape` 服務。線條物件支援所有的一般格式服務 (平面除外)。以下是與 `LineShape` 服務相關聯的所有屬性：

線條屬性 - `com.sun.star.drawing.LineProperties`

文字屬性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)

陰影屬性 - `com.sun.star.drawing.ShadowProperties`

以下示例將借助已命名屬性建立一條線，並對其進行格式化。在 `Location` 屬性中指定線條的原點，而 `Size` 屬性中所列出的座標指定線條的端點。

```
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)
```

多重多邊形

StarSuite 透過 com.sun.star.drawing.PolyPolygonShape 服務，還支援複雜的多邊形。嚴格地講，多重多邊形 不是簡單的多邊形，而是由多個多邊形組成的多邊形。因此，可以指定並組合包含角點的數個獨立清單，來構成一個完整的物件。

與矩形相似，多重多邊形也具有繪圖物件的所有格式屬性：

充填屬性 - com.sun.star.drawing.FillProperties

線條屬性 - com.sun.star.drawing.LineProperties

文字屬性 - com.sun.star.drawing.Text (包括 com.sun.star.style.CharacterProperties 和 com.sun.star.style.ParagraphProperties)

陰影屬性 - com.sun.star.drawing.ShadowProperties

PolyPolygonShape 服務還具有用於定義多邊形座標的屬性：

PolyPolygon (陣列型) - 包含多邊形座標的欄位 (雙精度型陣列，其點的類型爲 com.sun.star.awt.Point)

以下示例顯示了如何透過 PolyPolygonShape 服務定義三角形。

```
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)   ' 設定座標前，必須先執行 Page.add

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())
```

由於多邊形的點已定義爲絕對數值，因此無需指定多邊形的大小或起始位置。但是，您需要建立點的陣列，將此陣列併入第二個陣列中 (使用 Array (Coordinates ()) 呼叫)，然後將此陣列指定給多邊形。執行相應的呼叫前，必須先將多邊形插入文件中。

雙精度型陣列的定義允許透過合併數個多邊形來建立複雜的形狀。例如，您可以建立一個矩形，然後在其中插入另一個矩形，以在第一個矩形中建立一個孔：

```
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)   ' 設定座標前，必須先執行 Page.add

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())
```

對於區分哪些區域爲充填區域，哪些區域爲孔，StarSuite 採用了一條簡單的規則：外部形狀的邊緣總是作爲多重多邊形的外邊界。向內的下一條線條是此形狀的內邊框，並標記了第一個孔的邊界。如果向內還有另一條線，則該線條標記了充填區域的邊界。

圖形

此處所展示的最後的繪圖元素都是基於 `com.sun.star.drawing.GraphicObjectShape` 服務的圖形物件。這些物件可以在 StarSuite 中與任何圖形一同使用，其外觀可以透過各種屬性進行調整。

圖形物件支援的一般格式屬性有兩種：

文字屬性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)

陰影屬性 - `com.sun.star.drawing.ShadowProperties`

圖形物件還支援其他屬性：

`GraphicURL` (字串型) - 圖形的 URL

`AdjustLuminance` (短型) - 顏色的亮度，以百分比表示 (允許使用負值)

`AdjustContrast` (短型) - 對比度，以百分比表示 (允許使用負值)

`AdjustRed` (短型) - 紅色值，以百分比表示 (允許使用負值)

`AdjustGreen` (短型) - 綠色值，以百分比表示 (允許使用負值)

`AdjustBlue` (短型) - 藍色值，以百分比表示 (允許使用負值)

`Gamma` (短型) - 圖形的灰色係數值

`Transparency` (短型) - 圖形的透明度，以百分比表示

`GraphicColorMode` (列舉型) - 顏色模式，例如，標準、灰階、黑白 (標準值與 `com.sun.star.drawing.ColorMode` 一致)

以下示例顯示了將頁面插入圖形物件中的方法。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        ' 規格，没有實際意義，因爲後者的座標已綁定

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)
```

此程式碼插入了 `test.jpg` 圖形，並使用 `Adjust` 屬性調整其外觀。在此示例中，繪製的圖形呈 40% 的透明度，且未發生其他顏色轉換 (`GraphicColorMode` = `STANDARD`)。

編輯繪圖物件

分組物件

在許多情況下，將數個個別繪圖物件分組到一起，作爲一個大物件進行處理是有用的。

以下示例將組合兩個繪圖物件：

```
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
' 建立正方形繪圖元素
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' 建立圓繪圖元素
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)
' 組合正方形和圓這兩個繪圖元素
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' 置中組合的繪圖元素
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
```

此程式碼建立了矩形和圓，並將它們插入到頁面中。接著，又建立了一個支援 `com.sun.star.drawing.ShapeCollection` 服務的物件，並使用 `Add` 方法將矩形和圓加入此物件中。`ShapeCollection` 透過 `Group` 方法加入到頁面中，並傳回實際 `Group` 物件，該物件的編輯方法與個別 `Shape` 的編輯方法類似。

如果要格式化群組中的個別物件，請在將這些物件加入群組之前採用此格式。物件一旦位於群組中，就無法再修改。

旋轉和修剪繪圖物件

以上各節中介紹的所有繪圖物件都可以使用 `com.sun.star.drawing.RotationDescriptor` 服務來旋轉和修剪。

此服務提供以下屬性：

`RotateAngle (`長型**`)`** - 旋轉角度，以百分之一度爲單位

`ShearAngle (`長型**`)`** - 修剪角度，以百分之一度爲單位

以下示例將建立一個矩形，然後使用 `RotateAngle` 屬性將其旋轉 30 度：

```
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)
```

下一個示例建立的矩形與前一個示例中的矩形相同，但使用 ShearAngle 屬性將其修剪 30 度。

```
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)
```

搜尋和代替

與文字文件一樣，繪圖文件也提供搜尋和代替功能。此功能與第 6 章文字文件中所介紹的用於文字文件的相應功能類似。但是，在繪圖文件中，用於搜尋和代替的描述元物件不是直接透過文件物件建立的，而是透過相關聯的字元層級建立的。以下示例概述了繪圖中的替代過程：

```
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
```

此程式碼使用文件的第一個 DrawPage 來建立 ReplaceDescriptor，然後透過迴路將此描述元採用到繪圖文件的所有頁面中。

簡報

StarSuite 的簡報建立在繪圖文件的基礎上。簡報中的每一頁就是一張投影片。存取這些投影片的方法，與透過文件物件的 `DrawPages` 清單存取標準繪圖的方法相同。`com.sun.star.presentation.PresentationDocument` 服務 用於簡報文件，而且還能提供完整的 `com.sun.star.drawing.DrawingDocument` 服務。

使用簡報

除了 `Presentation` 屬性提供的繪圖功能以外，簡報文件還具有簡報物件，用於存取簡報的主要屬性和控制機制。例如，此物件可提供 `start` 方法，用來 啓動簡報。

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

此示例中所使用的程式碼建立了一個參照目前簡報文件的 `Doc` 物件，並建立了相關聯的簡報物件。此物件的 `start ()` 方法用於啓動示例，並執行螢幕簡報。

簡報物件可提供以下方法：

`start` - 啓動簡報

`end` - 結束簡報

`rehearseTimings` - 從頭開始啓動簡報，並建立其執行時間

還可以使用以下屬性：

`AllowAnimations (`布林型**`)`** - 在簡報中執行動畫

`CustomShow (`字串型**`)`** - 允許您指定簡報的名稱，以便在簡報中參照此名稱

`FirstPage (`字串型**`)`** - 簡報開始的投影片之名稱

`IsAlwaysOnTop (`布林型**`)`** - 簡報視窗總是作爲螢幕上的第一個視窗顯示

`IsAutomatic (`布林型**`)`** - 自動執行整個簡報

`IsEndless (`布林型**`)`** - 簡報一結束，就從開頭重新啓動

`IsFullScreen (`布林型**`)`** - 自動以全螢幕模式啓動簡報

`IsMouseVisible (`布林型**`)`** - 在簡報放映期間顯示滑鼠

`Pause (`長型**`)`** - 簡報放映結束後，空白螢幕的顯示時間

`StartWithNavigator (`布林型**`)`** - 啓動簡報時顯示助手視窗

`UsePn (`布林型**`)`** - 在簡報放映期間顯示指標

第 9 章

圖解 (圖表)

StarSuite 能將資料顯示爲圖解，此圖解可建立各種形式 (條圖、圓形圖、線條圖或其他元素) 的資料之間的圖形捷徑。資料可以顯示爲平面圖形或 3D 圖形，而圖解元素的外觀可以分別調整，處理方法與繪圖元素類似。

如果此資料可使用工作表形式，就可以動態連結至圖解。在這種情況下，對基本資料所做的所有修改都能立即顯示在指定的圖解中。本章將概括介紹 StarSuite 的圖解模組之程式設計介面，並重點說明工作表文件中圖解的用法。

在工作表中使用圖解

在 StarSuite 中，並不將圖解視爲獨立的文件，而是視爲內嵌於現有文件中的物件。

儘管文字文件和繪圖文件中的圖解與文件內容相互獨立，但在工作表文件中提供了一種允許在文件資料和內嵌的圖解之間建立捷徑的機制。以下示例解釋了工作表文件和圖解之間的互動關係：

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
```

儘管示例中所使用的程式碼看起來很複雜，但主要程序只有三行：第一個主要行建立 Doc 文件變數，它參照目前的工作表文件 (`Doc 行 = StarDesktop.CurrentComponent`)。然後，示例中所使用的程式碼建立包含第一個工作表之全部圖表的清單 (`Charts 行 = Doc.Sheets(0).`

Charts)。最後，使用 addNewByName 方法向此清單的最後一行中加入新的圖表。這時，使用者就可以看到新圖表了。

最後一行初始化 Rect 和 RangeAddress 輔助結構，addNewByName 方法也將其作爲參數提供。Rect 確定圖表在工作表中的位置。RangeAddress 確定要連結至圖表的資料區域。

前一個示例建立了一個條形圖解。如果需要不同類型的圖形，則必須明確地代替條形圖解：

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

第一行定義相應的圖表物件。第二行使用新的圖解 (在此示例中爲線條圖解) 代替目前圖解。

在 Excel 中，作爲獨立頁插入到 Excel 文件中的圖表與內嵌於表格頁中的圖表是有區別的。相應地，此處爲圖表定義的存取方法也有所不同。但在 StarSuite Basic 中没有這種區別，因爲 StarSuite Calc 中的圖表總是作爲表格頁的內嵌物件來建立的。這些圖表總是透過相關聯的 Sheet 物件之 Charts 清單來存取。

圖解的結構

圖解的結構以及此圖解所支援的服務與介面清單，都取決於圖解類型。例如，Z 軸的方法和屬性僅適用於 3D 圖解，不適用於平面圖解。在圓形圖中，使用軸時没有介面。

圖解的各個元素

標題、分標題及關鍵字

標題、分標題及關鍵字構成每個圖解之基本元素的組成部份。對於每個元素，圖解都提供了自己的物件。爲了管理這些元素，Chart 物件提供了以下屬性：

HasMainTitle (布林型**)** - 啓動標題。

Title (物件型**)** - 包含有關圖解標題之詳細資訊的物件 (支援 com.sun.star.chart.ChartTitle 服務)。

HasSubTitle (布林型**)** - 啓動分標題。

Subtitle (物件型**)** - 包含有關圖解分標題之詳細資訊的物件 (支援 com.sun.star.chart.ChartTitle 服務)。

HasLegend (布林型**)** - 啓動關鍵字。

Legend (物件型**)** - 包含有關圖解關鍵字之詳細資訊的物件 (支援 com.sun.star.chart.ChartLegendPosition 服務)。

在許多方面，指定的元素與繪圖元素相對應。這是因爲 com.sun.star.chart.ChartTitle 服務和 com.sun.star.chart.ChartLegendPosition 都支援構成繪圖元素技術程式基礎的 com.sun.star.drawing.Shape 服務。

所以，使用者能夠使用 Size 和 Position 屬性來確定元素的位置和大小。

此外，爲格式化這些元素，還提供了充填屬性、線條屬性 (com.sun.star.drawing.FillProperties 服務和 com.sun.star.drawing.LineStyle 服務) 以及字元屬性 (com.sun.star.style.CharacterProperties 服務)。

com.sun.star.chart.ChartTitle 不僅包含已命名的格式屬性，還包含另外兩種屬性：

TextRotation (長型**)** - 文字的旋轉角度，以百分之一度爲單位。

String (字串型**)** - 顯示爲標題或分標題的文字。

圖解關鍵字 (com.sun.star.chart.ChartLegend 服務) 包含以下其他屬性：

Alignment (列舉型**)** - 關鍵字出現的位置 (標準值與 com.sun.star.chart.ChartLegendPosition 一致)。

以下示例將建立一個圖解，並爲其指定標題「Test」、分標題「Test 2」和關鍵字。關鍵字的背景爲灰色，位於圖解底端，字元大小爲 7 點。

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts
Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.HasMainTitle = True
Chart.Title.String = "Test"

Chart.HasSubTitle = True
Chart.Subtitle.String = "Test 2"

Chart.HasLegend = True
Chart.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Chart.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Chart.Legend.FillColor = RGB(210, 210, 210)
Chart.Legend.CharHeight = 7
```

背景

每個圖解都具有背景區域。每個區域都具有一個物件，可以使用圖解物件的以下屬性來存取該物件：

`Area (`物件型**`)`** - 圖解的背景區域 (支援 `com.sun.star.chart.ChartArea` 服務)。

圖解的背景覆蓋整個背景區域，其中包括標題、分標題以及圖解關鍵字背後的區域。相關聯的 `com.sun.star.chart.ChartArea` 服務支援線條屬性和充填屬性，不再提供更多其他屬性。

圖解牆和底板

儘管圖解背景覆蓋圖解的整個區域，但圖解背景牆僅限於資料區域正後方的區域。

3D 圖解通常具有兩面圖解牆：一面位於資料區域的後方，一面位於 Y 軸的左側。3D 圖解通常還具有底板。

`Floor (`物件型**`)`** - 圖解的底板面板 (僅適用於 3D 圖解，支援 `com.sun.star.chart.ChartArea` 服務)。

`Wall (`物件型**`)`** - 圖解牆 (僅適用於 3D 圖解，支援 `com.sun.star.chart.ChartArea` 服務)。

指定的物件支援 `com.sun.star.chart.ChartArea` 服務，因此該物件可提供一般的充填屬性和線條屬性 (`com.sun.star.drawing.FillProperties` 服務和 `com.sun.star.drawing.LineStyle` 服務，請參閱第 8 章)。

圖解牆和底板可透過 `Chart` 物件進行存取，而此 `Chart` 物件又構成 `Chart` 物件的組成部份：

```
Chart.Area.FillBitmapName = "Sky"
```

以下示例顯示了如何將 StarSuite 中的已有圖形 (名爲 Sky) 用作圖解的背景。

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = "Sky"
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

軸

StarSuite 能識別可在圖解中使用的五種不同的軸。在最簡單的分析藍本中，包含 X 軸和 Y 軸。使用 3D 圖解時，有時也提供 Z 軸。對於其中各資料列的值相差很大的圖解，StarSuite 爲第二顯示比例作業提供了第二 X 軸和 Y 軸。

第一 X 軸、Y 軸和 Z 軸

除了實際的軸以外，第一 X 軸、Y 軸和 Z 軸也都可以有標題、描述、網格以及輔助網格。有一個選項可用於顯示或隱藏所有這些元素。爲了管理這些功能，圖解物件提供了以下屬性 (以 X 軸爲例，Y 軸和 Z 軸的屬性結構與之相同)：

HasXAxis (布林型**)** - 啓動 X 軸。

XAxis (物件型**)** - 包含有關 X 軸之詳細資訊的物件 (支援 `com.sun.star.chart.ChartAxis` 服務)。

HasXAxisDescription (布林型**)** - 啓動 X 軸的描述。

HasXAxisGrid (布林型**)** - 啓動 X 軸的主網格。

XMainGrid (物件型**)** - 包含有關 X 軸主網格之詳細資訊的物件 (支援 `com.sun.star.chart.ChartGrid` 服務)。

HasXAxisHelpGrid (布林型**)** - 啓動 X 軸的輔助網格。

XHelpGrid (物件型**)** - 包含有關 X 軸輔助網格之詳細資訊的物件 (支援 `com.sun.star.chart.ChartGrid` 服務)。

HasXAxisTitle (布林型**)** - 啓動 X 軸的標題。

XAxisTitle (物件型**)** - 包含有關 X 軸標題之詳細資訊的物件 (支援 `com.sun.star.chart.ChartTitle` 服務)。

第二 X 軸 和 Y 軸

第二 X 軸和 Y 軸具有以下屬性 (以第二 X 軸的屬性爲例)：

HasSecondaryXAxis (布林型**)** - 啓動第二 X 軸。

SecondaryXAxis (物件型**)** - 包含有關第二 X 軸之詳細資訊的物件 (支援 `com.sun.star.chart.ChartAxis` 服務)。

HasSecondaryXAxisDescription (布林型**)** - 啓動 X 軸的描述。

軸的屬性

StarSuite 圖解的軸物件支援 `com.sun.star.chart.ChartAxis` 服務。除了字元屬性 (`com.sun.star.style.CharacterProperties` 服務，請參閱第 6 章) 和線條屬性 (`com.sun.star.drawing.LineStyle` 服務，請參閱第 8 章) 以外，還提供以下屬性：

Max (雙精度型**)** - 軸的最大值。

Min (雙精度型**)** - 軸的最小值。

Origin (雙精度型**)** - 相交軸的交點。

StepMain (雙精度型**)** - 軸的兩條主線之間的距離。

StepHelp (雙精度型**)** - 軸的兩條輔助線之間的距離。

AutoMax (布林型**)** - 自動確定軸的最大值。

AutoMin (布林型**)** - 自動確定軸的最小值。

AutoOrigin (布林型**)** - 自動確定相交軸的交點。

AutoStepMain (布林型**)** - 自動確定軸的主線之間的距離。

AutoStepHelp (布林型**)** - 自動確定軸的輔助線之間的距離。

Logarithmic (布林型**)** - 以對數方式 (而不是線性方式) 顯示軸的比例。

DisplayLabels (布林型**)** - 啓動軸的文字貼標。

TextRotation (長型**)** - 軸的文字貼標之旋轉角度，以百分之一度爲單位。

Marks (常數型**)** - 指定軸的主線位於圖解區域之內還是之外的常數 (標準值與 `com.sun.star.chart.ChartAxisMarks` 一致)

HelpMarks (常數型**)** - 指定軸的輔助線位於圖解區域之內還是之外的常數 (標準值與 `com.sun.star.chart.ChartAxisMarks` 一致)

Overlap (長型**)** - 百分比，指定不同資料集的條圖互相重疊的程度 (100% 表示完全重疊，-100% 表示各條之間有相當於一條寬度的間隔)。

GapWidth (長型**)** - 百分比，指定圖表中不同條群組之間的間隔 (100% 表示相當於一條寬度的間隔)。

ArrangeOrder (列舉型**)** - 標籤位置的細節，除了用於定位在一行上的選項以外，還具有用於在兩行上交替放置貼標的選項 (標準值由 `com.sun.star.chart.ChartAxisArrangeOrderType` 決定)

TextBreak (布林型**)** - 允許換行。

TextCanOverlap (布林型**)** - 允許文字重疊。

NumberFormat (長型**)** - 數字格式 (請參閱第 7 章中數字、日期和文字的格式一節)

軸網格的屬性

軸網格物件基於 `com.sun.star.chart.ChartGrid` 服務，因此該物件支援 `com.sun.star.drawing.LineStyle` 支援服務的線條屬性 (請參閱第 8 章)。

軸標題的屬性

格式化軸標題所用的物件基於 `com.sun.star.chart.ChartTitle` 服務，此服務還用於圖解標題。

示例

以下示例將建立線條圖解。將圖解的背景牆顏色設定爲白色。X 軸和 Y 軸都有灰色輔助網格，以便於識別位置。將 Y 軸的最小值固定爲 0，最大值固定爲 100，這樣，即使變更了數值，圖解的解析度仍可保持不變。

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")

Chart.Diagram.Wall.FillColor = RGB(255, 255, 255)

Chart.Diagram.HasXAxisGrid = True
Chart.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.HasYAxisGrid = True
Chart.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)
Chart.Diagram.YAxis.Min = 0
Chart.Diagram.YAxis.Max = 100
```

3D 圖解

StarSuite 中的大多數圖解都可以使用 3D 圖形來顯示。提供此選項的所有圖解類型都支援 `com.sun.star.chart.Dim3DDiagram` 服務。此服務僅提供一種屬性：

`Dim3D (`布林型**`)`** - 啓動 3D 顯示。

重疊圖解

重疊圖解是將數個個別數值上下堆疊起來排序，從而生成總值的一種圖解。這種檢視不僅可以顯示個別數值，還可以顯示所有數值的摘要。

在 StarSuite 中，各種類型的圖解都能以重疊的形式顯示。所有這些圖解都支援 `com.sun.star.chart.StackableDiagram` 服務，因此提供以下屬性：

`Stacked` (布林型) - 啓動重疊檢視模式。

`Percent` (布林型) - 不顯示絕對數值，而顯示各自所佔的百分比。

圖解類型

線條圖解

線條圖解 (服務 `com.sun.star.chart.LineDiagram`) 支援一個 X 軸、兩個 Y 軸和一個 Z 軸。它們可以顯示爲平面圖形或 3D 圖形 (`com.sun.star.chart.Dim3Ddiagram` 服務)。這些線條可以重疊 (`com.sun.star.chart.StackableDiagram`)。

線條圖解提供以下屬性：

`SymbolType` (常數型) - 顯示資料點所使用的符號 (常數與 `com.sun.star.chart.ChartSymbolType` 一致)。

`SymbolSize` (長型) - 顯示資料點所使用的符號大小，以百分之一公釐爲單位。

`SymbolBitmapURL` (字串型) - 顯示資料點所使用的圖形之檔案名稱。

`Lines` (布林型) - 透過線條連結資料點。

`SplineType` (長型) - 平滑線條的齒條函數 (0：無齒條函數，1：三次齒條，2：B 齒條)。

`SplineOrder` (長型) - 齒條的多項式權重 (僅適用於 B 齒條)。

`SplineResolution` (長型) - 齒條計算所使用的支援點數。

平面圖解

平面圖解 (`com.sun.star.chart.AreaDiagram` 服務) 支援一個 X 軸、兩個 Y 軸和一個 Z 軸。它們可以顯示爲平面圖形或 3D 圖形 (`com.sun.star.chart.Dim3Ddiagram` 服務)。這些平面可以重疊 (`com.sun.star.chart.StackableDiagram`)。

條形圖解

條形圖解 (服務 `com.sun.star.chart.BarDiagram`) 支援一個 X 軸、兩個 Y 軸和一個 Z 軸。它們可以顯示爲平面圖形或 3D 圖形 (`com.sun.star.chart.Dim3Ddiagram` 服務)。這些條形可以重疊 (`com.sun.star.chart.StackableDiagram`)。

條形圖解提供以下屬性：

Vertical (布林型**) -** 垂直顯示這些條形，否則水平繪製這些條形。

Deep (布林型**)** - 在 3D 檢視模式中，將這些條形前後堆疊顯示，而不是並排顯示。

StackedBarsConnected (布林型**)** - 透過線條連結重疊圖解中的相關聯條形 (僅適用於水平方向的圖表)。

NumberOfLines (長型**)** - 要在重疊圖解中顯示爲線條而不是條形的線條數。

圓形圖解

圓形圖解 (`com.sun.star.chart.PieDiagram` 服務) 不包含任何軸，也無法重疊。它們可以顯示爲平面圖形或 3D 圖形 (`com.sun.star.chart.Dim3Ddiagram` 服務)。

第 **10** 章

資料庫存取

StarSuite 具有整合的資料庫介面 (獨立於任何系統)，稱爲 Star 資料庫連接 (SDBC)。開發此介面旨在儘可能多地存取不同的資料源。

若要實現這一目的，可透過驅動程式來存取資料源。驅動程式從中取得資料的資料源與 SDBC 使用者無關。有些驅動程式存取基於檔案的資料庫，並直接從中取得資料。其他驅動程式則使用標準介面，例如 JDBC 或 ODBC。但是，還有一些特殊的驅動程式，它們存取作爲資料源的 MAPI 通訊錄、LDAP 目錄或 StarSuite 工作表。

由於這些驅動程式都基於 UNO 程式元件，因此，可以開發其他驅動程式，從而開放新的資料源。如需有關詳細資訊，請參閱《StarSuite 開發者指南》。

從概念上講，SDBC 相當於 VBA 中的 ADO 程式庫和 DAO 程式庫。它允許對資料庫的高層級存取，而不考量底層資料庫後端。

自 StarSuite 6 問世以來，StarSuite 的資料庫介面逐步發展。雖然，過去主要透過 `Application` 物件的一系列方法來存取資料庫，但 StarSuite 6 的介面也細分爲數個物件。`DatabaseContext` 用作資料庫函數的根物件。

SQL：查詢語言

SQL 語言作爲一種查詢語言供 SDBC 使用者使用。爲了比較不同 SQL 語言之間的差別，StarSuite 中的 SDBC 程式元件都具有自己的 SQL 語言剖析器。它使用查詢視窗來檢查輸入的 SQL 指令，並校正簡單的語法錯誤，例如那些與字元大小寫相關聯的錯誤等。

如果驅動程式允許存取不支援 SQL 的資料源，則它必須單獨將被傳輸的 SQL 指令轉換爲所需的當地存取。

從 SDBC 實現 SQL 是爲了適應 SQL-ANSI 標準。不支援特定於 Microsoft 的延伸 (例如 `INNER JOIN` 結構)。這些延伸需要用標準指令代替 (例如，`INNER JOIN` 應使用 `WHERE` 子句代替)。

資料庫的存取類型

StarSuite 的資料庫介面適用於 StarSuite Writer 應用程式、StarSuite Calc 應用程式以及資料庫表單。

在 StarSuite Writer 中，可以借助 SDBC 資料源建立標準字母，然後可以將這些字母列印出來。還可以選擇使用拖放功能將資料從資料庫視窗移至文字文件中。

如果使用者將資料庫表格移到工作表中，StarSuite 會建立一個表格區域。如果修改了原來的資料，只需按一下滑鼠即可更新此區域。反之，工作表資料也可以移到資料庫表格中，此時執行資料庫匯入。

最後，StarSuite 提供了一種基於資料庫建立表單的機制。若要如此，使用者首先應建立一個標準 StarSuite Writer 表單或 StarSuite Calc 表單，然後將各欄位連結至資料庫。

此處所指定的所有選項都基於 StarSuite 的使用者介面。使用相應的函數不需要程式設計知識。

但是，本章較少涉及有關指定函數的資訊，主要討論的是 SDBC 的程式設計介面，此介面允許資料庫自動查詢，因此，允許使用更爲廣泛的應用程式。

不過，要完全理解以下各節，需要瞭解有關資料庫函數以及 SQL 查詢語言的基本知識。

資料源

資料庫是透過建立通常所說的資料源加入到 StarSuite 中的。使用者介面在**[Extras]**功能表中提供了用於建立資料源的相應選項。但是，也可以使用 StarSuite Basic 建立和使用資料源。

使用 `createUnoService` 函數所建立的資料庫上下文物件用作存取資料源的起點。它基於 `com.sun.star.sdb.DatabaseContext` 服務，是所有資料庫作業的根物件。

以下示例顯示了如何建立資料庫上下文，然後用它確定所有可用資料源的名稱。這些名稱將顯示在訊息方塊中。

```
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
```

個別資料源基於 `com.sun.star.sdb.DataSource` 服務，可以使用 `getByName` 方法透過資料庫上下文來確定：

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
```

```
DataSource = DatabaseContext.getByName("Customers")
```

此示例爲名爲 *Customers* 的資料源建立了一個 DataSource 物件。

資料源提供了一系列屬性，而這些屬性反過來又提供了有關資料來源的一般資訊以及有關存取方法的資訊。這些屬性是：

Name (字串型**)** - 資料源的名稱。

URL (字串型**)** - 資料源的 URL，格式爲 *jdbc:subprotocol :subname* 或 *sdbc:subprotocol : subname* 。

Info (陣列型**)** - 包含帶有連結參數 (通常至少包含使用者名稱和密碼) 的 PropertyValue 對的陣列。

User (字串型**)** - 使用者名稱。

Password (字串型**)** - 使用者密碼 (不儲存)。

IsPasswordRequired (布林型**)** - 需要密碼，且由使用者互動式請求此密碼。

IsReadOnly (布林型**)** - 允許唯讀存取資料庫。

NumberFormatsSupplier (物件型**)** - 此物件包含適用於資料庫的數字格式 (支援 com.sun.star.util.XNumberFormatsSupplier 介面，請參閱第 7 章中數字、日期和文字的格式一節)。

TableFilter (陣列型**)** - 要顯示的表格名稱之清單。

TableTypeFilter (陣列型**)** - 要顯示的表格類型之清單。可用的值是 TABLE、VIEW 和 SYSTEM TABLE。

SuppressVersionColumns (布林型**)** - 不顯示用於版本管理的欄。

StarSuite 的資料源與 ODBC 中的資料源不完全一致。ODBC 資料源僅包含有關資料來源的資訊，而 StarSuite 中的資料源還包含有關如何在 StarSuite 的資料庫視窗內顯示資料的一系列資訊。

查詢

可以爲資料源指定預先定義的查詢。StarSuite 會記錄 SQL 查詢指令，以備隨時使用。使用查詢可以簡化資料庫的操作，因爲只需按一下滑鼠即可開啓查詢。而且，查詢還爲不瞭解 SQL 的使用者提供了發出 SQL 指令的選項。

支援 com.sun.star.sdb.QueryDefinition 服務的物件隱藏在查詢中。這些查詢是透過資料源的 QueryDefinitions 方法來存取的。

以下示例將在訊息方塊中列出可以建立的資料源查詢之名稱。

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer
```

```
DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
      QueryDefinition = QueryDefinitions(I)
      MsgBox QueryDefinition.Name
Next I
```

除了示例所使用的 Name 屬性以外，com.sun.star.sdb.QueryDefinition 還提供了一整套其他屬性。它們是：

Name (字串型**)** - 查詢名稱。

Command (字串型**)** - SQL 指令 (通常是 SELECT 指令)。

UpdateTableName (字串型**)** - 適用於基於數個表格的查詢：可以修改其中之值的表格之名稱。

UpdateCatalogName (字串型**)** - 最新表格目錄的名稱。

UpdateSchemaName (字串型**)** - 最新表格圖解的名稱。

以下示例顯示了如何透過程序設計控制方式建立查詢物件，並將其指定給資料源。

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELECT * FROM Customer"

QueryDefinitions.insertByName("NewQuery", QueryDefinition)
```

首先使用 `createUnoService` 呼叫建立查詢物件，然後初始化此物件，並透過 `insertByName` 將其插入 `QueryDefinitions` 物件中。

與資料庫表單連結

爲了簡化資料源的操作，StarSuite 提供了用於將資料源與資料庫表單相連結的選項。這些連結是透過 `getBookmarks()` 方法來實現的。這樣會傳回一個已命名的容器 (`com.sun.star.sdb.DefinitionContainer`)，其中包含了資料源的所有連結。內文標籤可以透過 `Name` 或 `Index` 來存取。

以下示例將確定內文標籤 *MyBookmark* 的 URL。

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Bookmarks As Object
Dim URL As String
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("MyBookmark")
MsgBox URL
```

資料庫存取

存取資料庫時，需要資料庫連線。這是一種允許與資料庫直接通訊的傳輸通道。與上一節中提到的資料源不同，每次重新啓動程式時，必須重新建立資料庫連線。

StarSuite 提供了各種建立資料庫連線的方法。下面說明基於現有資料源的方法。

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
      Connection = DataSource.GetConnection("","")
Else
      InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
      Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If
```

示例中所使用的程式碼首先檢查資料庫是否採用密碼保護。如果没有，則使用 `GetConnection` 呼叫，建立所需的資料庫連線。指令行中的兩個空字串代表使用者名稱和密碼。

如果資料庫採用了密碼保護，此示例會建立一個 `InteractionHandler`，然後使用 `ConnectWithCompletion` 方法開啓資料庫連線。InteractionHandler 可確保 StarSuite 要求使用者提供所需的登入資料。

表格的循環

在 StarSuite 中，通常使用 `ResultSet` 物件存取表格。`ResultSet` 是一種標記類型，指示使用 `SELECT` 指令所得到的結果卷次中的目前資料集。

此示例顯示了如何使用 ResultSet 在資料庫表格中查詢值。

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object
Dim Statement As Object
Dim ResultSet As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
      Connection = DataSource.GetConnection("","")
Else
      InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
      Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If

Statement = Connection.createStatement()
ResultSet = Statement.executeQuery("SELECT CustomerNumber FROM Customer")

If Not IsNull(ResultSet) Then
      While ResultSet.next
            MsgBox ResultSet.getString(1)
      Wend
End If
```

資料庫連線一旦建成，示例中所使用的程式碼就首先透過 Connection.createObject 呼叫來建立 Statement 物件。然後，此 Statement 物件使用 executeQuery 呼叫傳回實際的 ResultSet。此時，程式檢查 ResultSet 是否實際存在，並使用迴路來遍歷資料條目。所需值 (在此示例中，爲 CustomerNumber 欄位中的值) 使用 getString 方法傳回 ResultSet，藉此，參數 1 確定此呼叫與第一欄的值有關。

SDBC 中的 ResultSet 物件相當於 DAO 和 ADO 中的 Recordset 物件，因爲它也提供對資料庫的反覆存取。

在 StarSuite 6.x 中，資料庫實際上是透過 ResultSet 物件來存取的。它反映了表格的內容或 SQL-SELECT 指令的結果。以往 ResultSet 物件在 Application 物件中提供了常駐方法，用於瀏覽資料 (例如 DataNextRecord)。

擷取值的特定於類型的方法

從上一節的示例中可以看出，StarSuite 提供了用於存取表格內容的 getString 方法。此方法提供了字串形式的結果。下列 get 方法可用：

getByte() - 支援數字、字元和字串的 SQL 資料類型。

`getShort()` - 支援數字、字元和字串的 SQL 資料類型。

`getInt()` - 支援數字、字元和字串的 SQL 資料類型。

`getLong()` - 支援數字、字元和字串的 SQL 資料類型。

`getFloat()` - 支援數字、字元和字串的 SQL 資料類型。

`getDouble()` - 支援數字、字元和字串的 SQL 資料類型。

`getBoolean()` - 支援數字、字元和字串的 SQL 資料類型。

`getString()` - 支援所有的 SQL 資料類型。

`getBytes()` - 支援二進制值的 SQL 資料類型。

`getDate()` - 支援數字、字串、日期與時間標記的 SQL 資料類型。

`getTime()` - 支援數字、字串、日期與時間標記的 SQL 資料類型。

`getTimestamp()` - 支援數字、字串、日期與時間標記的 SQL 資料類型。

`getCharacterStream()` - 支援數字、字串和二進制值的 SQL 資料類型。

`getUnicodeStream()` - 支援數字、字串和二進制值的 SQL 資料類型。

`getBinaryStream()` - 二進制值。

`getObject()` - 支援所有的 SQL 資料類型。

無論哪種情況，欄數都可以作爲參數 (其值可以被查詢) 列出。

ResultSet 變體

存取資料庫的關鍵問題是速度。因此，StarSuite 提供了數種最佳化 `ResultSets` 的方法，從而控制存取速度。`ResultSet` 提供的函數越多，通常執行起來就越複雜，因而存取函數的速度就越慢。

如「表格的循環」一節中所述，簡單的 `ResultSet` 提供最小範圍可用的函數。它僅允許採用向前循環，並僅適用於被詢問的值。因而不包括更多其他瀏覽選項，例如修改值的可能性等。

用於建立 ResultSet 的 Statement 物件提供了一些允許影響 ResultSet 之函數的屬性：

ResultSetConcurrency (常數型**)** - 有關是否可以修改資料的規格 (這些規格與 com.sun.star.sdbc.ResultSetConcurrency 一致)。

ResultSetType (常數型**)** - 有關 ResultSets 類型的規格 (這些規格與 com.sun.star.sdbc.ResultSetType 一致)。

在 com.sun.star.sdbc.ResultSetConcurrency 中定義的值包括：

UPDATABLE - ResultSet 允許修改值。

READ_ONLY - ResultSet 不允許進行修改。

常數的 com.sun.star.sdbc.ResultSetConcurrency 群組提供了以下規格：

FORWARD_ONLY - ResultSet 僅允許向前瀏覽。

SCROLL_INSENSITIVE - ResultSet 允許所有類型的瀏覽，但是不記錄原來資料的變更。

SCROLL_SENSITIVE - ResultSet 允許所有類型的瀏覽，對原來資料的變更會影響 ResultSet。

包含 READ_ONLY 屬性和 SCROLL_INSENSITIVE 屬性的 ResultSet，與 ADO 和 DAO 中的 Snapshot 類型的資料條目集相對應。

使用 ResultSet 的 UPDATEABLE 屬性和 SCROLL_SENSITIVE 屬性時，ResultSet 的函數之範圍相當於 ADO 和 DAO 中 Dynaset 類型 Recordset。

在 ResultSet 中瀏覽的方法

如果 ResultSet 爲 SCROLL_INSENSITIVE 類型或 SCROLL_SENSITIVE 類型，則它支援在許多種資料中瀏覽的一整套方法。其中，主要方法有：

next() - 瀏覽至下一個資料條目。

previous() - 瀏覽至上一個資料條目。

first() - 瀏覽至第一個資料條目。

last() - 瀏覽至最後一個資料條目。

beforeFirst() - 瀏覽至第一個資料條目之前。

afterLast() - 瀏覽至最後一個資料條目之後。

所有方法均傳回一個布林參數，此參數指定瀏覽是否成功。

爲了確定游標目前所在位置，提供了以下測試方法，這些方法都會傳回一個布林值：

isBeforeFirst() - ResultSet 位於第一個資料條目之前。

isAfterLast() - ResultSet 位於最後一個資料條目之後。

isFirst() - ResultSet 是第一個資料條目。

isLast() - ResultSet 是最後一個資料條目。

修改資料條目

如果已經透過 ResultSetConcurrency = UPDATEABLE 的值建立了 ResultSet，則可編輯其內容。只要 SQL 指令允許資料可重新寫入資料庫 (依據原則)，這一點就適用。例如，對於具有連結的欄或累積值的複雜 SQL 指令，這一點就不適用。

ResultSet 物件提供了用於修改值的 Update 方法，這些方法的結構與用於擷取值的 get 方法相同。例如，updateString 方法允許寫入字串。

修改值之後，必須使用 updateRow() 方法將值傳輸到資料庫中。執行下一個瀏覽指令之前，必須先進行此呼叫，否則將遺失這些值。

如果在修改期間發生了錯誤，則可使用 cancelRowUpdates() 方法來復原。僅當使用 updateRow()，卻未將資料重新寫入資料庫時，此呼叫才可用。

第 11 章

對話方塊

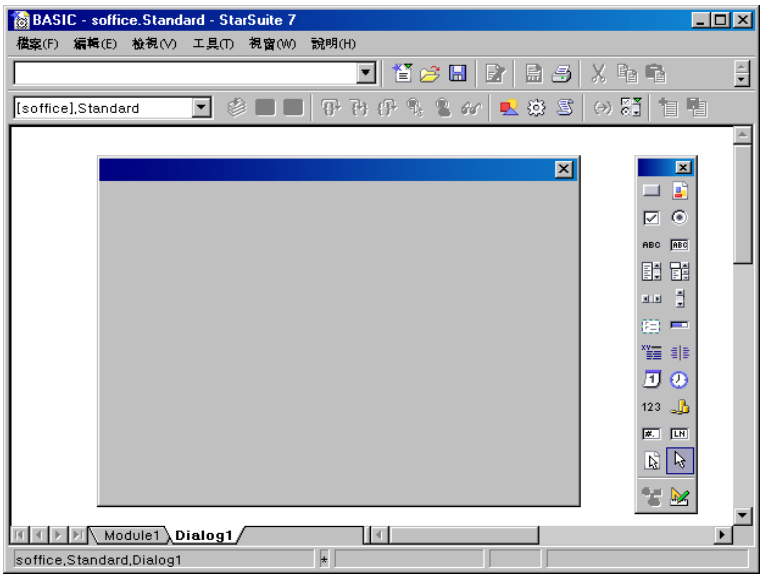
您可以將自訂對話方塊視窗和表單加入到 StarSuite 文件中。然後可以把這些視窗和表單連結到 StarSuite Basic 巨集，從而極大地擴充 StarSuite Basic 的使用範圍。例如，對話方塊可以顯示資料庫資訊，或以 自動檔案助理的形式指導使用者逐步執行建立新文件的程序。

使用對話方塊

StarSuite Basic 的對話方塊由一個可包含文字欄位、清單方塊、單選按鈕以及其他控制項元素的對話方塊視窗構成。

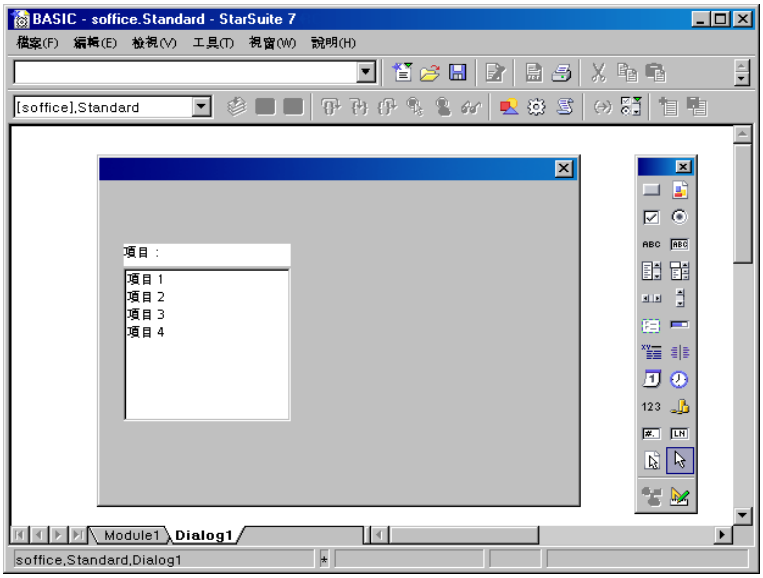
建立對話方塊

可以使用 StarSuite 對話方塊編輯器建立和構造對話方塊，使用 StarSuite 對話方塊編輯器的方法與使用 StarSuite Draw 的方法一樣：



實際上，也就是將所需控制項元素從設計面板 (右側) 拖曳到對話方塊區域中，您可在此區域中定義控制項元素的位置和大小。

下面的示例顯示了一個包含貼標和清單方塊的對話方塊。



可以使用以下程式碼開啓對話方塊：

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` 建立了一個名爲 Dlg 的物件，該物件參照了一個關聯的對話方塊。建立對話方塊之前，必須確保已載入對話方塊所用的程式庫 (在此示例中，爲 Standard 程式庫)。如果没有載入，則 `LoadLibrary` 方法將執行此工作。

一旦 `Dlg` 對話方塊物件進行了初始化，您即可使用 `Execute` 方法顯示此對話方塊。此類對話方塊均稱爲模態對話方塊，因爲僅在將其關閉時才允許其他程式動作。此類對話方塊開啓時，程式會保持在 `Execute` 呼叫狀態。

一旦程式結束，最後一行程式碼中的 `dispose` 方法會批準對話方塊使用的資源。

關閉對話方塊

使用[確定]或[取消]按鈕關閉

如果對話方塊中包含**[確定]**或**[取消]**按鈕，只需按下其中的一個按鈕，對話方塊就會自動關閉。本章的〈對話方塊控制項元素詳述〉一節將討論使用這些按鈕方面的更多資訊。

如果按一下**[確定]**按鈕關閉對話方塊，則 `Execute` 方法會傳回傳回值 `1`，否則會傳回值 0。

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
```

```
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
      MsgBox "Ok pressed"
Case 0
      MsgBox "Cancel pressed"
End Select
```

使用標題列中的[關閉]按鈕關閉

如果需要，可透過按一下對話方塊視窗標題列上的[關閉]按鈕來關閉對話方塊。在此情形下，對話方塊的 `Execute` 方法會傳回值 0，這與按[取消]按鈕時傳回的值相同。

使用明確的程式呼叫進行關閉

也可以使用 `endExecute` 方法關閉開啓的對話方塊視窗：

```
Dlg.endExecute()
```

存取個別控制項元素

對話方塊中可含有任意數目的控制項元素。可透過 `getControl` 方法 (能夠傳回控制項元素名稱) 存取這些元素。

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

此程式碼首先確定了用作 `MyButton` 控制項元素的物件，然後使用對元素的參照初始化 `Ctl` 物件變數。最後，程式碼將控制項元素的 `Label` 屬性值設定爲 `New Label` 值。

請注意，StarSuite Basic 區分控制項元素名稱字元的大小寫。

使用對話方塊和控制項元素的 *Model (*模型*)*

StarSuite API 中的許多地方都區分可視程式元素 (檢視) 與它們後面的資料或元素 (*Model (*模型*)*)。除了具有控制項元素的方法和屬性之外，對話方塊和控制項元素物件都具有一個下級 `Model (`模型`)` 物件。此物件用於直接存取對話方塊或控制項元素的內容。

在對話方塊中，資料和描述之間的區別並不總是像在其他 StarSuite API 區域中那樣清楚。此 API 元素既可以爲 View (檢視)，也可以爲 Model (模型)。

`Model` 屬性提供了一種由程式控制的存取方式，用於存取對話方塊和控制項元素物件的模型。

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

此示例借助於來自 `cmdNtext` 的模型物件關閉了 `Dlg` 對話方塊中的 `cmdNtext` 按鈕。

屬性

名稱和標題

每個控制項元素都有自己的名稱，這些名稱可以使用以下模型屬性查詢：

`Model.Name (`字串型**`)`** - 控制項元素名稱

您可以透過以下模型屬性指定要在對話方塊標題列中出現的標題：

`Model.Title (`字串型**`)`** - 對話方塊標題 (僅適用於對話方塊)。

位置和大小

您可以使用模型物件的以下屬性查詢控制項元素的大小和位置：

`Model.Height (`長型**`)`** - 控制項元素的高度 (以 ma 爲單位)

`Model.Width (`長型**`)`** - 控制項元素的寬度 (以 ma 爲單位)

`Model.PositionX (`長型**`)`** - 控制項元素的 X 位置，從對話方塊左側的內邊緣算起 (以 ma 爲單位)

`Model.PositionY (`長型**`)`** - 控制項元素的 Y 位置，從對話方塊上方的內邊緣算起 (以 ma 爲單位)

爲了確保對話方塊外觀的平台獨立性，StarSuite 使用 *Map AppFont (ma)* 內部單位來指定對話方塊內部的位置和大小。ma 單位的高度定義爲作業系統中定義的系統字型字元平均高度的八分之一，寬度爲其四分之一 。透過使用 ma 單位，StarSuite 可確保其對話方塊在使用不同系統設定的不同系統上外觀都一致。

如果要變更執行期間的控制項元素的大小或位置，請先確定此對話方塊的總大小，然後再將控制項元素的值調整爲相應的比率。

Map AppFont (ma) 代替 Twip 單位，以獲得更佳的平台獨立性。

焦點和定位鍵序列

您可以透過按 TAB 鍵在任意對話方塊中瀏覽的控制項元素。在此情形下，可以在控制項元素模型中使用以下屬性：

Model.Enabled (布林型**)** - 啓動控制項元素

Model.Tabstop (布林型**)** - 允許透過 Tab 鍵存取控制項元素

Model.TabIndex (長型**)** - 控制項元素在啓動順序中的位置

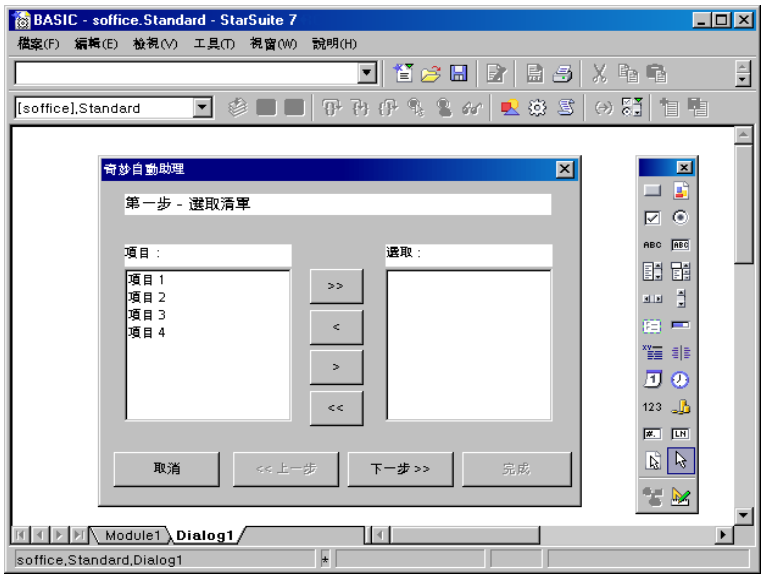
最後，控制項元素會提供 `getFocus` 方法以確保基礎控制項元素獲得焦點：

getFocus - 控制項元素獲得焦點 (僅適用於對話方塊)

多頁對話方塊

StarSuite 中的對話方塊可以有多個標籤頁。對話方塊的 `Step` 屬性用於定義對話方塊的目前標籤頁，而控制項元素的 `Step` 屬性則用於指定要顯示此控制項元素的標籤頁。

作爲 `Step` 值的 `0` 是一個特殊情形。如果在對話方塊中將此值設定爲零，則會顯示所有控制項元素，而不管它們具有什麼 `Step` 屬性值。同樣地，如果將 控制項元素的這一值設定爲零，則該元素將會在對話方塊的所有標籤頁上顯示。



在以上示例中，也可以將 `Step` 值 `0` 指定給分隔線條以及[取消]、[繼續]、[返回]和[完成]按鈕，以便在所有頁面上顯示這些元素。還可以將這些元素指定給個別標籤頁 (例如，第 `1` 頁)。

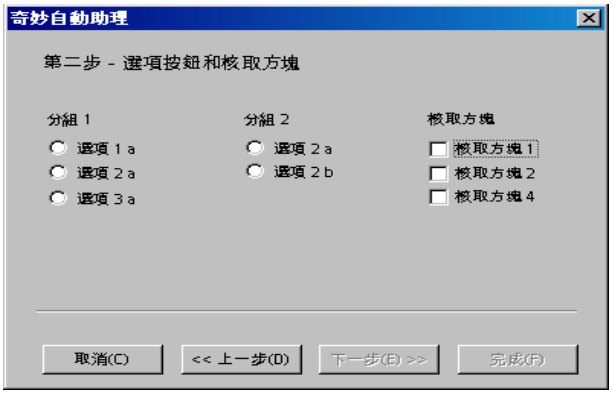
以下程式碼顯示了如何增大或減小[繼續]和[返回]按鈕的事件處理程式中的 `Step` 值，以及如何變更這些按鈕的狀態。

```
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
```

必須含有一個參照了開啓的對話方塊的全域 Dlg 變數，此示例才可能 實現。然後，對話方塊的外觀會做出以下變更：

第 **1** 頁：

第 **2** 頁：



事件

StarSuite 對話方塊和表單都基於事件導向的程式設計模型，在這個模型中，可以把事件處理程式指定給控制項元素。當發生特定的動作時，即使此動作爲另一個事件，事件處理程式都會執行一個預先定義的程序 。使用事件處理功能也可以編輯文件或開啓資料庫，以及存取其他控制項元素。

StarSuite 控制項元素能夠識別在不同情形下觸發的不同事件類型。這些事件類型可分爲四個群組：

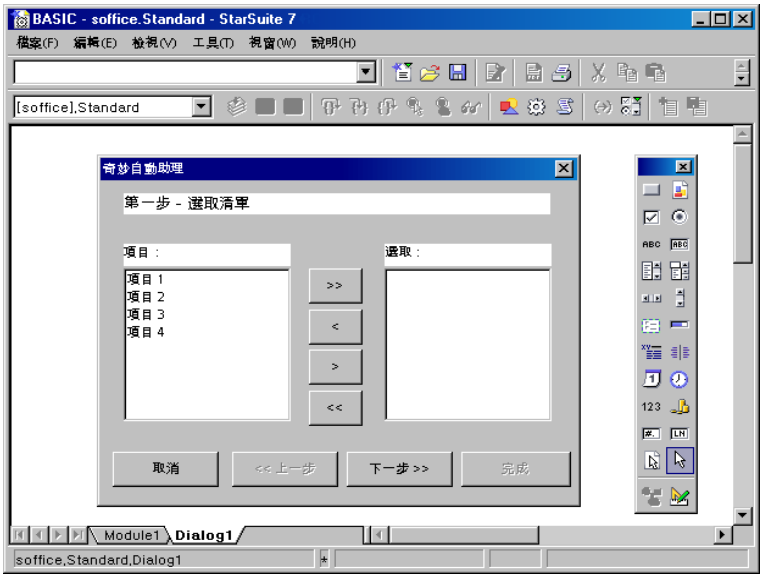
滑鼠控制：與滑鼠動作相對應的事件 (例如，簡單滑鼠移動或按一下特定的螢幕位置)

鍵盤控制：由鍵盤擊鍵觸發的事件

焦點修改：啓動或關閉控制項元素時 StarSuite 執行的事件

控制項元素特有的事件：僅在特定的控制項元素上才會發生的事件

使用事件時，請確保已在 StarSuite 開發環境中建立了與之關聯的對話方塊，並確保此對話方塊包含所需控制項元素或文件 (如果要將此事件採用到表單中)。



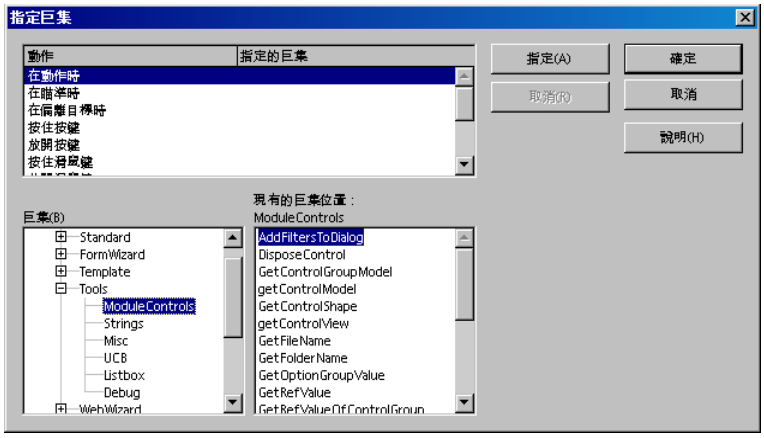
上圖以包含兩個清單方塊的對話方塊視窗顯示了 StarSuite Basic 開發環境。您可以使用兩個清單方塊之間的按鈕將資料從一個清單移至另一個清單中。

如果要在螢幕上顯示版式，則應建立與之關聯的 StarSuite Basic 程式，以便事件處理程式呼叫這些程式。盡管在任何模組中都可以使用這些程序，但最好限於僅供兩個模組使用。爲使程式碼更易於閱讀，應爲這些程序指定有意義的名稱。直接從巨集跳換到一般程式程序會導致程式碼不明確。因此，爲了簡化程式碼維護和疑難排解，就應建立另一個程序，作爲事件處理的輸入點，即使它只執行對目標程序的單一呼叫。

以下示例中的程式碼會將一個條目從對話方塊的左清單方塊移到右清單方塊中。

```
Sub cmdSelect_Initiated
      Dim objList As Object
      lstEntries = Dlg.getControl("lstEntries")
      lstSelection = Dlg.getControl("lstSelection")

      If lstEntries.SelectedItem > 0 Then
            lstSelection.AddItem(lstEntries.SelectedItem, 0)
            lstEntries.removeItems(lstEntries.SelectedItemPos, 1)
      Else
            Beep
      End If
End Sub
```

如果此程式是在 StarSuite Basic 中建立的，則可以使用對話方塊編輯器的屬性視窗將其指定給所需的事件。



指定對話方塊清單中列出了所有的 StarSuite Basic 程序。若要將一個程序指定給某個事件，請選取該程序，然後按一下**[**指定**]**。

參數

發生某個特定事件時並不總能獲得相應的回應。可能還需要其他資訊。例如，要按一下滑鼠，可能會需要按滑鼠按鈕時其所在的螢幕位置。

在 StarSuite Basic 中，可以使用物件參數將一個事件的更多相關資訊提供給一個程序，例如：

```
Sub ProcessEvent(Event As Object)

End Sub
```

構造 `Event` 物件時所需的準確性及其屬性取決於程序呼叫觸發的事件類型。以下幾節將詳細介紹這些事件類型。

不管在何種類型的事件中，所有物件都提供對相關控制項元素及其模型的存取。要存取控制項元素，可以使用

```
Event.Source
```

要存取模型，可以使用

```
Event.Source.Model
```

您可以使用這些屬性觸發事件處理程式中的事件。

滑鼠事件

StarSuite Basic 能夠識別以下滑鼠事件：

滑鼠移動 - 使用者移動滑鼠

按住鍵時移動滑鼠 - 使用者在按住鍵的同時拖曳滑鼠

按下滑鼠按鈕 - 使用者按下滑鼠按鈕

釋放滑鼠按鈕 - 使用者釋放滑鼠按鈕

滑鼠結束 - 使用者將滑鼠移出目前視窗

這些關聯事件物件的結構在 `com.sun.star.awt.MouseEvent` 結構中定義，此結構提供了以下資訊：

Buttons (短型**)** - 按下的按鈕 (一個或多個與 `com.sun.star.awt.MouseButton` 一致的常數)。

X (長型**)** - 從控制項元素左上角算起的滑鼠的 X 座標，以像素爲單位

Y (長型**)** - 從控制項元素左上角算起的滑鼠的 Y 座標，以像素爲單位

ClickCount (長型**)** - 與滑鼠事件關聯的點擊次數 (如果 StarSuite 的回應速度足夠快，由於只引發了一個事件，按兩下的 ClickCount 值也爲 1)。

`com.sun.star.awt.MouseButton` 中定義的滑鼠按鈕常數爲：

LEFT - 滑鼠左按鈕

RIGHT - 滑鼠右按鈕

MIDDLE - 滑鼠的中間按鈕

以下示例將輸出滑鼠位置及按下的滑鼠按鈕：

```
Sub MouseUp(Event As Object)
      Dim Msg As String
      Msg = "Keys: "
      If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
            Msg = Msg & "LEFT "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
            Msg = Msg & "RIGHT "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
            Msg = Msg & "MIDDLE "
      End If
      Msg = Msg & Chr(13) & "Position: "
      Msg = Msg & Event.X & "/" & Event.Y
      MsgBox Msg
End Sub
```

StarSuite Basic 中不提供 VBA 的 `Click` 和 `Doubleclick` 事件。而是用 StarSuite Basic 的 `MouseUp` 事件替代 `click` 動作，並透過變更應用程式邏輯模擬 `Doubleclick` 事件。

鍵盤事件

StarSuite Basic 提供以下鍵盤事件：

按下鍵 - 使用者按下一個鍵

釋放鍵 - 使用者釋放一個鍵

這兩個事件都與邏輯鍵動作相關，而與實體動作無關。如果使用者要按多個鍵以輸出一個字元 (例如，爲字元加入重音符號)，StarSuite Basic 僅會建立一個事件。

使用修飾鍵 (如 SHIFT 鍵或 ALT 鍵) 執行的單鍵動作不會建立一個獨立事件。

StarSuite Basic 會將事件物件提供的有關按下的鍵的資訊提供給程序，以便進行事件處理。它包含以下屬性：

`KeyCode` **(**短型**)** - 按下的鍵的程式碼 (與 `com.sun.star.awt.Key` 一致的標準值)

`KeyChar` **(**字串型**)** - 輸入的字元 (包括修飾鍵)

以下示例使用 KeyCode 屬性確定是否按下了 ENTER 鍵、TAB 鍵或其他控制鍵之一。如果按了上述鍵之一，則將傳回鍵的名稱，否則將傳回輸入的字元：

```
Sub KeyPressed(Event As Object)
      Dim Msg As String
      Select Case Event.KeyCode
      Case com.sun.star.awt.Key.RETURN
            Msg = "Return pressed"
      Case com.sun.star.awt.Key.TAB
            Msg = "Tab pressed"
      Case com.sun.star.awt.Key.DELETE
            Msg = "Delete pressed"
      Case com.sun.star.awt.Key.ESCAPE
            Msg = "Escape pressed"
      Case com.sun.star.awt.Key.DOWN
            Msg = "Down pressed"
      Case com.sun.star.awt.Key.UP
            Msg = "Up pressed"
      Case com.sun.star.awt.Key.LEFT
            Msg = "Left pressed"
      Case com.sun.star.awt.Key.RIGHT
            Msg = "Right pressed"
      Case Else
            Msg = "Character " & Event.KeyChar & " entered"
      End Select
      MsgBox Msg
End Sub
```

有關其他鍵盤常數的資訊，可在「API 參照」中 com.sun.star.awt.Key 常數群組下找到。

焦點事件

焦點事件表明控制項元素是獲得焦點還是失去焦點。例如，可以使用這些事件確定使用者是否已處理完某個控制項元素，以便更新對話方塊中的其他元素。可以使用以下焦點事件：

在獲得焦點時 - 元素獲得焦點

在失去焦點時 - 元素失去焦點

焦點事件的 Event 物件的構造如下所示：

FocusFlags (短型**) -** 焦點變更的原因 (與 com.sun.star.awt.FocusChangeReason 一致的標準值)。

NextFocus (物件型**) -** 獲得焦點的物件 (僅適用於 在失去焦點時事件)

Temporary (布林型**) -** 暫時遺失焦點

控制項元素特有的事件

除了上述所有控制項元素都支援的事件之外，還有一些控制項元素特有的事件，這些事件僅爲某些控制項元素而定義。其中最重要的事件爲：

項目變更時 - 控制項元素的值發生變更

項目狀態已變更 - 控制項元素的狀態發生變更

文字已修改 - 控制項元素的文字發生變更

在初始化時 - 觸發控制項元素時可以執行的動作 (例如，按一個按鈕)

使用這些事件時，請注意，每次在一些控制項元素上 (例如，在單選按鈕上) 按一下滑鼠時，都會執行某些事件，例如，在初始化時事件。此時並不會執行任何動作以檢查控制項元素的狀態是否真的發生了變更。若要 避免這樣的「盲目事件」，請將舊的控制項元素值保存在一個全域變數中，然後檢查執行事件時此值是否發生變更。

項目狀態已變更事件的屬性如下：

Selected (長型**)** - 目前選取的條目

Highlighted (長型**)** - 目前反白顯示的條目

ItemId (長型**)** - 條目的 ID

對話方塊控制項元素詳述

StarSuite Basic 能夠識別一系列控制項元素，這些元素分爲以下幾個群組：

條目欄位：

文字欄位

日期欄位

時間欄位

數值欄位

貨幣欄位

採用任何格式的欄位

按鈕：

標準按鈕

核取方塊

單選按鈕

選擇清單：

清單方塊

組合方塊

其他控制項元素：

捲軸 (水平和垂直方向)

群組欄位

進度列

分隔線條 (水平和垂直方向)

圖形

檔案選擇欄位

下面將介紹這些控制項元素中最重要的元素。

按鈕

按一下按鈕時，它會執行一項動作。

最簡單的分析藍本就是使按鈕在使用者按一下時觸發在初始化時事件。也可以使用 `PushButtonType` 屬性將另一個動作與按鈕連結在一起，以便開啓對話方塊。如果您所按下按鈕的該屬性值已設定爲 0，則對話方塊不受影響。如果您所按下按鈕的該屬性值已設定爲 1，則對話方塊將關閉，而且對話方塊的 `Execute` 方法會傳回值 1 (對話方塊序列已正確結束)。如果 `PushButtonType` 的值爲 2，則關閉對話方塊，而且對話方塊的 `Execute` 方法會傳回值 0 (對話方塊已關閉)。

下面是可透過按鈕模型使用的所有屬性：

`Model.BackgroundColor (`長型**`)`** - 背景顏色

`Model.DefaultButton (`布林型**`)`** - 按鈕用作標準值，並當此按鈕沒有焦點時回應 ENTER 鍵。

`Model.FontDescriptor (`結構型**`)`** - 用於指定所要使用字型細節的結構 (與 `com.sun.star.awt.FontDescriptor` 結構一致)

`Model.Label (`字串型**`)`** - 按鈕上顯示的貼標

`Model.Printable (`布林型**`)`** - 可以列印控制項元素

`Model.TextColor (`長型**`)`** - 控制項元素的文字顏色

`Model.HelpText (`字串型**`)`** - 將滑鼠游標移至控制項元素上方時所顯示的說明文字

`Model.HelpURL (`字串型**`)`** - 與控制項元素對應的線上說明的 URL

`PushButtonType (`短型**`)`** - 與按鈕連結的動作 (0：無動作；1：[確定]；2：[取消]

選項按鈕

這些按鈕通常以群組的形式使用，並且允許您從數個選項中選取一個。當您選取了一個選項時，群組中的其他所有選項都會關閉。這樣可確保一次僅設定一個選項按鈕。

每個選項按鈕控制項元素都提供兩種屬性：

State (布林型**)** - 啓動按鈕

Label (字串型**)** - 按鈕上顯示的貼標

也可以使用選項按鈕模型中的以下屬性：

Model.FontDescriptor (結構型**)** - 含有所使用字型的細節的結構 (與 `com.sun.star.awt.FontDescriptor` 一致)

Model.Label (字串型**)** - 控制項元素上顯示的貼標

Model.Printable (布林型**)** - 可以列印控制項元素

Model.State (短型**)** - 如果此屬性爲 1，則啓動該選項，否則關閉該選項

Model.TextColor (長型**)** - 控制項元素的文字顏色

Model.HelpText (字串型**)** - 滑鼠游標置於控制項元素上方時所顯示的說明文字

Model.HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL

若要將多個選項按鈕組合到一個群組中，必須無間隔地依啓動順序逐個定位這些選項按鈕 (`Model.TabIndex` 屬性，在對話方塊編輯器中被描述爲[Order])。如果其他控制項元素中斷了啓動順序，則 StarSuite 將自動啓動一個新控制項元素群組 (此時不考量第一個控制項元素群組)。

與 VBA 不同的是，不能在 StarSuite Basic 中的控制項元素群組內插入選項按鈕。透過在控制項元素的周圍繪製框架，StarSuite Basic 的控制項元素群組僅用於確保可視分割區。

核取方塊

核取方塊用於輸入值[是]或[否]，依所處的模式，它們可採用兩種或三種狀態。除了[是]和[否]狀態之外，如果相應的[是]或[否]狀態具有多重含義或含義不清，核取方塊可以具有一種中間狀態。

核取方塊提供以下屬性：

State (短型**)** - 核取方塊的狀態 (0：否，1：是，2：中間狀態)

Label (字串型**)** - 控制項元素的貼標

enableTriState (布林型**)** - 除了啓動或關閉狀態之外，還可以使用中間狀態

核取方塊的模型物件可提供以下屬性：

Model.FontDescriptor (結構型**)** - 含有所用字型的細節的結構 (與 `com.sun.star.awt.FontDescriptor` 結構一致)

Model.Label (字串型**)** - 控制項元素的貼標

Model.Printable (布林型**)** - 可以列印控制項元素

Model.State (短型**)** - 核取方塊的狀態 (0：否，1：是，2：中間狀態)

Model.Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素

Model.TextColor (長型**)** - 控制項元素的文字顏色

Model.HelpText (字串型**)** - 滑鼠游標置於控制項元素上方時所顯示的說明文字

Model.HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL

文字欄位

文字欄位允許使用者輸入數字和文字。`com.sun.star.awt.UnoControlEdit`. 服務構成了文字欄位的基礎。

文字欄位可包含一行或多行文字，並可對其進行編輯用於使用者輸入，或禁止使用者的輸入。文字欄位也可以用作特殊貨幣欄位和數值欄位，還可用作特殊工作的螢幕欄位。由於這些控制項元素都基於 `UnoControlEdit` Uno 服務，他們的程式控制處理很相似。

文字欄位提供以下屬性：

Text (字串型**)** - 目前文字

SelectedText (字串型**)** - 目前反白顯示的文字

Selection (結構型**)** - 以唯讀的方式反白顯示細節 (與 `com.sun.star.awt.Selection` 一致的結構，透過 `Min` 和 `Max` 屬性指定目前反白顯示的文字的開頭和結尾)

MaxTextLen (短型**)** - 可在欄位中輸入的最大字元數

Editable (布林型**)** - 值爲 `True` 時啓動輸入文字的選項,值爲 `False` 時禁止輸入選項 (不能直接呼叫此屬性，只能透過 `IsEditable` 進行呼叫)

IsEditable (布林值**)** - 可以變更控制項元素的內容，也可使內容唯讀。

另外，透過與之關聯的模型物件還提供了以下屬性：

Model.Align (短型**)** - 文字方向 (0：左對齊，1：置中，2：右對齊)

Model.BackgroundColor (長型**)** - 控制項元素的背景顏色

Model.Border (短型**)** - 邊框類型 (0：無邊框，1：3D 邊框，2：簡單邊框)

Model.EchoChar (字串型**)** - 密碼欄位的回應字元。

Model.FontDescriptor (結構型**)** - 含有所用字型的細節的結構 (與 `com.sun.star.awt.FontDescriptor` 結構一致)

Model.HardLineBreaks (布林型**)** - 在控制項元素文字中永久插入自動換行

Model.HScroll (布林型**)** -

Model.MaxTextLen (短型**)** - 最大文字長度，其中 0 對應於無長度限制

Model.MultiLine (布林型**)** - 允許條目跨越多行

Model.Printable (布林型**)** - 可以列印控制項元素

Model.ReadOnly (布林型**)** - 控制項元素的內容爲唯讀

Model.Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素

Model.Text (字串型**)** - 文字與控制項元素相關聯

Model.TextColor (長型**)** - 控制項元素的文字顏色

Model.VScroll (布林型**)** - 文字有一個垂直捲軸

Model.HelpText (字串型**)** - 滑鼠游標置於控制項元素上方時所顯示的說明文字

Model.HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL

清單方塊

清單方塊 (`com.sun.star.awt.UnoControlListBox` 服務) 支援以下屬性：

ItemCount (短型**)** - 元素的數目，唯讀

SelectedItem (字串型**)** - 反白顯示條目的文字，唯讀

SelectedItems (字串陣列**)** - 含有反白顯示條目的資料欄位，唯讀

SelectedItemPos (短型**)** - 目前所反白顯示的條目的數目，唯讀

SelectedItemsPos (短型陣列**)** - 含有反白顯示條目數目的資料欄位 (適用於支援多重選擇的清單)，唯讀

MultipleMode (布林型**)** - 值爲 `True` 時會啓動選擇多重條目的選項；值爲 `False` 時會禁止多重選擇 (不能直接呼叫此屬性，只能透過 `IsMultipleMode` 進行存取)

IsMultipleMode (布林型**)** - 允許在清單內進行多重選擇，唯讀

清單方塊提供以下方法：

addItem (Item, Pos) - 將在 Item 中指定的字串在 Pos 位置輸入清單

addItems (ItemArray, Pos) - 將字串的 ItemArray 資料欄位中列出的條目在 Pos 位置輸入清單

removeItems (Pos, Count) - 自 Pos 位置起移除 Count 條目

selectItem (Item, SelectMode) - 依 SelectMode 布林型變數啓動或關閉在字串 Item 中所指定元素的反白顯示

makeVisible (Pos) - 捲動過清單欄位，以便顯示透過 Pos 指定的條目

清單方塊的模型物件提供以下屬性：

Model.BackgroundColor (長型**)** - 控制項元素的背景顏色

Model.Border (短型**)** - 邊框類型 (0：無邊框，1：3D 邊框，2：簡單邊框)

Model.FontDescriptor (結構型**)** - 含有所用字型的細節的結構 (與 com.sun.star.awt.FontDescriptor 結構一致)

Model.LineCount (短型**)** - 控制項元素中的行數

Model.MultiSelection (布林型**)** - 允許多重選擇條目

Model.SelectedItems (字串陣列**)** - 反白顯示的條目之清單

Model.StringItemList (字串陣列**)** - 所有條目的清單

Model.Printable (布林型**)** - 列印控制項元素

Model.ReadOnly (布林型**)** - 控制項元素的內容爲唯讀

Model.Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

Model.TextColor (長型**)** - 控制項元素的文字顏色

Model.HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字

Model.HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL

StarSuite Basic 中没有 VBA 中用於配給清單條目數字其他值的選項 (ItemData)。如果您要管理數值型值 (例如資料庫 ID) 以及自然語言文字，必須建立與清單方塊進行並行管理的輔助資料欄位

第 **12** 章

表單

StarSuite 表單的結構在許多方面都與前一章所討論的對話方塊相同。但二者之間也存在數個關鍵區別：

對話方塊以單一對話方塊視窗的形式顯示在文件的上方，並且在對話方塊結束之前，不允許執行除對話方塊處理之外的任何動作。而表單則像繪圖元素一樣直接顯示在文件中。

系統提供了一個用於建立對話方塊的對話方塊編輯器，可在 StarSuite Basic 開發環境中找到此編輯器。而表單則可使用**[**表單功能**]**工具列直接在文件內建立。

所有的 StarSuite 文件都提供對話方塊功能，而僅在文字和工作表中才提供完整的表單功能。

表單中的控制項元素可以連結到外部資料庫表格。而此功能在對話方塊中則不可用。

對話方塊和表單中的控制項元素存在多方面的差異。

如果使用者要在表單中使用自己的方法進行事件處理，則應參閱第 11 章 (對話方塊)。這一章中所介紹的機制與表單所使用的機制完全一致。

使用表單

StarSuite 表單中可能包含直接插入文字或工作表中的文字欄位、清單方塊、單選按鈕以及一系列其他控制項元素。**[**表單功能**]**工具列用於編輯表單。

StarSuite 表單可能會採用以下兩種模式中的一種：草稿模式和顯示模式。在草稿模式下，可以變更控制項元素的位置，還可以使用屬性視窗編輯他們的屬性。

[表單功能**]**工具列也用於在模式之間進行切換。

確定物件表單

StarSuite 將表單的控制項元素置於繪圖物件層級。實際的物件表單可以透過繪圖層級的 Forms 清單進行存取。在文字文件中，可按以下方式存取物件：

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

GetByIndex 方法傳回了索引編號爲 0 的表單。

使用工作表時，需要透過 Sheets 清單獲得一個中間級，因爲繪圖層級不是直接位於文件內，而是分佈在個別工作表中：

```
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)
```

正如 GetByIndex 方法的名稱所暗示的，一個文件可能會含有多個表單。這很有用，例如，如果要在一個文件內顯示不同資料庫的內容，或者要在一個表單內顯示一種 1:n 的資料庫關係。建立子表單的選項也是爲此目的提供的。

控制項元素表單的三個方面

表單的控制項元素具有三個方面：

第一個是控制項元素的模型。對於 StarSuite Basic 程式設計師來說，它是使用控制項元素表單時的關鍵物件。

與此相對應的是控制項元素的檢視，它管理著顯示資訊。

由於在文件內，對控制項元素表單的管理就像管理特殊的繪圖元素那樣，因此也有一個形狀物件，可用於反映控制項元素的繪圖元素特有屬性 (尤其是其位置和大小)。

存取控制項元素表單的模型

可透過物件型表單的 GetByName 方法獲得表單控制項元素的模型：

```
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")
```

此示例確定了 MyListBox 控制項元素的模型，該元素位於目前開啓的文字文件的第一個表單中。

如果不確定某個控制項元素所屬的表單，則可以使用搜尋選項對所有表單進行搜尋，以找出所需的控制項元素：

```
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
```

此示例使用 HasByName 方法檢查文字文件中的所有表單，以確定這些表單是否包含一個名爲 MyListBox 的控制項元素模型。如果找到了相應的模型，則將此模型的參照儲存在 Ctl 變數中，並且終止搜尋。

存取控制項元素表單的檢視

要存取控制項元素表單的檢視，首先需要的是關聯的模型。然後，利用模型，使用文件控制器確定控制項元素的檢視

```
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
```

此示例中列出的程式碼與上一示例 (用於確定控制項元素模型) 中列出的程式碼非常相似。它不僅使用了 `Doc` 文件物件，而且還使用了參照目前文件視窗的 `DocCrl` 文件控制器物件。借助此控制器物件和控制項元素的模型，該程式碼接著使用 `GetControl` 方法確定控制項元素表單的檢視 (`CtlView` 變數)。

存取控制項元素表單的形狀物件

用於存取控制項元素形狀物件的方法也使用文件的相應繪圖層級。爲確定一個特殊的控制項元素，就必須搜尋該繪圖層級的所有繪圖元素。

```
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
```

此示例檢查了所有繪圖元素，以確定他們是否支援控制項元素表單所需要的 `com.sun.star.drawing.XControlShape` 介面。如果支援，則 `Control.Name` 屬性會接著檢查此控制項元素的名稱是否爲 `MyListBox`。如果是，函數將結束搜尋。

確定控制項元素的大小和位置

如前所述，可以使用關聯的形狀物件確定控制項元素的大小和位置。像所有其他形狀物件那樣，控制項元素的形狀物件也爲此提供了 `Size` 和 `Position` 屬性：

`Size (`結構型**`)`** - 控制項元素的大小 (`com.sun.star.awt.Size` 資料結構)。

`Position (`結構型**`)`** - 控制項元素的大小 (`com.sun.star.awt.Point` 資料結構)。

以下示例顯示如何使用關聯的形狀物件設定控制項元素的位置和大小：

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

要使程式碼能夠執行，控制項元素的形狀物件必須爲已知物件。如果不是已知物件，則必須使用以上程式碼確定此物件。

控制項元素表單詳述

在表單中可用的控制項元素與對話方塊中的控制項元素類似。可以選擇簡單文字欄位、清單方塊和組合方塊以及各種按鈕。

在下文中，將介紹控制項元素表單的多個最重要屬性。所有屬性均爲關聯模型物件的組成部份。

除了標準控制項元素，還可以在表單中使用表格控制項元素，從而允許加入完整的資料庫表格。第 11 章的〈資料庫表單〉一節中對此內容做了描述。

按鈕

表單按鈕的模型 物件提供以下屬性：

BackgroundColor (長型**)** - 背景顏色。

DefaultButton (布林型**)** - 作爲標準值的按鈕。在此情形下，按鈕即使未獲取焦點，也會回應輸入按鈕。

Enabled (布林型**)** - 可以啓動控制項元素。

Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

TabIndex (長型**)** - 控制項元素在啓動序列中的位置。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

Tag (字串型**)** - 含有其他資訊的字串，可儲存在按鈕中，用於由程式控制的存取。

TargetURL (字串型**)** - URL 類型的按鈕所對應的目標 URL。

TargetFrame (字串型**)** - 視窗 (或訊框) 的名稱，在視窗 (或訊框) 中，當按鈕 (URL 類型的按鈕) 啓動時，將開啓 TargetURL。

Label (字串型**)** - 按鈕貼標。

TextColor (長型**)** - 控制項元素的文字顏色。

HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字。

HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL。

ButtonType (列舉型**)** - 與按鈕連結的動作 (標準值來自 com.sun.star.form.FormButtonType)。

透過使用 ButtonType 屬性，可以自行定義按下按鈕時要自動執行的動作。與之關聯的 com.sun.star.form.FormButtonType 常數群組提供以下值：

PUSH - 標準按鈕。

SUBMIT - 結束表單輸入 (特別適用於 HTML 表單)。

RESET - 將表單內的所有值重設爲其原來的數值。

URL - 在 TargetURL (在透過 TargetFrame 指定的視窗中開啓) 中定義的 URL 的呼叫。

表單中不支援對話方塊中提供的**[**確定**]**按鈕和**[**取消**]**按鈕。

選項按鈕

可透過其模型物件使用選項按鈕的以下屬性：

Enabled (布林型**)** - 可以啓動控制項元素。

Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

TabIndex (長型**)** - 控制項元素在啓動序列中的位置。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

Tag (字串型**)** - 含有其他資訊的字串，可儲存在按鈕中，用於由程式控制的存取。

Label (字串型**)** - 按鈕上的文字。

Printable (布林型**)** - 可以列印控制項元素。

State (短型**)** - 如果值爲 1，則啓動該選項，否則關閉該選項。

RefValue (字串型**)** - 用於儲存其他資訊的字串 (例如，用於管理資料條目 ID)。

TextColor (長型**)** - 控制項元素的文字顏色。

HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字。

HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL。

透過群組選項按鈕的機制可區分用於對話方塊的控制項元素和用於表單的控制項元素。在對話方塊中並排顯示的控制項元素自動組合成一個群組，而表單中的群組則依名稱執行。爲此，每個群組中的所有選項按鈕都必須包含相同的名稱。StarSuite 將這些已群組的控制項元素組合到一個陣列中，這樣即可使用與上述介紹相同的方式存取 StarSuite Basic 程式的個別按鈕。

以下示例顯示了如何確定控制項元素群組的模型。

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
      Form = Forms.GetbyIndex(I)
      If Form.HasByName("MyOptions") Then
            Ctl = Form. GetGroupbyName("MyOptions")
            Exit Function
      End If
Next I
```

此程式碼與用於確定簡單控制項元素模型的上一示例一致。它透過迴路搜尋目前文字文件內的所有表單，並使用 `HasByName` 方法檢查相應表單是否包含正在搜尋的名爲 `MyOptions` 的元素。如果包含此元素，則使用 `GetGroupByName` 方法 (而不是確定簡單模型的 `GetByName` 方法) 存取模型陣列。。

核取方塊

核取方塊表單的模型物件提供以下屬性：

Enabled (布林型**)** - 可以啓動控制項元素。

Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

TabIndex (長型**)** - 控制項元素在啓動序列中的位置。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

Tag (字串型**)** - 含有其他資訊的字串，可儲存在按鈕中以用於由程式控制的存取。

Label (字串型**)** - 按鈕貼標。

Printable (布林型**)** - 可以列印控制項元素。

State (短型**)** - 如果值爲 1，則啓動該選項，否則關閉該選項。

RefValue (字串型**)** - 用於儲存其他資訊的字串 (例如，用於管理資料條目 ID)。

TextColor (長型**)** - 控制項元素的文字顏色。

HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字。

HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL。

文字欄位

文字欄位表單的模型物件提供以下屬性：

Align (短型**)** - 文字方向 (0：左對齊，1：置中，2：右對齊)。

BackgroundColor (長型**)** - 控制項元素的背景顏色。

Border (短型**)** - 邊框類型 (0：無邊框，1：3D 邊框，2：簡單邊框)。

EchoChar (字串型**)** - 密碼欄位的回應字元。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

HardLineBreaks (布林型**)** - 在控制項元素文字中永久插入自動換行。

HScroll (布林型**)** - 文字具有一個水平捲軸。

MaxTextLen (短型**)** - 最大文字長度；如果指定值爲 0，則無長度限制。

MultiLine (布林型**)** - 允許多行輸入。

Printable (布林型**)** - 可以列印控制項元素。

ReadOnly (布林型**)** - 控制項元素的內容爲唯讀。

Enabled (布林型**)** - 可以啓動控制項元素。

Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

TabIndex (長型**)** - 控制項元素在啓動序列中的位置。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

Text (字串型**)** - 控制項元素的文字。

TextColor (長型**)** - 控制項元素的文字顏色。

VScroll (布林型**)** - 文字具有一個垂直捲軸。

HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字。

HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL。

清單方塊

清單方塊表單的模型物件提供以下屬性：

BackgroundColor (長型**)** - 控制項元素的背景顏色。

Border (短型**)** - 邊框類型 (0：無邊框，1：3D 邊框，2：簡單邊框)。

FontDescriptor (結構型**)** - 含有所用字型細節的結構 (與 `com.sun.star.awt.FontDescriptor` 結構一致)。

LineCount (短型**)** - 控制項元素的行數。

MultiSelection (布林型**)** - 允許多重選擇條目。

SelectedItems (字串陣列**)** - 反白顯示的條目之清單。

StringItemList (字串陣列**)** - 所有條目的清單。

ValueItemList (變體陣列**)** - 含有各條目的其他資訊的清單 (例如，用於管理資料條目 ID)。

Printable (布林型**)** - 可以列印控制項元素。

ReadOnly (布林型**)** - 控制項元素的內容爲唯讀。

Enabled (布林型**)** - 可以啓動控制項元素。

Tabstop (布林型**)** - 可以透過 TAB 鍵存取控制項元素。

TabIndex (長型**)** - 控制項元素在啓動序列中的位置。

FontName (字串型**)** - 字型類型的名稱。

FontHeight (單精度型**)** - 以點數 (pt) 爲單位的字元高度。

Tag (字串型**)** - 含有其他資訊的字串，可儲存在按鈕中以用於由程式控制的存取。

TextColor (長型**)** - 控制項元素的文字顏色。

HelpText (字串型**)** - 將滑鼠游標置於控制項元素上方時自動顯示的說明文字。

HelpURL (字串型**)** - 與控制項元素對應的線上說明的 URL。

透過其 `ValueItemList` 屬性，清單方塊表單可提供一個與 VBA 屬性 `ItemData` 相對應的屬性，用於管理個別清單條目的其他資訊。

此外，還透過清單方塊中的檢視物件提供了以下方法：

addItem (Item, Pos) - 將 `Item` 中指定的字串插入清單中的 `Pos` 位置。

addItems (ItemArray, Pos) - 將在字串的 `ItemArray` 資料欄位中列出的條目插入清單的 `Pos` 位置。

removeItems (Pos, Count) - 自 `Pos` 位置起移除 `Count` 個條目。

selectItem (Item, SelectMode) - 依變數 `SelectMode` 啓動或關閉在字串 `Item` 中指定的元素的反白顯示 。

makeVisible (Pos) - 捲動過清單欄位，以便顯示透過 `Pos` 指定的條目。

資料庫表單

StarSuite 表單可以直接連結到資料庫。以此方式建立的表單，不但不需要獨立的程式設計工作，而且還提供了完整資料庫前端的所有功能。

使用者可以翻頁瀏覽和搜尋選取的表格和查詢，以及變更資料條目或插入新的資料條目等選項。StarSuite 會自動確保可從資料庫擷取到相關的資料，並且所做的任何變更都將寫回資料庫中。

資料庫表單基本上相當於標準的 StarSuite 表單。除標準屬性之外，還必須在表單中設定以下資料庫特有的屬性：

DataSourceName (字串型**)** - 資料源的名稱 (請參閱第 10 章資料庫存取；資料庫存取；必須以全域方式在 StarSuite 中建立資料源)。

Command (字串型**)** - 要爲其建立捷徑的表格、查詢或 SQL select 指令的名稱。

CommandType (常數型**)** - 指定 Command 爲一個表格、查詢還是 SQL 指令 (com.sun.star.sdb.CommandType 列舉中的值)。

com.sun.star.sdb.CommandType 列舉包含以下值：

TABLE - 表格

QUERY - 查詢

COMMAND - SQL 指令

資料庫欄位透過此屬性指定給個別控制項元素：

DataField (字串型**)** - 已連結的資料庫欄位的名稱。

表格

爲了使用資料庫，還提供了另一個控制項元素：表格控制項元素。此元素表示完整的資料庫表格內容或查詢的內容。在最簡單的情況下，會使用自動檔案助理表單將表格控制項元素連結到資料庫，這會將所有的欄與相應的資料庫欄位連結起來 (依照使用者規格)。由於關聯的 API 相當複雜，有關這方面的 API 的內容，本文將不提供完整的描述。

附錄

VBA 遷移提示

StarSuite 5.x 遷移提示

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

T

U

V

W

X

Y