StarSuite™ 7 Office Suite

A Sun™ ONE Software Offering

Basic 编程手册

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

部件号：817-3927-10
2003，修订版 A

Copyrights and Trademarks

Copyright © 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054. , U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.sun.com/patents and one or more additional patents or pending patent applications in the U.S. and in other countries.

This document and the product to which it pertains are distributed under licenses restricting their use, copying, distribution, and decompilation. No part of the product or of this document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any.

Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

This product is based in part on the work of the Independent JPEG Group and The FreeType Project.

Portions Copyright 2000 SuSE, Inc. Word for Word Copyright © 1996 Inso Corp. International CorrectSpell spelling correction system Copyright © 1995 by Lernout & Hauspie Speech Products N.V. All rights reserved.

Sun, Sun Microsystems, the Sun logo, Java, Solaris, Starsuite, the Butterfly logo, the Solaris logo, and the Starsuite logo are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

UNIX is a registered trademark in the U.S. and in other countries, exclusively licensed through X/Open Company, Ltd. Screen Beans and Screen Beans clipart characters are registered trademarks of A Bit Better Corporation.

Federal Acquisitions: Commercial Software - Government Users Subject to Standard License Terms and Conditions.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

版权所有 (c) 2003 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, California 95054. , U.S.A.保留所有权利。

Sun Microsystems, Inc. 持有本文档所述产品中包含的技术的知识产权。特别是 (但不仅限于) 这些知识产权可能包含 http://www.sun.com/patents 中所列的一个或多个美国专利，以及一个或多个美国及其他国家/ 地区的其他专利或待定的专利申请。

本文档及其所含产品受版权保护，其使用、复制、发行和反编译均受许可证限制。未经 Sun 及其许可方的事先书面许可，不得以任何形式、任何手段复制本产品或文档的任何部分。

包括字体技术在内的第三方软件受 Sun 供应商的版权保护和许可证限制。

本产品的某些部分基于独立 JPEG 组和 FreeType 项目的工作成果。

部分版权所有 2000 SuSE, Inc.。Word for Word 版权所有 (c) 1996 Inso Corp.。International CorrectSpell 拼写更正系统版权所有 (c) 1995 Lernout & Hauspie Speech Products N.V.。保留所有权利。

Sun、Sun Microsystems、Sun 徽标、Java、Solaris、StarSuite、蝴蝶徽标、Solaris 徽标和 StarSuite 徽标是 Sun Microsystems, Inc. 在美国和其他国家/ 地区的商标或注册商标。

UNIX 是由 X/Open Company, Ltd. 在美国和其他国家/ 地区独家许可的注册商标。Screen Beans 和 Screen Beans 剪贴画特征是 A Bit Better Corporation 的注册商标。

联邦政府使用：商业软件 - 政府使用者应遵守标准许可证条款和条件。

本文档按“原样”提供，对所有明示或暗示的条件、陈述和担保，包括适销性、适用于特定用途和非侵权的暗示保证，均不承担任何责任，除非此免责声明的适用范围在法律上无效。

内容目录

第 1 章

简介

本书介绍如何使用 StarSuite Basic 6.0 进行编程，并说明使用 StarSuite 中的 StarSuite Basic 可以得到哪些应用程序。要想充分利用本书，还应熟悉其他编程语言。

本书提供了大量示例，可以帮助您快速开发自己的 StarSuite Basic 程序。

整本书中，为 Microsoft Visual Basic 程序员或使用过 StarSuite Basic 早期版本的程序员提供了许多迁移提示，它们由页面边缘处的一个小图标指示。本书的〈附录〉含有所有迁移提示的索引，便于您快速定位到想要阅读的提示。

关于 StarSuite Basic

StarSuite Basic 编程语言是专门为使用 StarSuite 而开发的。此编程语言已经集成于办公软件包之内。

顾名思义，StarSuite Basic 是一种源自 Basic 系列的编程语言。以前使用过其他 Basic 语言 (尤其是 Microsoft 的 Visual Basic 或 Visual Basic for Applications [VBA]) 的使用者，很快就能熟悉 StarSuite Basic。StarSuite Basic 的基本结构大部分与 Visual Basic 兼容。

可以将 StarSuite Basic 编程语言分为四个部分：

- **StarSuite Basic** 语言：定义基本语言结构，例如，变量声明、循环和函数。
- 运行时库：提供不直接引用 StarSuite 的标准函数，例如，用于编辑数字、字串、日期值和文件的函数。
- **StarSuite API (**应用程序编程接口**)**：允许访问 StarSuite 文档，并允许建立、存盘、更改和打印这些文档。
- 对话框编辑器：建立个人对话视窗，并为加入控制元素和事件处理程序提供空间。

StarSuite Basic 和 VBA 之间的兼容性与 StarSuite Basic 语言和运行时库相关。StarSuite API 和对话框编辑器与 VBA 不兼容 (标准化这些接口会使 StarSuite 中提供的许多概念无法实现)。

StarSuite Basic 的最佳使用者

StarSuite Basic 扩充了 StarSuite 标准功能。因此，可以在 StarSuite Basic 中自动执行例行任务，可以建立与其他程序的链接 (例如，与数据库服务器的链接)，还可以使用预设脚本在单击按钮时执行复杂活动。

StarSuite Basic 可以完全访问 StarSuite 的所有功能，支持所有的功能，还可以修改文档类型，以及提供用于建立个人对话视窗的选项。

StarSuite Basic 的使用

任何 StarSuite 使用者，无需任何附加程序和辅助工具，就可以使用 StarSuite Basic。
即使在标准安装中，StarSuite Basic 也具有建立自己的 Basic 宏所需的所有组件，包括：

- 集成开发环境 (IDE)，它提供一个用于输入和测试宏的编辑器。
- 解释程序，执行 StarSuite Basic 宏时需要该程序。
- 接口，该接口指向允许直接访问 Office 文档的各种 StarSuite 应用程序。

本书的结构

前三章向读者简要介绍 StarSuite Basic：

- 第 2 章：StarSuite Basic 语言
- 第 3 章：StarSuite Basic 运行时库
- 第 4 章：StarSuite API 简介

这三章对 StarSuite Basic 进行概述，打算编写 StarSuite Basic 程序的任何人都应该阅读这三章。

其余各章更为详尽地描述 StarSuite API 的各个组件，读者可根据需要有选择地阅读：

- 第 5 章：使用 StarSuite 文档
- 第 6 章：文本文档
- 第 7 章：工作表文档
- 第 8 章：绘图和演示文稿
- 第 9 章：图表
- 第 10 章：访问数据库
- 第 11 章：对话框
- 第 12 章：表单

更多信息

本书中讨论的 StarSuite API 组件是根据它们对 StarSuite Basic 程序员的实际价值而选择出来的。一般只讨论部分接口。如果需要更多信息，可以通过国际互联网上的以下地址获得 API 参考：

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

《StarSuite 开发者指南》比本书更为详尽地描述了 StarSuite API，但该指南主要针对 Java 和 C++ 程序员。已经熟悉 StarSuite Basic 编程的任何人都可以在《StarSuite 开发者指南》中找到有关 StarSuite Basic 和 StarSuite 编程的附加信息。可以通过国际互联网上的以下地址下载《StarSuite 开发者指南》：

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

希望直接使用 Java 或 C++ 而不是 StarSuite Basic 的程序员应查阅《StarSuite 开发者指南》，而不是本书。而与采用 StarSuite Basic 进行编程相比，采用 Java 或 C++ 进行 StarSuite 编程是一个复杂得多的过程。

第 2 章

StarSuite Basic 语言

StarSuite Basic 属于 Basic 语言系列。StarSuite Basic 中的许多部分与 Microsoft Visual Basic for Applications 和 Microsoft Visual Basic 相同。用过这些语言的任何人都可以很快熟悉 StarSuite Basic。

其他语言 (例如 Java、C++ 或 Delphi) 的程序员也会发现很容易熟悉 StarSuite Basic。StarSuite Basic 是一种彻底开发的过程编程语言，它不再使用不完善的控制结构，例如 `GoTo` 和 `GoSub`。

由于通过 StarSuite Basic 中的接口可以使用外部对象库，因此还可以利用面向对象编程的优点。整个 StarSuite API 就是基于这样一些接口，将在后面更加详细地描述这些接口。

本章概述 StarSuite Basic 语言的关键元素和结构，以及 StarSuite Basic 中应用程序和程序库的框架。

StarSuite Basic 程序概述

StarSuite Basic 是一种解释程序语言。与 C++ 和 Turbo Pascal 不同，StarSuite 编译器不建立能够自动执行的可执行文件或自解压文件。但是，可以通过单击某个按钮来执行 StarSuite Basic 程序。首先检查代码，看是否有明显错误，然后逐行执行。

程序行

Basic 解释程序的面向行工作方式是 Basic 与其他编程语言的主要区别之一。在 Java、C++、Delphi 等程序的源代码中，在何处进行硬换行都没有关系，而在 Basic 程序中，各行均形成一个自包含单元。函数调用、数学表达式以及象函数和循环的开头部分这样的其他语言元素必须在同一行中完成。

如果没有足够空间，或者这样会导致行过长，则可以通过加入下划线 _ 将多个行链接起来。以下示例显示了如何将一个数学表达式中的四行链接起来：

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

下划线必须是一个链接行的最后一个字符，而且其后不能带有空格和制表符，否则代码就会生成错误。

除了链接各行以外，在 StarSuite Basic 中，程序员可以使用冒号将一行分为若干区段，这样，就有足够空间容纳多个表达式。例如，赋值

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

可以写成以下形式：

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

注解

除了要执行的程序代码以外，StarSuite Basic 程序中还可以含有注解，用于解释程序的各个部分并提供重要信息，借助这些信息可以在以后某个时候 (例如故障排除部分) 再现程序代码。

StarSuite Basic 提供两种在程序代码中插入注解的方法：

- 单引号后面的所有字符都会被视为注解：

  ```
  Dim A    ' 这是变量 A 的注解
  ```

- 关键字 Rem，后跟注解。

  ```
  Rem 此注解由关键字 Rem 引入。
  ```

一条注解通常包括向右直到行末的所有字符。然后，StarSuite Basic 接着按照常规说明解释后续行。如果注解占用若干行，则应将其中的每行都标识为注解：

```
Dim B       ' 变量 B 的此注解相当长
            ' 且占用多行 。
            ' 因此必须在每行重复
            ' 注解字符。
```

标记

一个 StarSuite Basic 程序可以含有几十个、数百个甚至数千个标记 (变量、常数、函数等的名称)。为一个标记选择名称时，以下条件适用：

- 标记只能含有拉丁字母、数字和下划线 (_)。
- 标记的第一个字符必须是字母或下划线。
- 标记不能含有特殊字符，例如，ä、â、î、ß 等。
- 标记的最大长度为 255 个字符。
- **不区分字符的大小写。例如，OneTestVariable 标记既可以定义 onetestVariable 变量，也可以定义 ONETESTVARIABLE 变量。**

  但是，此规则有一个例外情况：UNO-API 常数区分字符的大小写。如果需要有关 UNO 的更多信息，请参阅第 4 章。

StarSuite Basic 中用于构造标记的规则与 VBA 中的不同。例如，与 VBA 不同，

StarSuite Basic 不允许标记中含有特殊字符，因为特殊字符在国际项目中会造成问题。

下面是正确标记和错误标记的几个示例：

```
Surname             ' 正确
Surname5            ' 正确 (数字 5 不是第一个字符)
First Name          ' 错误 (不允许有空格)
DéjàVu                    ' 错误 (不允许有诸如 é、à 之类的字母)
5Surnames           ' 错误 (第一个字符不能是数字)
First,Name          ' 错误 (不允许有逗号和句点)
```

使用变量

隐式变量声明

Basic 语言设计得非常易于使用。因此，在 StarSuite Basic 中，通过直接使用就可以建立一个变量，而无需进行显式声明。换句话说，从将一个变量包含在代码中的那个时刻起，该变量就已存在。以下代码段最多可以声明三个新变量，具体的新变量个数取决于已经存在的变量：

```
a = b + c
```

隐式声明变量不是一种好的形式，而且会导致由于键入错误等而无意中引入新的变量。对于键入错误，解释程序不会生成错误报告，而只是将其初始化成一个值为 0 的新变量。在代码中，这类错误很难发现。

显式变量声明

为了避免隐式变量声明导致的错误，StarSuite Basic 提供了一个切换，即：

```
Option Explicit
```

这必须在每个模块的第一个程序行中列出，并确保如果没有声明就使用变量，则发出错误报告。`Option Explicit` 切换应该包含在所有 Basic 模块中。

显式变量声明命令的最简单形式如下：

```
Dim MyVar
```

此示例声明一个变量，其名称为 `MyVar`，类型为变体。变体是一种通用变量，可以记录所有可能的值，包括字串、整数、浮点数和布尔值。下面是变体变量的几个示例：

```
MyVar = "Hello World"               ' 用一个字串进行赋值
MyVar = 1                           ' 用一个整数进行赋值
MyVar = 1.0                         ' 用一个浮点数进行赋值
MyVar = True                        ' 用一个布尔值进行赋值
```

上面声明的变量甚至可以用于同一个程序中的不同变量类型。尽管这会提供很大的灵活性，但最好将一个变量限于一种变量类型。当 StarSuite Basic 在某个特定上下文中遇到错误定义的变量类型时，就会生成一个错误报告。

进行限定类型的变量声明时，请使用以下样式：

```
Dim MyVar As Integer    ' 声明一个整数类型的变量
```

该变量被声明为整数类型，可以记录整数值。也可以使用以下样式来声明整数类型的变量：

```
Dim MyVar%              ' 声明一个整数类型的变量
```

`Dim` 指令可以记录多个变量声明：

```
Dim MyVar1, MyVar2
```

如果要将变量指定成某种类型，就必须针对每个变量进行单独指定：

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

如果不为变量声明类型，StarSuite Basic 就将该变量指定成变体类型。例如，在下面的变量声明中，`MyVar1` 是一个变体，而 `MyVar2` 则是一个整数：

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

以下几节列出 StarSuite Basic 中可以使用的变量类型，并描述如何使用和声明这些变量类型。

字串

字串和数字是 StarSuite Basic 中最重要的基本类型。字串由一列连续的单个字符组成。计算机内部将字串存盘为一列数字，每个数字代表一个特定字符。

从一种 ASCII 字符集到统一码

字符集将一个字串中的字符与一个表中相应的代码 (数字和字符) 相匹配，此表描述计算机如何在屏幕上或在打印机上打印该字串。

ASCII 字符集

ASCII 字符集是一组表示数字、字符、特殊符号的单字节代码。0 到 127 ASCII 码对应于字母表和常用符号 (例如句点、括号和逗号) 以及一些特殊屏幕和打印机控制代码。ASCII 字符集常用作计算机之间传送文字数据的标准格式。

但是，此字符集不包括欧洲使用的各种特殊字符 (例如 â、ä 和 î)，以及诸如西里尔字母表之类的其他字符格式。

ANSI 字符集

Microsoft 的 Windows 产品基于美国国家标准学会 (ANSI) 字符集，这种字符集逐渐扩展为包括 ASCII 字符集中没有的字符。

代码页

ISO 8859 字符集提供了一个姗姗来迟的国际标准。ISO 字符集的前 128 个字符与 ASCII 字符集相对应。ISO 标准引入新的字符集 (代码页)，以便更多语言能够正确地被显示。然而，结果是，相同的字符值在不同的语言中表示不同的字符。

统一码

统一码将一个字符的长度增加到四个字节，并将不同的字符集组合起来，以建立一种可以描述世界上尽可能多的语言的标准。统一码 2.0 版现在得到许多程序的支持，其中包括 StarSuite 和 StarSuite Basic。

字串变量

StarSuite Basic 用统一码中的字串变量存盘字串。一个字串变量最多可以存储 65535 个字符。在内部，StarSuite Basic 存盘每个字符相关联的统一码值。一个字串变量所需的工作内存取决于字串的长度。

字串变量声明示例：

```
Dim Variable As String
```

也可以将此声明写为：

```
Dim Variable$
```

移植 VBA 应用程序时，请确保遵守 StarSuite Basic 中允许的最大字串长度 (65535 个字符)。

显式字串规范

要将一个显式字串赋给一个字串变量，请将该字串放在引号 (") 中。

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

要将一个字串分开使其位于两行上，请在第一行末尾添加一个加号：

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

要在字串中包含一个引号 (")，请在相应点处输入引号两次：

```
Dim MyString As String
MyString = "a ""-quotation mark."   ' 生成一个 " - 引号
```

数字

StarSuite Basic 支持五种处理数字的基本类型：

- 整数
- 长整数
- 单精度
- 双精度
- 货币

整数变量

整数变量可以存储 -32768 和 32767 之间的任何整数。一个整数变量占用两个字节内存。整数变量的类型声明符号为 % 。整数变量的计算速度非常快，这非常适合用作循环计数器。如果将一个浮点数赋给一个整数变量，该数字会被四舍五入为接近的整数。

整数变量的声明示例：

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

长整数变量

长整数变量可以存储 2147483648 和 2147483647 之间的任何整数。长整数变量占用四个字节内存。长整数的类型声明符号为 & 。长整数变量的计算速度非常快，这非常适合用作循环计数器。如果将一个浮点数赋给一个长整数变量，该数字会被四舍五入为接近的整数。

长整数变量的声明示例：

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

单精度变量

单精度变量可以存储 3.402823 x 10^{38} 和 1.401298 x 10^{-45} 之间的任何正或负浮点数。一个单精度变量占用四个字节内存。单精度变量的类型声明符号为 ！ 。

最初，使用单精度变量是为了减少更为精确的双精度变量所需的计算时间。但是，由于不再考虑这些速度问题，因而对单精度变量的需要也相应地减少。

单精度变量的声明示例：

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

双精度变量

双精度变量可以存储 1.79769313486232 x 10^{308} 和 4.94065645841247 x 10^{-324} 之间的任何正或负浮点数。一个双精度变量占用八个字节内存。双精度变量适合进行精确计算。类型声明符号为 # 。

双精度变量的声明示例：

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

货币变量

货币变量在处理值的方式上与其他变量类型不同。小数点固定，后面带有四位小数。该变量的小数点之前最多可以含有 15 个数字。货币变量可以存储 -922337203685477.5808 和 +922337203685477.5807 之间的任何值，占用八个字节内存。货币变量的类型声明符号为 @ 。

货币变量主要用于商业计算，在此类计算中，使用浮点数会产生无法预料的舍入误差。

货币变量的声明示例：

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

显式数字规范

可以采用多种方式显示数字，例如，以十进制格式或以科学计数法，甚至使用十进制系统以外的其他基数。以下规则适用于 StarSuite Basic 中的数字字符：

整数

整数的使用最简单。只需要在源文字中列出整数，并且无需用逗号分隔千位数字：

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

数字前面可以有一个加号 (+) 或减号 (-) (数字与符号之间可以有空格，也可以没有空格)：

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

小数

输入小数时，请使用句点 (.) 作为小数点。此规则确保无需转换即可将源文字从一个国家/ 地区传送到另一个国家/ 地区。

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

A = 1223.53       ' 被四舍五入
B = - 23446.46    ' 被四舍五入
C = + 3532.76323
```

小数前面也可以使用加号 (+) 或减号 (-) (同样可以有空格，也可以没有空格)。

如果将一个小数赋给一个整数变量，StarSuite Basic 会将该数四舍五入。

指数书写样式

StarSuite Basic 允许以指数书写样式指定数字，例如，可以将数字 1.5×10^{-10} (0.00000000015) 写成 1.5e-10。字母“e”可以是小写，也可以是大写，其前可以有一个加号 (+)，也可以没有。

下面是用指数格式表示数字的几个正确和错误示例：

```
Dim A As Double

A = 1.43E2            ' 正确
A = + 1.43E2          ' 正确 (加号和基本数字之间有空格)
A = - 1.43E2          ' 正确 (减号与基本数字之间有空格)
A = 1.43E-2           ' 正确 (负指数)

A = 1.43E -2          ' 错误 (数字中不允许有空格)
A = 1,43E-2           ' 错误 (不允许用逗号作为小数点)
A = 1.43E2.2          ' 错误 (指数必须为整数)
```

请注意，在第一个和第三个错误示例中，尽管变量返回错误的数值，但不会生成错误报告。表达式

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

被解释为 1.43 减去 2，与值 -0.57 相对应。但是，值 $1.43 * 10^{-2}$ (对应于 0.0143) 是目标值。对于值

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

StarSuite Basic 忽略小数点后面的指数部分，并将表达式解释为

```
A = 1.43E2
```

十六进制值

十六进制系统 (基数为 16 的系统) 的优点是，每两位数字精确地对应于一个字节。这样就可以通过一种更接近于反映计算机体系结构的方式处理数字。在十六进制系统中，将数字 0 到 9 以及字母 A 到 F 用作数字。字母 A 表示十进制数字 10，而字母 F 表示十进制数字 15。StarSuite Basic 允许使用十六进制的整数值，但整数前面应加上 &H。

```
Dim A As Longer

A = &HFF      ' 十六进制值 FF，对应于十进制值 255
A = &H10      ' 十六进制值 10，对应于十进制值 16
```

八进制值

StarSuite Basic 还能理解使用 0 到 7 的数字的八进制系统 (基数为 8 的系统)，但整数前面应加上 &O。

```
Dim A As Longer

A = &O77      ' 八进制值 77，对应于十进制值 63
A = &O10      ' 八进制值 10，对应于十进制值 8
```

True 和 False

布尔变量

布尔变量只能含有以下两个值之一：`True` 和 `False`。它们适合于只能采用已命名状态之一的二进制定义。布尔值被内部存盘为两字节整数值，其中，0 对应于 `False`，而任何其他值对应于 `True`。布尔变量没有类型声明符号，只能使用补充的 *As Boolean* 进行声明。

布尔变量的声明示例：

```
Dim Variable As Boolean
```

日期和时间

日期变量

日期变量可以含有日期和时间值。存盘日期值时，StarSuite Basic 使用一种内部格式，该格式允许对日期和时间值进行比较和数学运算。日期变量没有类型声明符号，只能使用补充的 *As Date* 进行声明。

日期变量的声明示例：

```
Dim Variable As Date
```

数据字段

除了简单变量 (标量) 外，StarSuite Basic 还支持数据字段 (数组)。数据字段含有多个变量，这些变量通过一个索引进行寻址。

简单数组

数组声明与简单变量声明相似，但与变量声明不同的是，数组名称后面有括号，其中含有元素数目定义。表达式

```
Dim MyArray(3)
```

声明了一个数组，其中含有四个变体数据类型的变量，即 `MyArray(0)`、`MyArray(1)`、`MyArray(2)` 和 `MyArray(3)`。

也可以在一个数组中声明特定类型的变量，例如，以下行声明一个具有四个整数变量的数组：

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

在上面的示例中，数组的索引总是以标准起始值零开始。另外，也可以对数据字段声明指定一个含有起始值和最终值的有效性区域。以下示例声明一个含有六个整数值的数据字段，而且该数据字段可使用索引 5 到 10 进行寻址：

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

索引不必为正值。以下示例也显示的是一个正确声明，但具有负的数据字段限制值。

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

它声明一个具有六个值的整数数据字段，可以使用索引 -10 到 -5 进行寻址。

定义数据字段索引时必须遵守三个限制值：

- 可能的最小索引为 -32768。
- 可能的最大索引为 32767。
- 最大的元素数目 (在一个数据字段维数中) 为 16368。

其他限制值有时适用于 VBA 中的数据字段索引。同样也适用于每一维上可能的最大元素数目。对应的有效值可在相关的 VBA 文档中找到。

起始索引的指定值

一个数据字段的起始索引通常以值 0 开始。也可以调用以下命令将所有数据字段声明的起始索引更改为值 1：

```
Option Base 1
```

如果要对某个模块中的所有数组声明采用此调用，就必须将它包含在该模块的标头中。但是，此调用不影响通过 StarSuite API 定义的 UNO 序列，该序列的索引始终以 0 开始。要使程序更清晰，应避免使用 Option Base 1。

如果使用 Option Base 1，数组中元素的数目不会受到影响，只有起始索引会发生变化。声明

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

建立了 4 个整数变量，它们可以用表达式 `MyInteger(1)`、`MyInteger(2)`、`MyInteger(3)` 和 `MyInteger(4)` 进行描述。

与 VBA 不同，在 StarSuite Basic 中，表达式 Option Base 1 不影响数组中的元素数目。它只使 StarSuite Basic 中的起始索引发生改变。在 VBA 中，声明 `MyInteger(3)` 建立索引为 1 到 3 的三个整数值；而在 StarSuite Basic 中，该声明建立索引为 1 到 4 的四个整数值。

多维数据字段

除了单维数据字段之外，StarSuite Basic 还支持使用多维数据字段。相应的维之间用逗号分隔。示例

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

定义了一个两维整数数组，每一维都具有 6 个索引（可通过索引 0 到 5 进行寻址）。整个数组总共可记录 6 × 6 = 36 个整数值。

尽管可以在 StarSuite Basic 数组中定义成百上千个维数，但是，可以使用的内存数量限制了可以拥有的维数。

数据字段维数的动态更改

上面的示例基于已指定维数的数据字段。也可以定义数据字段的维数可以动态更改的数组。例如，可以定义一个数组，该数组含有一个文本中所有以字母 A 开头的字词。由于最初不知道这些字词的数目，因此需要以后能够更改字段限制值。为此，在 StarSuite Basic 中使用以下调用：

```
ReDim MyArray(10)
```

与 VBA 中只能使用 Dim MyArray() 定义动态数组维数不同，在 StarSuite Basic 中，可以使用 ReDim 更改静态和动态数组。

以下示例更改初始数组的维数，这样该数组就可以记录 11 或 21 个值：

```
Dim MyArray(4) As Integer           ' 带有五个元素的声明

' ...

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

' ...

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

当重设某个数组的维数时，可以使用前面几节中概述的任何选项。这包括声明多维数据字段和/ 或指定明确的起始值和最终值。更改数据字段的维数后，所有内容会被丢失。如果要保留原始数值，请使用 Preserve 命令：

```
Dim MyArray(10) As Integer                          ' 定义初始
                                                    ' 维数

' ...

ReDim Preserve MyArray(20)   As Integer             ' 数据字段中的
                                                    ' 维数增加时，
                                                    ' 保留内容
```

使用 Preserve 时，确保维数和变量类型保持相同。

在 VBA 中，使用 Preserve 时只能更改一个数据字段的最后一维的上限；而在 StarSuite Basic 中，使用该命令还可以更改其他维。

如果将 ReDim 与 Preserve **一起使用，就必须使用**原始数据字段声明中指定的数据类型。

变量的作用域和生存期

StarSuite Basic 中的变量具有有限的生存期和有限的作用域，在这个范围中，其他程序段中可以读取和使用该变量。一个变量可以被保留的时间长度和可以被访问的位置都取决于其指定的位置和类型。

局部变量

在函数或过程中声明的变量称为局部变量：

```
Sub Test
      Dim MyInteger As Integer

      ' ...
End Sub
```

局部变量仅在该函数或过程执行时有效，然后会被重设为零。每次调用该函数时，以前生成的值就不再可用。

要保留以前的值，必须将变量定义为**静态**：

```
Sub Test
      Static MyInteger As Integer

      ' ...

End Sub
```

与 VBA 不同，StarSuite Basic 确保不会在模块头部中将一个局部变量的名称同时用作全局变量和私有变量。将 VBA 应用程序导入 StarSuite Basic 时，必须更改任何重复的变量名称。

公用域变量

公用域变量是通过关键字 `Dim` 在一个模块的头部区域中定义的。这些变量可用于其库中的所有模块：

模块 A：

```
Dim A As Integer

Sub Test
      Flip
      Flop
End Sub

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

模块 B：

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

变量 `A` 的值不是通过 `Test` 函数更改的，而是在 `Flip` 函数中被递增 1，在 `Flop` 函数中被递减 1。该变量的这两种更改都是全局性的。

也可以使用关键字 `Public` 代替 `Dim` 来声明一个公用域变量：

```
Public A As Integer
```

一个公用域变量仅在关联的宏执行时可用，然后会被重设。

全局变量

就其功能而言，全局变量与公用域变量相似，不同的是全局变量的值在执行关联的宏之后仍然保留。全局变量是使用关键字 `Global` 在一个模块的头部区域中声明的：

```
Global A As Integer
```

私有变量

私有 变量只能在定义它们的模块中使用。可以使用关键字 `Private` 定义私有变量：

```
Private MyInteger As Integer
```

如果多个模块中含有相同名称的**私有**变量，StarSuite Basic 会在该名称每次出现时建立一个不同的变量。在以下示例中，模块 `A` 和 `B` 都具有一个**私有**变量 `C`。`Test` 函数先在模块 `A` 中设定**私有**变量，然后在模块 `B` 中设定**私有**变量。

模块 A：

```
Private C As Integer

Sub Test
      SetModuleA            ' 设定模块 A 中的变量 C
      SetModuleB            ' 设定模块 B 中的变量 C

      ShowVarA              ' 显示模块 A 中的变量 C (等于 10)
      ShowVarB              ' 显示模块 B 中的变量 C (等于 20)
End Sub

Sub SetmoduleeA
      A = 10
End Sub

Sub ShowVarA
      MsgBox C              ' 显示模块 A 中的变量 C。
End Sub
```

模块 B：

```
Private C As Integer

Sub SetModuleB
     A = 20
End Sub

Sub ShowVarB
      MsgBox C              ' 显示模块 B 中的变量 C。
End Sub
```

常数

在 StarSuite Basic 中，使用关键字 `Const` 来声明常数。

```
Const A = 10
```

如果需要，还可以在声明中指定常数类型：

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

运算符

StarSuite Basic 识别通用数学运算符、逻辑运算符和比较运算符。

数学运算符

数学运算符适用于所有数字类型，而 + 运算符还可用于链接字串。

`+` 数字或日期值之间的相加以及字串链接

`-` 数字或日期值之间的相减

`*` 数字之间的相乘

`/` 数字之间的相除

`\` 数字之间的相除且结果为一个整数 (被四舍五入)

`^` 计算数字的幂

`MOD` 模运算 (计算除法的余数)

逻辑运算符

使用逻辑运算符，可以按照布尔代数规则链接元素。如果对布尔值采用运算符，链接将直接提供需要的结果。如果与整数值和长整数值一起使用，则在位级进行链接。

`AND` 和链接

`OR` 或链接

`XOR` 异或链接

`NOT` 否定

`EQV` 相等测试 (`True` 或 `False` 两部分)

`IMP` 隐含 (如果第一个表达式为真，则第二个表达式也为真)

比较运算符

比较运算符适用于所有基本变量类型 (数字、日期、字串和布尔值)。

`=` 适用于数字、日期值和字串的相等比较运算符

<>　　适用于数字、日期值和字串的不相等比较运算符

>　　适用于数字、日期值和字串的大于比较运算符

>=　　适用于数字、日期值和字串的大于等于比较运算符

<　　适用于数字、日期值和字串的小于比较运算符

>=　　适用于数字、日期值和字串的小于等于比较运算符

StarSuite Basic 不支持 VBA 中的 Like 比较运算符。

分支

使用分支语句可以限制代码块的执行，直到满足某个特殊条件。

If...Then...Else

最常用的分支语句是 If 语句，如以下示例所示：

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

只有当变量 A 的值大于 3 时，B = 2 赋值才会发生。If/Else 子句是 If 语句的变体：

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

在此示例中，当 A 大于 3 时，变量 B 的值被指定为 2，否则，B 的值被指定为 0。

如果需要更复杂的语句，可以级联 If 语句，例如：

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

如果变量 A 的值等于零，就将 B 的值指定为 0。如果 A 小于 3 (但是不等于 0)，则将 B 的值指定为 1。在所有其他情况下 (也就是说，如果 A 大于等于 3)，就将 B 的值指定为 2。

Select...Case

Select...Case 指令可替代级联的 If 语句，它用于需要按照各种条件来检查某个值的场合：

```
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，一个工作日的名称对应于一个数字，因此，当名称为 Sunday 时，将 DayOfWeek 变量的值指定为 1，当名称为 Monday 时，将该变量的值指定为 2，依此类推。

Select 命令不限于简单的 1:1 赋值，也可以在一个 Case 分支中指定比较运算符或表达式列单。以下示例列出最重要的语法变体：

```
Select Case Var
Case 1 To 5

      ' ... Var 介于数字 1 和 5 之间

Case 6, 7, 8

      ' ... Var 为 6、7 或 8

Case Var > 8 And Var < 11

      ' ... Var 大于 8，且小于 11

Case Else

      ' ... 所有其他情况

End Select
```

循环

循环是按照指定的遍数执行某个代码块。也可以有未定义遍数的循环。

For...Next

For...Next 循环具有固定的遍数。循环计数器定义循环要执行的次数。在以下示例中，

```
Dim I

For I = 1 To 10

      ' ... 循环的内部

Next I
```

变量 I 为循环计数器，其初始值为 1。在每遍结束时，计数器递增 1。当变量 I 等于 10 时，循环停止。

如果要在每遍结束时使循环计数器递增一个非 1 的值，请使用 Step 功能：

```
Dim I

For I = 1 To 10 Step 0.5

      ' ... 循环的内部

Next I
```

在上面的示例中，计数器在每遍结束时递增 0.5，该循环执行了 19 次。

也可以使用负的 Step 值：

```
Dim I

For I = 10 To 1 Step -1

      ' ... 循环的内部

Next I
```

在此示例中，计数器从 10 开始，每遍结束时递减 1，直到计数器为 1。

Exit For 指令用于提前退出某个 For 循环。在以下示例中，在第五遍过程中终止循环：

```
Dim I

For I = 1 To 10

      If I = 5 Then
            Exit For
      End If

      ' ... 循环的内部

Next I
```

StarSuite Basic 不支持 VBA 中的 For Each...Next 循环变体。

Do...Loop

`Do...Loop` 不与一个固定的遍数相链接。相反，`Do...Loop` 一直执行，直到满足某个条件为止。`Do...Loop` 有四种变体 (在以下示例中，`A > 10` 代表任意条件)：

1. `Do While...Loop` 变体

```
  Do While A > 10
      ' ... 循环主体
  Loop
```

   每遍之前检查是否仍然满足条件，只有满足条件时才执行循环。

2. `Do Until...Loop` 变体

```
Do Until A > 10
      ' ... 循环主体
Loop
```

   执行循环，直到**不再**满足条件。

3. `Do...Loop While` 变体

```
Do
      ' ... 循环主体
Loop While A > 10
```

   在执行第一遍循环之后才检查条件，如果**满足**条件，就终止循环。

4. `Do...Loop Until` 变体

```
Do
      ' ... 循环主体
Loop Until A > 10
```

   也是在第一遍之后检查循环的条件，但继续执行循环，直到**不再**满足条件为止。

象 `For...Next` 循环中一样，`Do...Loop` 也提供一个终止命令。`Exit Do` 命令可以在循环中的任何位置退出循环。

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

      ' ... 循环主体
While A > 10
```

编程示例：用嵌入的循环进行排序

使用循环的方法有许多种，例如，搜索列单、返回值或执行复杂的数学任务。以下示例是一个使用循环按名称对列单进行排序的算法。

```
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
```

值被成对交换若干次，直到最后按升序排序。象气泡一样，变量逐渐迁移到恰当的位置。基于这个原因，此算法也称为冒泡排序。

过程和函数

过程和函数形成一个程序结构中的枢轴点。它们提供将一个复杂问题分为各种子任务的框架。

过程

过程用于执行操作，它不提供显式值。其语法是

```
Sub Test

      ' ... 此处是过程的实际代码

End Sub
```

该示例定义了一个称为 Test 的过程，可从程序中的任意位置访问该过程含有的代码。可以通过在程序的相关位置输入过程名来调用该过程：

```
Test
```

函数

函数象过程一样，将程序块组合起来，以作为一个逻辑单元来执行。但是，与过程不同的是，函数提供返回值。

```
Function Test

      ' ... 此处是函数的实际代码

      Test = 123
End Function
```

返回值是使用简单赋值来指定的。该赋值不必一定要位于函数末尾，可以在函数的任何位置进行赋值。

可以在某个程序中调用上面的函数，如下所示：

```
Dim A

A = Test
```

该代码定义了一个变量 A，并将 Test 函数的结果赋给该变量。

在函数中可以多次该写返回值。就象典型的变量赋值一样，此示例中的函数返回最后一次指定的函数值。

```
Function Test

      Test = 12

      ' ...

      Test = 123

End Function
```

在此示例中，函数的返回值为 123。

如果不指定，函数就返回**零**值 (对于数字值是数字 0，而对于字串是一个空格)。

一个函数的返回值可以是任意类型，类型的声明方法与变量声明相同：

```
Function Test As Integer

      ' ... 此处是函数的实际代码

End Function
```

如果不指定一个显式值，则返回值的类型被指定为变体。

提前终止过程和函数

在 StarSuite Basic 中，可以使用 Exit Sub 和 Exit Function 命令提前终止过程或函数，例如错误处理。这些命令停止过程或函数，并使程序返回到调用该过程和/ 或函数的地方。

以下示例显示了一个过程，当
ErrorOccured 变量的值为 True 时，该过程将终止实现。

```
Sub Test
      Dim ErrorOccured As Boolean

      ' ...

      If ErrorOccured Then
            Exit Sub
      End If

      ' ...

End Sub
```

传递参数

函数和过程可以接收一个或多个参数。必须将需要的参数放在函数或过程名称后面的括号中。示例

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

```
End Sub
```

定义了一个过程，该过程需要一个整数值 A 和一个字串 B 作为参数。

在 StarSuite Basic 中，通常通过引用来传递参数。退出过程或函数时，对变量所做的更改会被保留：

```
Sub Test
      Dim A As Integer
      A = 10
      ChangeValue(A)
      ' 参数 A 现在的值为 20
End Sub
Sub ChangeValue(TheValue As Integer)
       TheValue = 20
End Sub
```

在此示例中，Test 函数中定义的值 A 被作为参数传递给 ChangeValue 函数。然后该值被更改为 20 并被传递给 TheValue，当退出函数时，该值被保留。

如果不希望以后更改参数将会影响原来传递的值，还可以将参数作为值进行传递。要指定将一个参数作为值传递，请确保在函数头部的变量声明前加上 ByVal 关键字。

在上面的示例中，如果用以下函数替换 ChangeValue 函数

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

，则上级变量 A 不会受到此更改的影响。调用 ChangeValue 函数之后，变量 A 的值仍保持为 10。

在 StarSuite Basic 中，将参数传递到过程和函数的方法与 VBA 中的几乎相同。默认情况下，参数按引用传递。要将参数作为值传递，请使用 ByVal 关键字。在 VBA 中，还可以使用关键字 ByRef 来强制按引用传递参数。StarSuite Basic 不支持此关键字，因为在 StarSuite Basic 中，这已经是默认过程。

原则上，StarSuite Basic 中的函数和过程为 Public。StarSuite Basic 不支持 VBA 中使用的 Public 和 Private 关键字。

可选参数

只有当在调用时传递了所有必需参数后，才能调用函数和过程。

在 StarSuite Basic 中，可以将参数定义为**可选的**，也就是说，如果调用时不包括相应的值，StarSuite Basic 就会传递一个空参数。在示例

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

```
End Sub
```

中，参数 A 是必需的，而参数 B 是可选的。

IsMissing 函数用于检查某个参数是被传递了，还是被省略了。

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

      ' 检查 B 参数是否实际存在
      If Not IsMissing (B) Then
             B_Local = B            ' B 参数存在
      Else
             B_Local = 0            ' 缺少 B 参数 -> 默认值 0
      End If

      ' ... 启动实际函数

End Sub
```

示例首先测试是否已经传递 B 参数，如果已经传递，将该参数传递给内部 B_Local 变量。如果相应的参数不存在，则将默认值 (在此实例中，默认值为 0) 而不是传递的参数传递给 B_ Local。

StarSuite Basic 不支持 VBA 中为定义可选参数的默认值而提供的选项。

StarSuite Basic 不支持 VBA 中存在的 ParamArray 关键字。

递归

StarSuite Basic 中现在支持递归。递归过程或函数能够一直调用其本身，直到检测到已经满足某个基本条件。用基本条件调用递归函数时，会返回一个结果。

以下示例使用递归函数来计算数字 42、-42 和 3.14 的阶乘。

```
Sub Main
 Msgbox CalculateFactorial( 42 ) ' 显示 1,40500611775288E+51
 Msgbox CalculateFactorial( -42 ) ' 显示 "Invalid number for factorial!"
 Msgbox CalculateFactorial( 3.14 ) ' 显示 "Invalid number for factorial!"
End Sub

Function CalculateFactorial( Number )
 If Number < 0 Or Number <> Int( Number ) Then
 CalculateFactorial = "Invalid number for factorial!"
 ElseIf Number = 0 Then
 CalculateFactorial = 1
Else
 ' 这是递归调用：
 CalculateFactorial = Number * CalculateFactorial( Number - 1 )
 Endif
End Function
```

该示例通过递归调用 CalculateFactorial 函数，直到满足基本条件 0! = 1 为止，来返回数字 42 的阶乘。

请注意，目前 StarSuite Basic 中的递归级别限于 500。

错误处理

对错误情形进行纠正处理是编程中最耗时的工作之一。StarSuite Basic 提供了一系列简化错误处理的工具。

On Error 指令

On Error 指令是任何错误处理的关键：

```
Sub Test
      On Error Goto ErrorHandler

      ' ... 执行可能会发生错误的任务

Exit Sub

ErrorHandler:

      ' ... 用于进行错误处理的单独代码

End Sub
```

`On Error Goto ErrorHandler` 行定义当发生错误时 StarSuite Basic 如何继续。`Goto ErrorHandler` 确保 StarSuite Basic 退出当前程序行，然后执行 `ErrorHandler:` 代码。

Resume 命令

`Resume Next` 命令用于在执行错误处理程序中的代码之后，从程序中发生错误的位置的下一行开始继续执行程序：

```
ErrorHandler:

      ' ... 用于进行错误处理的单独代码

      Resume Next
```

使用 `Resume Proceed` 命令可以指定一个跳转点，用于在错误处理之后继续执行程序：

```
ErrorHandler:

      ' ... 用于进行错误处理的单独代码
      Resume Proceed

Proceed:

      ' ... 程序在错误处理之后从此处继续执行
```

要在发生错误时不生成错误报告的情况下继续执行程序，请使用以下格式：

```
Sub Test
      On Error Resume Next

      ' ... 执行可能会发生错误的任务

End Sub
```

请谨慎使用 `On Error Resume Next` 命令，因为其影响是全局性的。如果需要更多信息，请参阅有关结构化错误处理的提示。

有关错误信息的查询

在错误处理中，如果有错误说明并知道发生错误的位置和原因，会非常有用：

- `Err` 变量含有已发生的错误数目。
- `Error$` 变量含有错误描述。
- `Erl` 变量含有已发生的错误所在的行编号。

调用

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

显示如何在一个消息窗口中显示错误信息。

VBA 在一个名为 Err 的统计对象中总结错误报告，而 StarSuite Basic 提供的是 Err、Error$ 和 Erl 变量。

状态信息一直有效，直到程序遇到 Resume 或 On Error 命令，随后将会重设信息。

在 VBA 中，Err 对象的 Err.Clear 方法在错误发生后重设错误状态。而在 StarSuite Basic 中，这是通过 On Error 或 Resume 命令实现的。

有关结构化错误处理的提示

定义命令 On Error 和返回命令 Resume 都是 Goto 结构的变体。

如果要清晰地构造代码，以防止在使用此结构时产生错误，就不应该在无监视的情况下使用跳转命令。

应谨慎使用 On Error Resume Next 命令，因为这会取消所有打开的错误报告。

最好的解决办法是在一个程序中仅使用一种错误处理方法 - 将错误处理与实际程序代码分隔开来，且错误发生后不跳转回到原始代码。

下面是错误处理过程的一个示例：

```
Sub Example

	' 在函数开头处定义错误处理程序
	On Error Goto ErrorHandler

	' ... 此处是实际的程序代码

	' 关闭错误处理
	On Error Goto 0

	' 结束常规程序实现
Exit Sub

' 错误处理的起始位置
ErrorHandler:

	' 检查是否存在预期的错误
	If Err = ExpectedErrorNo Then
		' ... 处理错误
	Else
		' ... 警告存在意外错误
	End If

	On Error Goto 0 				' 关闭错误处理
End Sub
```

此过程的开头处定义了一个错误处理程序，然后是实际的程序代码。在程序代码结尾处，调用 On Error Goto 0 来关闭错误处理，并通过 Exit Sub 命令 (不要与 End Sub 混淆) 来结束过程实现。

该示例首先检查错误编号是否与预期的编号 (存储在虚拟的 `ExpectedErrorNo` 常数中) 相对应，然后相应地处理错误。如果发生另一个错误，系统就会输出一个警告。检查错误编号非常重要，因为这样可以检测到非预期的错误。

代码结尾处调用 `On Error Goto 0` 来重设错误的状态信息 (`Err` 系统变量中的错误代码)，这样就可以清楚地识别以后发生的错误。

第 3 章

StarSuite Basic 运行时库

以下各节介绍运行时库的主要函数。

转换函数

在许多场合，会遇到需要将一种类型的变量更改为另一种类型的变量的情况。

隐式和显式类型转换

将变量从一种类型更改为另一种类型的最简单方法是使用赋值。

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

在此示例中，变量 A 是一个字串，而变量 B 是一个整数。StarSuite Basic 确保在给变量 A 赋值的过程中，将变量 B 转换为一个字串。此转换比看起来要复杂得多：整数 B 以一个两字节长的数字形式保留在工作内存中。而 A 是一个字串，计算机为每个字符 (每个数字) 存盘一个一字节或两字节长的值。因此，在将内容从 B 复制到 A 之前，必须将 B 转换为 A 的内部格式。

与大多数其他编程语言不同，Basic 自动执行类型转换。但是，这可能会导致严重的后果。以下代码序列

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

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

乍看起来非常简单，但经过进一步研究，最后会发现其中存在陷阱。Basic 解释程序首先计算加法过程的结果，然后将此结果转换为一个字串，结果就会得到字串 2。
另外一种情况，如果 Basic 解释程序首先将起始值 B 和 C 转换为一个字串，并对结果采用加号运算符，则会得到字串 11。

使用变体变量时也是如此：

```
Dim A
```

```
Dim B
Dim C

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

由于变体变量可以含有数字和字串，因此，不能确定是将数字 2 还是将字串 11 赋给变量 A。

要避免因隐式类型转换而著称的错误根源，必须以一种严谨的方式编写程序，例如，不使用变体数据类型 (我们再次建议这样做)。

为避免由隐式类型转换而导致的其他错误，StarSuite Basic 提供了一系列转换函数，用于定义一项计算的数据类型应该在何时被转换：

- **CStr(Var)** - 将任何数据类型转换为一个字串。
- **CInt(Var)** - 将任何数据类型转换为整数值。
- **CLng(Var)** - 将任何数据类型转换为长整数值。
- **CSng(Var)** - 将任何数据类型转换为单精度值。
- **CDbl(Var)** - 将任何数据类型转换为双精度值。
- **CBool(Var)** - 将任何数据类型转换为布尔值。
- **CDate(Var)** - 将任何数据类型转换为日期值。

使用这些转换函数可以定义 StarSuite Basic 执行这些类型转换计算的方式：

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

B = 1
C = 1

A = CInt(B + C)              ' 首先将 B 和 C 相加，然后转换
                              (得到数字 2)
A = CStr(B) + Cstr(C)        ' 将 B 和 C 转换为一个字串，然后
                             ' 连接起来 (得到字串“11”)
```

在此示例的第一个加法中，StarSuite Basic 首先将整数变量相加，然后将结果转换为一个字串。A 被赋值为字串 2。在第二种情况中，首先将整数变量转换为两个字串，然后通过赋值将两者链接起来。因此，A 被赋值为字串 11。

CSng 和 CDbl 数字转换函数也接受小数。必须将特定于国家/ 地区的相应设定中定义的符号用作小数点符号。相反，CStr 方法在格式化数字、日期和时间细节时，则使用当前选定的特定于国家/ 地区的设定。

Val 函数与 Csng、Cdbl 和 Cstr 方法不同。它将一个字串转换为一个数字，但是始终需要用句点作为小数点符号。

```
Dim A As String
```

```
Dim B As Double

A = "2.22"
B = Val(A)        ' 在不考虑特定于国家/地区的设定的情况下正确转换
```

检查变量内容

在某些情况下，不能转换日期：

```
Dim A As String
Dim B As Date

A = "test"
B = A                    ' 建立错误报告
```

在上面显示的示例中，将“test”字串赋给一个日期变量没有任何意义，因此，Basic 解释程序会报告一个错误。当尝试将一个字串赋给一个布尔变量时，也是如此：

```
Dim A As String
Dim B As Boolean

A = "test"
B = A                    ' 建立错误报告
```

同样，Basic 解释程序会报告一个错误。

通过在赋值之前检查程序以确定要赋值的变量内容是否与目标变量的类型相匹配，可以避免这些错误报告。
StarSuite Basic 为此提供了以下测试函数：

- **IsNumeric(Value)** - 检查某个值是否为数字。
- **IsDate(Value)** - 检查某个值是否为日期。
- **IsArray(Value)** - 检查某个值是否为数组。

当查询使用者输入时，这些函数尤其有用。例如，可以检查使用者是否输入了有效的数字或日期。

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

在上面的示例中，如果 UserInput 变量含有一个有效的数字值，就将此值赋给 ValidInput 变量。如果 UserInput 不含有有效值，则 ValidInput 的值被指定为 0，而且返回错误报告。

虽然 Basic 中存在用于检查数字、日期和数组的测试函数，却不存在用于检查布尔值的相应函数。但是，可以使用 IsBoolean 函数来模拟该功能：

```
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 函数定义一个布尔类型的 Dummy 内部帮助变量，并尝试将传送的值赋给此变量。如果赋值成功，函数就返回 True。如果赋值失败，就会生成一个运行时错误，这会终止测试函数以返回错误。

如果 StarSuite Basic 中的字串含有一个非数字值，而且将此值赋给一个数字，StarSuite Basic 不会生成错误报告，而是将值 0 传送给变量。此过程与 VBA 不同。在 VBA 中，如果执行相应赋值，就会触发一个错误，并且终止程序实现。

字串

使用字符集

管理字串时，StarSuite Basic 使用统一码字符集。Asc 和 Chr 函数可以确定一个字符的统一码值，并且/ 或者找到统一码值相应的字符。以下表达式将各种统一码值赋给 code 变量：

```
Code = Asc("A")              ' 拉丁字母 A (统一码值 65)
Code = Asc("€")              ' 欧元字符 (统一码值 8364)
Code = Asc("ë")              ' 西里尔字母 _(统一码值 1083)
```

相反，表达式

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

确保用数字值 13 (表示硬换行) 初始化 MyString 字串。

Basic 语言中经常使用 Chr 命令在一个字串中插入控制字符。
因此，赋值

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

确保文字前面有一个制表字符 (统一码值 9) 并在文字后面加入一个硬换行 (统一码值 13)。

读取字串中的部分字符

StarSuite Basic 提供四个返回部分字串的函数：

- **Left(MyString, Length)** - 返回 MyString 的前 Length 个字符。
- **Right(MyString, Length)** - 返回 MyString 的最后 Length 个字符。
- **Mid(MyString, Start, Length)** - 返回 MyString 中从 Start 位置开始的连续 Length 个字符。
- **Len(MyString)** - 返回 MyString 中的字符数目。

下面是调用这些函数的几个示例：

```
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”
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
```

搜寻和替换

StarSuite Basic 提供了 InStr 函数，用于在一个字串中搜寻另一个部分字串：

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

SearchString 参数指定要在 MyString 中搜寻的字串。该函数返回一个数字，该数字含有 SearchString 在 MyString 中首次出现的位置。如果要搜寻该字串的其他匹配，该函数还可以指定 StarSuite Basic 开始搜寻的可选起始位置。在这种情况下，该函数的语法为：

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

在上面的示例中，InStr 忽略字符的大小写。要更改搜寻以使 InStr 区分大小写，请加入参数 0，如以下示例所示：

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

使用上面用于编辑字串的函数，程序员可以在一个字串中搜寻或替换另一个字串：

```
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 <> 0
        Loop
    End If
    Replace = Result
End Function
```

该函数在原始项 Source 中，通过 InStr 以循环的方式搜寻传送的 Search 字串。如果搜寻到搜寻项，就会获取表达式之前的部分并将其写到 Result 返回缓冲区。它在搜寻项 Search 所在位置加入 NewPart 部分。如果找不到搜寻项的更多匹配项，函数就确定仍然保留的字串部分并将其加入到返回缓冲区中。函数返回通过这种方式生成的字串，作为替换过程的结果。

由于替换部分字符序列是最常用的功能之一，StarSuite Basic 中的 Mid 函数已被扩展，从而可以自动执行此任务。以下示例

```
Dim MyString As String
```

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

用字串 `is` 替换 `MyString` 字串中从第六个位置开始的三个字符。

格式化字串

`Format` 函数将数字格式化为字串。为此，该函数需要指定一个**格式**表达式，然后将该表达式用作格式化数字的样式。样式内每个占位符确保在输出值中此项被相应地格式化。一个样式中五个最重要的占位符是**零** (0)、英镑符号 (#)、句点 (.)、**逗号** (,) 和 美**元符号** ($)。

样式中的零字符确保总是在相应位置放置一个数字。如果没有提供数字，就会在对应的位置显示 0。

句点表示在特定于国家/ 地区的设定中由操作系统定义的小数点符号。

下面的示例显示了**零**和句点字符如何定义一个表达式中小数点后的数字：

```
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”
```

同样，可以在一个数字前面加入零，以达到所需的长度：

```
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”
```

逗号代表操作系统使用的千位分隔符，而英镑符号表示一个数字或位置，仅在输入字串时如果需要才会显示。

```
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` 函数用系统定义的相关货币符号代替美元符号占位符进行显示：

```
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 不支持 VBA 中使用的 Format 指令来格式化日期和时间细节。

日期和时间

StarSuite Basic 提供了**日期**数据类型，它以二进制格式存盘日期和时间细节。

程序代码中的日期和时间细节规范

通过指定一个简单字串，可以将某个日期赋给一个日期变量：

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

此赋值可以正常工作，因为 StarSuite Basic 自动将作为字串定义的日期值转换为一个日期变量。但是，此赋值类型可能会导致错误，日期和时间值在不同国家/ 地区以不同的方式定义和显示。

由于 StarSuite Basic 在将字串转换为日期值时，使用特定于国家/ 地区的操作系统设定，因而只有当特定于国家/ 地区的设定与字串表达式相匹配时，上面显示的表达式才能正常工作。

为了避免此问题，应使用 `DateSerial` 函数将某个固定值赋给一个日期变量：

```
Dim MyVar As Date

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

函数参数的顺序必须为：年、月、日。该函数确保变量被实际赋值为正确的值，而不管特定于国家/地区的设定。

`TimeSerial` 函数按照 `DateSerial` 函数格式化日期的方式格式化时间细节：

```
Dim MyVar As Date

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

它们的参数应按照以下顺序指定：小时、分钟、秒。

提取日期和时间细节

以下函数执行的是 `DateSerial` 和 `TimeSerial` 函数的逆运算：

- **`Day(MyDate)`** - 返回的数值指明 `MyDate` 是当月中的第几天
- **`Month(MyDate)`** - 返回的数值指明 `MyDate` 是哪一月
- **`Year(MyDate)`** - 返回的数值指明 `MyDate` 是哪一年
- **`Weekday(MyDate)`** - 返回的数值指明 `MyDate` 是星期几
- **`Hour(MyTime)`** - 返回的数值指明 `MyTime` 对应的小时数
- **`Minute(MyTime)`** - 返回的数值指明 `MyTime` 对应的分钟数
- **`Second(MyTime)`** - 返回的数值指明 `MyTime` 对应的秒数

这些函数从一个指定的**日期变**量中提取日期和/ 或时间部分。示例

```
Dim MyDate As Date

' ... 初始化 MyDate

If Year(MyDate) = 2003 Then

      ' ... 指定的日期的年份为 2003
End If
```

检查 `MyDate` 中存盘的日期的年份是否为 2003。同样，示例

```
Dim MyTime As Date

' ... 初始化 MyTime

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

      ' ... 指定的日期的年份为 2003
End If
```

检查 MyTime 的小时数是否介于 12 和 14 之间。

Weekday 函数返回传送的日期是星期几：

```
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
```

备注：星期日被视为一个星期的第一天。

检索系统日期和时间

在 StarSuite Basic 中，可以使用以下函数来检索系统时间和系统日期：

- **Date** - 返回当前日期
- **Time** - 返回当前时间
- **Now** - 返回当前的时间点 (日期和时间的组合值)

文件和目录

使用文件是一个应用程序的基本任务之一。StarSuite API 为您提供了一整套对象，用于建立、打开和更改 Office 文档。第 4 章中详细介绍了这些对象。尽管如此，在某些情况下，需要直接访问文件系统、搜寻目录或编辑文本文件。StarSuite Basic 运行时库为这些任务提供了多个基本函数。

StarSuite 6.x 中不再提供某些 DOS 特有的文件和目录函数，或者其功能非常有限。例如，不支持 ChDir、ChDrive 和 CurDir 函数。
需要文件属性作为参数的函数中不再使用某些 DOS 特有的属性 (例如，将隐藏文件与系统文件区分开来)。为确保 StarSuite 的平台独立性达到尽可能高的级别，此更改是必需的。

管理文件

搜寻目录

StarSuite Basic 中的 Dir 函数负责从目录中搜寻文件和子目录。初次请求时，必须将一个字串赋值给 Dir，作为它的第一个参数，该子串含有要搜寻的目录的路径。Dir 的第二个参数指定要搜寻的文件或目录。StarSuite Basic 返回搜寻到的第一个目录条目的名称。为了检索下一个条目，应在不含参数的情况下请求 Dir 函数。如果 Dir 函数没有搜寻到条目，就返回一个空字串。

以下示例显示了如何使用 Dir 函数来请求位于某个目录的所有文件。该过程将各个文件名称存盘在 AllFiles 变量中，然后将此变量显示在一个消息框中。

```
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 作为 Dir 函数中的第二个参数可以确保 Dir 仅返回文件名称而忽略目录。此处可以指定以下参数：

- 0：返回常规文件
- 16：返回子目录

以下示例几乎与上面的示例相同，只是 Dir 函数将值 16 作为一个参数传送，该函数返回文件夹的子目录，而不是文件名称。

```
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
```

与 VBA 不同，当在 StarSuite Basic 中请求时，使用参数 16 的 Dir 函数仅返回某个文件夹的子目录。(在 VBA 中，该函数还返回标准文件的名称，这样，需要进一步检查才能只检索目录。)

StarSuite Basic 中不存在 VBA 中提供的用于在目录中专门搜寻具有 隐藏、系统文件、存档和卷名属性的选项，因为相应的文件系统函数不能在所有的操作系统上可用。

在 VBA 和 StarSuite Basic 中，Dir 中列出的路径规范都可以使用 * 和 ? 占位符。但是，在 StarSuite Basic 中，* 占位符只能是文件名和/或文件扩展名的最后一个字符，而在 VBA 中却并非如此。

建立和删除目录

StarSuite Basic 提供了用于建立目录的 MkDir 函数。

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

此函数用于建立目录和子目录。如果需要，还建立一个等级式结构中需要的所有目录。例如，如果仅存在 C:\SubDir1 目录，则调用

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

将同时建立 C:\SubDir1\SubDir2 目录和 C:\SubDir1\SubDir2\SubDir3 目录。

RmDir 函数用于删除目录。

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

如果目录中含有子目录或文件，它们也会被删除。 因此，使用 RmDir 时应谨慎。

在 VBA 中，MkDir 和 RmDir 函数仅与当前目录相关。而在 StarSuite Basic 中，MkDir 和 RmDir 可用于建立或删除多级目录。

在 VBA 中，如果目录中含有文件，则 RmDir 会生成一个错误报告。而在 StarSuite Basic 中，目录及其所有文件都被删除。

复制、重命名和删除文件以及检查文件是否存在

调用

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

将以名称 `Destination` 建立 `Source` 文件的复制件。

借助函数

```
Name OldName As NewName
```

可以将 `OldName` 文件重命名为 `NewName`。`As` 关键字语法以及不使用逗号这一事实可追溯到 Basic 语言的起源。

调用

```
Kill(Filename)
```

将删除 `Filename` 文件。如果要删除目录 (包括其文件)，请使用 `RmDir` 函数。

`FileExists` 函数可用于检查某个文件是否存在：

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

读取和更改文件属性

使用文件时，能够确定文件属性、上次更改文件的时间以及文件长度有时非常重要。

调用

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

将返回一个文件的某些属性。返回值是作为一个位掩码提供的，其中可以包含以下值：

- 1 : 只读文件
- 16 : 目录名称

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
```

确定 `test.txt` 文件的位掩码并检查文件是否为只读以及是否是一个目录。如果都不是，就将 “normal” 字串赋给 `FileDescription`。

StarSuite Basic 不支持 VBA 中用于查询隐藏、系统文件、存档和卷名文件属性的标志，因为它们是 Windows 特有的标志，而在其他操作系统中不可用或者只是部分可用。

`SetAttr` 函数允许更改某个文件的属性。因此调用

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

可用于为一个文件提供只读状态。可以通过以下调用来删除现有的只读状态：

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

`FileDateTime` 函数提供了上次更改文件时的日期和时间。此处，按照操作系统上使用的特定于国家/ 地区的设定来格式化日期。

```
FileDateTime("test.txt")      ' 提供上次更改文件时的日期和时间。
```

`FileLen` 函数以字节为单位确定一个文件的长度 (返回值为长整数)。

```
FileLen("test.txt")           ' 以字节为单位提供文件长度
```

编写和读取文本文件

StarSuite Basic 提供了一整套读取和编写文件的方法。以下说明与使用文本文件 (不是文本文档) 有关。

编写文本文件

访问一个文本文件之前，必须先打开该文本文件。为此，需要一个自由的文件句柄，它清楚地标识文件，以便随后访问文件。

`FreeFile` 函数用于建立一个自由的文件句柄。句柄用作 `Open` 指令的一个参数，该指令可以打开文件。要打开一个文件以便将其指定为文本文件，`Open` 调用为：

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

`Filename` 是一个含有文件名称的字串。 `FileNo` 是通过 `FreeFile` 函数建立的句柄。

打开文件后，就可以逐行描述 `Print` 指令：

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

`FileNo` 在此还代表文件句柄。第二个参数指定的文字将存盘为文本文件中的一行。

一旦完成编写过程，就必须使用一个 `Close` 调用来关闭文件：

```
Close #FileNo
```

此处同样应该指定文件句柄。

以下示例显示如何打开、描述和关闭文本文件：

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

Filename = "c:\data.txt"                              ' 定义文件名称
FileNo = Freefile                                     ' 建立自由的文件句柄

Open Filename For Output As #FileNo                   ' 打开文件 (编写模式)
Print #FileNo, "This is a line of text"               ' 存盘行
Print #FileNo, "This is another line of text"         ' 存盘行
Close #FileNo                                         ' 关闭文件
```

读取文本文件

文本文件的读取方式与编写方式相同。用于打开文件的 `Open` 指令含有取代 `For Output` 表达式的 `For Input` 表达式，并且不是使用 `Print` 命令编写数据，而应使用 `Line Input` 指令读取数据。

最后，当调用一个文本文件时，指令

```
eof(FileNo)
```

用于检查是否到达文件末尾。

以下示例显示如何读取文本文件：

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

' 定义文件名称
Filename = "c:\data.txt"

' 建立自由的文件句柄
FileNo = Freefile

' 打开文件 (读取模式)
Open Filename For Input As FileNo

' 检查是否到达文件末尾
Do While not eof(FileNo)

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

Loop

' 关闭文件
Close #FileNo

Msgbox Msg
```

通过一个 `Do While` 循环来检索各行，检索到的各行存盘在 `Msg` 变量中，最后显示在一个消息框中。

消息框和输入框

StarOffice Basic 提供了 `MsgBox` 和 `InputBox` 函数以用于简单的使用者通信。

输出消息

`MsgBox` 显示一个简单的信息框，该框可以含有一个或多个按钮。在其最简单的变体

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

中，`MsgBox` 仅含有文字和一个“确定”按钮。

可以使用一个参数来更改信息框的外观。该参数提供用于添加附加按钮、定义预先指定的按钮以及添加信息符号的选项。用于选择按钮的值为：

- 0 -“确定”按钮
- 1 -“确定”和“取消”按钮
- 2 -“取消”和“重试”按钮
- 3 -“是”、“否”和“取消”按钮
- 4 -“是”和“否”按钮
- 5 -“重试”和“取消”按钮

要将一个按钮设定为默认按钮，请用以下值之一加上按钮选择列单的参数值。例如，要建立“是”、“否”和“取消”按钮 (值 3) 并使“取消”按钮为默认按钮 (值 512)，参数值就是 3 + 512 = 515。

- 0 - 第一个按钮为默认值
- 256 - 第二个按钮为默认值
- 512 - 第三个按钮为默认值

最后，可以使用以下信息符号，还可通过添加相关参数值来显示这些信息符号：

- 16 - 停止符号
- 32 - 问号
- 48 - 感叹号
- 64 - 提示图标

调用

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

显示一个含有“是”和“否”按钮 (值 4) 的信息框，其中，第二个按钮 (“否”) 被设定为默认值 (值 256)，而且该信息框还含有一个问号 (值 32)，因此参数值为 4+256+32=292。

如果一个信息框含有多个按钮，则应查询返回值，以确定按了哪个按钮。在这种情况下可以使用以下返回值：

- 1 -“确定”
- 2 -“取消”
- 4 -“重试”
- 5 -“忽略”
- 6 -“是”
- 7 -“否”

在前面的示例中，可以按以下方式检查返回值：

```
If MsgBox ("Do you want to continue?",  292) = 6 Then
      ' 按了“是”按钮
Else
      ' 按了“否”按钮
End IF
```

除了信息文字和用于对信息框进行排序的参数之外，MsgBox 还允许有第三个参数，该参数定义框标题的文字：

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

如果不指定框标题，默认值是“soffice”。

用于查询简单字串的输入框

InputBox 函数查询使用者输入的简单字串。因此，它是配置对话框的另一种简单方法。InputBox 接受三个标准参数：

- 信息文字，
- 框标题，
- 可在输入区域中加入的默认值。

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

InputBox 将使用者输入的字串作为返回值。

其他函数

Beep

Beep 函数使系统播放一种声音，该声音可用于警告使用者执行了错误的操作。Beep 不含任何参数：

```
Beep   ' 建立一种信息音调
```

Shell

使用 `Shell` 函数可以启动外部程序。

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

`Pathname` 定义要执行的程序的路径。`Windowstyle` 定义用于启动程序的视窗。可能的值如下：

- 0 - 程序接收到焦点，并在一个隐藏视窗中启动。
- 1 - 程序接收到焦点，并在一个正常大小的视窗中启动。
- 2 - 程序接收到焦点，并在一个最小化视窗中启动。
- 3 - 程序接收到焦点，并在一个最大化视窗中启动。
- 4 - 程序在一个正常大小的视窗中启动，但没有接收到焦点。
- 6 - 程序在一个最小化视窗中启动，焦点保留在当前视窗中。
- 10 - 程序以全屏模式启动。

第三个参数 `Param` 允许将命令行参数传送到要启动的程序。

Wait

`Wait` 函数将程序执行终止一段时间。等待时间以毫秒为单位指定。命令

```
Wait 2000
```

指定一个 2 秒 (2000 毫秒) 的中断。

Environ

`Environ` 函数返回操作系统的环境变量。此处可以存盘各种类型的数据，具体情况取决于系统和配置。调用

```
Dim TempDir

TempDir=Environ ("TEMP")
```

确定操作系统临时目录的环境变量。

第 **4** 章

StarSuite API 简介

StarSuite API 是访问 StarSuite 的通用编程接口。使用 StarSuite API 可以建立、打开、更改和打印 StarSuite 文档。它通过个人宏扩展了 StarSuite 功能范围并允许设计个人对话框。

StarSuite API 不仅可以与 StarSuite Basic 一起使用，还可与其他编程语言 (例如 Java 和 C++) 一起使用。这是因为一种名为通用网络对象 (UNO) 的技术，该技术为各种编程语言提供了一个接口。

本章主要介绍如何借助 UNO 在 StarSuite Basic 中使用 StarSuite，并从 StarSuite Basic 程序员的立场对 UNO 中的主要概念进行说明。本章之后的章节中详细介绍了如何使用 StarSuite API 的各部分。

通用网络对象 (UNO)

StarSuite 提供了通用网络对象 (UNO) 形式的编程接口。这是一个面向对象的编程接口，StarSuite 又将其细分为各种对象，从而确保可以使用程序来控制对 Office 程序包的访问。

由于 StarSuite Basic 是一种过程编程语言，因此必须要在其中加入几种语言结构才能使用 UNO。

为了在 StarSuite Basic 中使用通用网络对象，需要为相关的对象进行变量声明。进行声明时要使用 `Dim` 指令 (请参阅第 2 章)。应使用**对象**类型指定来声明对象变量：

```
Dim Obj As Object
```

该调用声明了一个名为 `Obj` 的对象变量。

接着需要初始化建立的对象变量，这样才可以使用该变量。这可以使用 `createUnoService` 函数来完成：

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

此调用将新建立的对象的一个引用赋给 `Obj` 变量。`com.sun.star.frame.Desktop` 类似于一个对象类型，但在 UNO 术语中，它被称作“服务”而不是类型。按照 UNO 思想，`Obj` 被描述为支持 `com.sun.star.frame.Desktop` **服务**的对象的**一个引用**。因此，StarSuite Basic 中使用的“服务”术语，与其他编程语言中使用的“类型”或“类”相对应。

但存在一个主要区别：通用网络对象可同时支持多种服务。而有些 UNO 服务又支持其他服务，这样，通过一个对象就可以提供一系列的服务。例如，上述基于 `com.sun.star.frame.Desktop` 服务的对象还可以包括用于装入文档和结束程序的其他服务。

在 VBA 中，对象的结构是通过其所属的类来定义的，而在 StarSuite Basic 中，对象的结构是通过其所支持的服务来定义的。VBA 对象总是被指定到一个特定的类，而

StarSuite Basic 对象可以支持多种服务。

属性和方法

StarSuite Basic 中的对象提供了一系列的属性和方法，可以通过该对象来调用这些属性和方法。

属性

此处的属性与对象具有的属性类似，例如 `Document` 对象的 `Filename` 和 `Title`。

属性可以通过简单的赋值来设定：

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

属性象常规变量一样，其类型定义了可以记录的值。
上面的 `Filename` 和 `Title` 属性是字串类型。

真实属性和模拟属性

在 StarSuite Basic 中，对象的大多数属性都按照服务的 UNO 描述来定义。除了这些“真实”属性之外，在 StarSuite Basic 中，还有一些属性包括 UNO 级的两种方法。其中的一种方法用于查询属性的值，另一种用于设定属性的值 (`get` 和 `set` 方法)。该属性实际上是由两种方法模拟而成的。例如，UNO 中的字符对象提供了 `getPosition` 和 `setPosition` 两种方法，通过这些方法可以调用和更改关联的关键点。StarSuite Basic 程序员可以通过 `Position` 属性来访问这些值。除了此属性以外，原来的方法也可以使用 (此示例中为 `getPosition` 和 `setPosition`)。

方法

可以将方法理解为与对象直接相关的函数，并可以通过该函数调用对象。例如，上面的 Document 对象可以提供 Save 方法，该方法的调用方式如下：

```
Document.Save()
```

方法就象函数一样，可以含有参数和返回值。这种方法调用的语法面向的是典型的函数。调用

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

在请求 Save 方法时，还为文档对象指定 True 参数。
方法完成后，Save 将返回值存盘在 Ok 变量中。

模块、服务和接口

StarSuite 提供了上百种服务。为对这些服务做个简单的概述，将其合并成了模块。对于 StarSuite Basic 程序员来说，这些模块没有提供重要功能。只有在指定服务名称时，模块名称才是重要的，这是因为指定的名称中也必须要列出模块名称。完整的服务名称包括 com.sun.star 表达式，该表达式指定它是 StarSuite 服务；然后是模块名称，例如 frame；最后是实际的服务名称，例如 Desktop。上述示例中的完整名称如下所示：

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

除了模块和服务两个术语之外，UNO 还引入了术语“接口”。 Java 程序员可能比较熟悉这个术语，但 Basic 中不使用它。

一个接口可以组合多种方法。从最严格的字面意义上来说，UNO 中的服务不支持方法，但支持接口，而接口又提供了不同的方法。换句话说，方法以接口的形式被指定 (作为组合) 到服务中。Java 或 C++ 程序员可能会对这个细节特别感兴趣，因为在这两种语言中请求方法时需要使用接口。但在 StarSuite Basic 中，方法与接口无关，而是通过相关的对象来直接调用方法。

但是，按照 API 的思想，将方法指定到便于操作的各种接口非常有用，因为不同的服务中都要使用许多接口。如果您比较熟悉某个接口，这些知识也可以在服务之间借鉴。

有些主要接口的使用非常频繁，本章结尾处将再次显示这些由不同服务所触发的接口。

使用 UNO 时所需的工具

现在的问题是我们要保留 UNO 术语中的哪些对象或服务，支持哪些属性、方法和接口，及如何确定它们。除了本书，可以从以下来源获得有关对象的更多信息：supportsService 方法、调试方法、《StarSuite 开发者指南》和 API 参考。

supportsService 方法

许多 UNO 对象都支持 supportsService 方法，可以使用该方法来确定一个对象是否支持某种特殊服务。例如，调用

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

确定 TextElement 对象是否支持 com.sun.star.text.Paragraph 服务。

调试属性

StarSuite Basic 中的每个 UNO 对象都**知道**已经含有哪些属性、方法和接口，并提供了以列单的形式返回这些信息的属性。相应的属性有：

DBG_properties - 返回一个含有某个对象所有属性的字串

DBG_methods - 返回一个含有某个对象所有方法的字串

DBG_supportetInterfaces - 返回一个含有支持某个对象的所有接口的字串。

以下程序代码显示了如何在实际应用程序中使用 DBG_properties 和 DBG_methods。首先建立 com.sun.star.frame.Desktop 服务，然后在消息框中显示支持的属性和方法。

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

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

在使用 DBG_properties **时，请注意**该函数将返回某个特定服务在理论上可以支持的所有属性，但并不确保相关的对象也可以使用这些属性。因此，在调用属性之前，必须使用 IsEmpty 函数来检查对象是否真的可用。

API 参考

如果需要有关可用的服务及其接口、方法和属性方面的更多信息，请参阅 StarSuite API 参考。此参考可以从 www.openoffice.org 上的以下地址获得：

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

几个重要接口的概述

在 StarSuite API 的许多部分中都可以找到一些 StarSuite 接口。它们为抽象的任务定义了一系列方法，适用于各种问题。这里，将概述一些最常用的接口。

有关对象的起源将在本书的后面章节中进行说明。在此，仅介绍对象的一些抽象方面，StarSuite API 为这些对象提供了一些重要接口。

建立上下文相关的对象

StarSuite API 提供了两种建立对象的方法。其中的一种方法是本章开始处提到的 `createUnoService` 函数。`createUnoService` 用于建立通用的对象。这类对象和服务也称为上下文不相关服务。

除了上下文不相关服务以外，还存在上下文相关服务，此类服务的对象只有与其他对象一起使用。例如，工作表文档中的绘图对象只在这一种文档中存在。

com.sun.star.lang.XMultiServiceFactory 接口

上下文相关对象通常是通过该对象依赖的某种对象方法而建立的。在 `XMultiServiceFactory` 接口中定义的 `createInstance` 方法，专门用于文档对象。

例如，可以使用工作表对象来建立前面提到的绘图对象，方法如下：

```
Dim RectangleShape As Object

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

文本文档中的段落样式也可以用同样的方式建立，方法如下：

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

命名访问下级对象

XNameAccess 和 XNameContainer 接口用在含有下级对象的对象中，用自然语言名称就可以找到这些下级对象。

XNamedAccess 允许访问各个对象，XNameContainer 可以插入、更改和删除元素。

com.sun.star.container.XNameAccess 接口

工作表文档中的工作表对象提供了 XNameAccess 的使用示例。该示例组合了工作表文档中的所有页。可以使用 XNameAccess 中的 getByName 方法来访问各个页，如下所示：

```
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称的摘要。因此，它返回一个含有这些名称的数据字段。以下示例显示了如何在一个循环中确定并显示某个工作表中的所有元素名称：

```
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 接口中的 hasByName 方法揭示基本对象中是否存在具有特定名称的下级对象。因此，以下示例显示一条消息，告知使用者 Spreadsheet 对象中是否含有一个名为 Sheet1 的页。

```
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 接口

XNameContainer 接口用于插入、删除和更改基本对象的下级元素。负责实现这些功能的函数为：insertByName、removeByName 和 replaceByName。

以下是此接口的一个实际示例。该示例调用一个文本文档，文档中含有一个 StyleFamilies 对象，使用此对象又可以获得文档的段落样式 (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` 行将在 `ParagraphStyles` 对象中插入 `NewStyle` 样式，并以 “NewStyle”命名该样式。`replaceByName` 行将 `ChangingStyle` 背后的对象更改为 `NewStyle`。最后，`removeByName` 调用将 `OldStyle` 背后的对象从 `ParagraphStyles` 中删除。

基于索引访问下级对象

`XIndexAccess` 和 `XIndexContainer` 接口用于含有下级对象并可以使用索引找到这些下级对象的对象。

`XIndexAccess` 提供访问各个对象的方法。
`XIndexContainer` 提供插入和删除元素的方法。

com.sun.star.container.XIndexAccess 接口

`XIndexAccess` 提供了 `getByIndex` 和 `getCount` 两种调用下级对象的方法。`getByIndex` 通过一个特定的索引来提供对象。`getCount` 返回可用对象的数目。

```
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)
      ' 编辑工作表
Next I
```

该示例显示了一个遍历所有工作表元素的循环，并用 `Sheet` 对象变量存盘每个元素的一个引用。使用索引时，请注意 `getCount` 将返回元素数目，但 `getByIndex` 中的元素是从 `0` 开始编号，因此，循环中的计数变量是从 `0` 到 `getCount()-1`。

com.sun.star.container.XIndexContainer 接口

`XIndexContainer` 接口提供了 `insertByIndex` 和 `removeByIndex` 函数。参数的结构与 `XNameContainer` 中的相应函数相同。

交互访问下级对象

在有些情况下，某个对象可能含有一个下级对象列单，但无法通过名称或索引找到这些下级对象。这时，使用 XEnumeration 和 XenumerationAccess 接口比较适合。它们提供了一种机制来遍历某个对象的所有下级元素，而无需使用直接寻址。

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

基本对象必须提供 XEnumerationAccess 接口，该接口仅含有 createEnumeration 方法。此方法返回一个辅助对象，而这个对象又提供了带有 hasMoreElements 和 nextElement 方法的 XEnumeration 接口。这样，就可以访问下级对象。

以下示例遍历一个文本文档的所有段落：

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

该示例首先建立了一个 ParagraphEnumeration 辅助对象。 然后在一个循环中利用此对象逐步返回文本文档的各个段落。 一旦 hasMoreElements 方法返回 False 值 (表示已经到达文本文档的末尾)，循环就会终止。

第 5 章

使用 StarSuite 文档

StarSuite API 是一种结构化的 API，因而其绝大多数功能都可普遍地应用在不同的任务中。这些功能包括各种接口和服务，分别用于建立、打开、保存、转换和打印文档以及样式管理。由于这些功能适用于各类文档，因此本章首先介绍这些功能。

使用文档时，以下两项服务为最常使用的服务：

- `com.sun.star.frame.Desktop` 服务，此服务与 StarSuite 的核心服务类似。它为 StarSuite 框对象提供了各种功能，并且对内部的所有文档视窗都进行了分类。此服务也可用于建立、打开和输入文档。
- 单个文档对象的基本功能由 `com.sun.star.document.OfficeDocument` 服务提供。此服务提供了保存、输出和打印文档的方法。

启动 StarSuite 时，`com.sun.star.frame.Desktop` 服务将自动打开。为此，StarSuite 建立了一个可以通过全局名称 `StarDesktop` 访问的对象。

最重要的 `StarDesktop` 接口为 `com.sun.star.frame.XComponentLoader`。它通常包含 `loadComponentFromURL` 方法，此方法负责建立、输入和打开文档。

`StarDesktop` 对象的名称起源可回溯至 StarOffice 5，在此版本中，所有文档视窗都嵌入在一个名为 StarDesktop 的通用应用程序中。在 StarSuite 目前的版本中，可视化的 StarDesktop 已不再使用。然而，StarSuite 的框对象却保留了这个名称 StarDesktop，因为此名称可清楚地表明它是整个应用程序的一个基本对象。

`StarDesktop` 对象替代了以前作为根对象使用的 StarOffice 5 `Application` 对象。但与旧的 `Application` 对象不同的是，它主要负责打开新文档。旧的 `Application` 对象内用于控制 StarSuite 屏幕描述的函数 (例如：`FullScreen`、`FunctionBarVisible`、`Height`、`Width`、`Top`、`Visible`) 已不再使用。

在 Word 中，使用中的文档通过 `Application.ActiveDocument` 访问，在 Excel 中，通过 `Application.ActiveWorkbook` **访问，然而在** StarOffice 中，则由 `StarDesktop` 负责此项任务。在 StarSuite 6 中，使用中的文档对象通过 `StarDesktop.CurrentComponent` 属性进行访问。

关于 StarSuite 中文档的基本信息

使用 StarSuite 文档时，应学会处理 StarSuite 中某些文档管理方面的基本问题，这一点非常有用。这些问题包括 StarSuite 文档的文件名构成方式以及这些文件的保存格式。

用 URL 表示的文件名

由于所设计的 StarSuite 是一种与平台无关的应用程序，因此它使用 URL 表示法 (此表示法不依赖于任意操作系统)，此表示法在 “Internet Standard RFC 1738”中的文件名部分进行了定义。使用此系统的标准文件名都以前缀

```
file:///
```

开头，后跟本地路径。如果文件名中含有子目录，则这些子目录使用**单个的正斜杠**进行分隔，而不是 Windows 通常使用的反斜杠。以下路径引用的是驱动器 `C:` 上 `doc` 目录中的 `test.sxw` 文件。

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

为了将本地文件名转换为 URL，StarSuite 提供了 `ConvertToUrl` 函数。
为了将 URL 转换为本地文件名，StarSuite 提供了 `ConvertFromUrl` 函数：

```
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
```

此示例先将本地文件名转换为一个 URL，并显示在信息框中。然后再将 URL 转换为一个文件名，并且也显示出来。

示例所依据的 “Internet Standard RFC 1738”允许使用 `0-9`、`a-z` 以及 `A-Z` 等字符。其他所有字符都将作为换码编码插入 URL 中。为此，这些字符需转换成 ISO 8859-1 (ISO-Latin) 字符集中相应的十六进制值，并以百分号为前缀。例如，基于这个原因，本地文件名中的空格会变为 URL 中的 `%20`。

XML 文件格式

从 6.0 版起，StarSuite 提供了一种基于 XML 的文件格式。通过使用 XML，用户可以选择在其他程序中打开和编辑这些文件。

文件压缩

由于 XML 是一种基于标准文本的格式，所生成的文件通常都很大。因此，StarSuite 将压缩这些文件，并将其保存为 ZIP 文件。

通过使用 `storeAsURL` 方法选项，用户可以直接保存初始 XML 文件。请参阅第 81 页中的 storeAsURL 方法选项。

建立、打开和输入文档

打开、输入和建立文档将使用方法

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

loadComponentFromURL 中的第一个参数指定了关联文件的 URL。

至于第二个参数，loadComponentFromURL 需要一个 StarSuite 为进行管理而在其内部建立的视窗中的框对象名称。此处通常指定为预定义的名称 _blank，这样就可确保 StarSuite 建立一个新视窗。此外，也可以指定为 _hidden，这样就可确保加载相应文档，但不显示。

只需使用这两个参数，用户就可以打开 StarSuite 文档，因为最后两个参数可赋值为占位符 (虚设的值)：

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

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

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

以上调用先打开 text.sxw 文件，然后在新视窗内显示此文件。

可以在 StarSuite Basic 中用此方法打开无数个文档，然后再使用返回的文档对象进行编辑。

StarDesktop.loadComponentFromURL 取代了旧的 StarSuite API 中的 Documents.Add 和 Documents.Open 方法。

替换文档视窗的内容

Frame 参数中名为 _blank 和 _hidden 的值确保 StarSuite 为来自 loadComponentFromURL 的每个调用建立一个新的视窗。在某些情况下，替换现有视窗的内容非常有用。在这种情况下，视窗的框对象应含有一个明确的名称。请注意，此名称不得以下划线开头。而且，必须设定 SearchFlags 参数，以便建立相应的框架 (如果此框架尚未存在)。SearchFlags 的相应常数为：

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

以下示例显示了如何借助框参数和 SearchFlags 替换打开的视窗内容：

```
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)
```

此示例首先在具有框名为 MyFrame 的新视窗中打开 test.sxw 文件。一旦在信息框中进行了确认，此示例就使用 test2.sxw 文件替换了视窗的内容。

loadComponentFromURL 方法选项

loadComponentFromURL 函数的第四个参数为 PropertyValue 数据字段，此字段用于为 StarSuite 提供各种打开和建立文档的选项。此数据字段必须为每个选项提供一个 PropertyValue 结构。在此结构中，选项名称保存为字串或关联的值。

loadComponentFromURL 支持以下选项：

- **AsTemplate (布尔)** - 如果为 True，就从给定的 URL 中加载一个未命名的新文档。如果为 False，就加载要编辑的样式文件。
- **CharacterSet (字串)** - 定义文档基于哪个字符集。
- **FilterName (字串)** - 为 loadComponentFromURL 函数指定一个特殊筛选。可用筛选名在 \share\config\registry\instance\org\openoffice\office\TypeDetection.xml 文件中定义。
- **FilterOptions (字串)** - 定义筛选的其他选项。
- **JumpMark (字串)** - 一旦文档打开后，就跳转到 JumpMark 定义的位置上。
- **Password (字串)** - 传送受保护文件的密码。
- **ReadOnly (布尔)** - 加载只读文档。

以下示例显示如何使用 FilterName 选项打开 StarSuite Calc 中的以逗点分隔的文本文件。

```
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 (StarSuite Calc)"

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

FileProperties 数据字段中只有一个值，因为它只记录了一个选项。
Filtername 属性定义了 StarSuite 是否使用 StarSuite Calc 文本筛选打开文件。

建立新文档

如果 URL 中指定的文档为一种样式，StarSuite 将自动建立一个新文档。

此外，如果只需要一个未经任何改编的空文档，就可以指定一个 private:factory URL：

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

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

以上调用建立了一个空的 StarSuite Writer 文档。

文档对象

前一节中介绍的 loadComponentFromURL 函数会返回一个文档对象。这一文档对象支持 com.sun.star.document.OfficeDocument 服务，此服务按顺序提供两个中心接口：

- com.sun.star.frame.XStorable 接口，此接口负责保存文档，以及
- com.sun.star.view.XPrintable 接口，此接口含有用于打印文档的方法。

变换至 StarSuite 6 后，请注意文档对象的功能范围多半都未发生变化。例如，文档对象仍然提供保存和打印文档的方法。但是，这些方法的名称和参数却发生了变化。

保存和输出文档

StarSuite 文档是直接通过文档对象保存的。为此，com.sun.star.frame.XStorable 接口提供了 store 方法：

```
Doc.store()
```

只要未给文档指定一个内存空间，此调用就会生效。新文档则不是如此。对于新文档，需使用 storeAsURL 方法。此方法也是在 com.sun.star.frame.XStorable 中定义的，并可用于定义文档的位置：

```
Dim URL As String
```

```
Dim Dummy()

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

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

除以上这些方法外，`com.sun.star.frame.XStorable` 也提供一些帮助方法，这些方法在保存文档时非常有用。这些方法为：

- **`hasLocation()`** - 指定是否已经为文档指定了一个 URL。
- **`isReadonly()`** - 指定文档是否具有只读保护。
- **`isModified()`** - 指定文档自上次保存以来是否经过修改。

用于保存文档的代码可以通过这些选项进行扩充，以便仅当确实修改了对象之后才保存文档，以及仅当确实需要时才查询文件名：

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

此示例首先检查相关文档在上次保存以来是否经过修改。只有进行了修改，才会继续保存过程。如果文档中已经指定了一个 URL，而且不是只读文档，则此文档会保存在现有的 URL 下。如果文档没有 URL 或以只读状态打开，则会保存在新的 URL 下。

storeAsURL 方法选项

像使用 `loadComponentFromURL` 方法时一样，使用 `storeAsURL` 方法时也可以将部分选项指定为 `PropertyValue` 数据字段形式。这将确定保存文档时 StarSuite 所要使用的过程。`storeAsURL` 提供了以下选项：

- **`CharacterSet (字串)`** - 定义文档基于哪个字符集。
- **`FilterName (字串)`** - 为 `loadComponentFromURL` 函数指定一个特殊筛选。可用筛选名在 `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml` 文件中定义。
- **`FilterOptions (字串)`** - 定义筛选的其他选项。
- **`Overwrite (布尔)`** - 无需查询即可覆盖一个已经存在的文件。
- **`Password (字串)`** - 传送一个受保护文件的密码。
- **`Unpacked (布尔)`** - 在子目录中保存文档 (未压缩)。

以下示例显示如何将 `Overwrite` 选项与 `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

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

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

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

在此示例中，如果存在同名文件，则会将 `Doc` 以指定的文件名保存。

打印文档

与保存文档相似，打印文档也是直接通过文档对象进行的。为此，`com.sun.star.view.Xprintable` 接口提供了 `Print` 方法。最简单的 `print` 调用形式为：

```
Dim Dummy()

Doc.print(Dummy())
```

与使用 `loadComponentFromURL` 方法时一样，`Dummy` 参数是一个 `PropertyValue` 数据字段，通过此字段，StarSuite 可以指定多个打印选项。

Print 方法的选项

print 方法需要一个 PropertyValue 数据字段作为其参数，此字段反映了 StarSuite [打印] 对话框的设定：

- **CopyCount (整数)** - 指定要打印的份数。
- **FileName (字串)** - 打印指定文件中的文档。
- **Collate (布尔)** - 建议打印机逐份打印。
- **Sort (布尔)** - 打印多页时对页面进行排序 (CopyCount > 1)。
- **Pages (字串)** - 含有要打印的页面列单 (在 [打印] 对话框中指定的语法)。

以下示例显示如何使用 Pages 选项打印一个文档的多个页面：

```
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())
```

打印机选择和设定

com.sun.star.view.XPrintable 接口提供了 Printer 属性，用于选择打印机。此属性会收到一个具有以下设定的 PropertyValue 数据字段：

- **Name (字串)** - 指定打印机的名称。
- **PaperOrientation (枚举)** - 指定纸张的方向 (com.sun.star.view.PaperOrientation.PORTRAIT 值用于纵向格式，com.sun.star.view.PaperOrientation.LANDSCAPE 用于横向格式)。
- **PaperFormat (枚举)** - 指定纸张的格式 (例如，com.sun.star.view.PaperFormat.A4 指定 DIN A4 或 com.sun.star.view.PaperFormat.Letter 指定 US 字母)。
- **PaperSize (大小)** - 指定纸张的大小，以百分之一毫米为单位。

以下示例显示如何借助 Printer 属性更换打印机以及设定纸张的大小。

```
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      ' 等于 20 cm
PaperSize.Height = 20000     ' 等于 20 cm

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

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

Doc.Printer = PrinterProperties()
```

此示例定义了一个名为 PaperSize 的对象，其类型为 com.sun.star.awt.Size。这是指定纸张大小时需要的对象。另外，示例还为两个 PropertyValue 条目建立了一个数据字段，名为 PrinterProperties。然后使用要设定并赋给 Printer 属性的值初始化此数据字段。以 UNO 的观点来看，printer 不是一个真实属性，而是一个**模拟**属性。

样式

样式是含有格式属性的命名列单。它们穿行于 StarSuite 的所有应用程序之间，帮助有效地简化格式。如果用户更改了样式的其中一个属性，则 StarSuite 会根据此属性自动调整所有文档区域。例如，基于这个原因，用户可以通过对文档进行一次集中修改，而更改所有一级标题的字体类型。根据相关的文档类型，StarSuite 能够识别不同类型的几组样式。

StarSuite Writer 支持

- 字符样式
- 段落样式
- 框样式
- 页面样式
- 编号样式

StarSuite Calc 支持

- 单元格样式
- 页面样式

StarSuite Impress 支持

- 字符元素样式
- 演示文稿样式

在 StarSuite 术语中，各种不同类型的样式统称为 `StyleFamilies`，此名称与这些样式所基于的 `com.sun.star.style.StyleFamily` 服务一致。
`StyleFamilies` 是通过文档对象访问的：

```
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")
```

此示例使用工作表文档的 `StyleFamilies` 属性建立了一个含有所有可用单元格样式的列单。

每个样式都可通过索引直接访问：

```
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
```

与前一个示例相比，此示例添加了一个循环，该循环会在信息框内逐个显示所有单元格样式名称。

关于各种格式选项的细节

每种样式类型都提供一套完整的格式属性。此处概要地列出了最重要的格式属性以及介绍这些属性的具体位置：

- 字符属性，第 6 章文本文档
  `com.sun.star.style.CharacterProperties` 服务
- 段落属性，第 6 章文本文档
  `com.sun.star.text.Paragraph` 服务
- 单元格属性，第 7 章工作表文档
  `com.sun.star.table.CellProperties` 服务
- 页面属性，第 7 章工作表文档
  `com.sun.star.style.PageStyle` 服务
- 字符元素属性，第 7 章工作表文档
  各种服务

这些格式属性决不仅限于在解释它们时所提到的应用程序中使用，而是可以广泛地使用。例如，基于此原因，第 7 章中所介绍的大多数页面属性不仅可以用于 StarSuite Calc，而且还可以用于 StarSuite Writer。

如果需要有关使用样式的更多信息，请参阅第 6 章文本文档中的字符和段落属性的默认值一节。

第 6 章

文本文档

除纯字串外，文本文档中还含有格式信息。这些格式信息可以出现在文本文档中的任意位置。如果使用表格，文档结构将更为复杂。表格不但包含一维字串，还包含二维字段。目前，大多数字处理程序都允许将绘图对象、文字框和其他对象插入到文字中。这些对象可以位于文字流之外，放到页面中的任意位置。

本章将讨论文本文档的主要接口和服务。第一节将详细介绍文本文档的结构，并通过一个 StarSuite 文档来集中说明如何使用 StarSuite Basic 程序执行迭代步骤。这一节主要讨论段落、段落部分及其格式。

第二节集中介绍如何高效地处理文本文档。为提高工作效率，StarSuite 提供了几个辅助对象，如 `TextCursor` 对象，这些对象是对第一节介绍的对象的补充。

第三节介绍一些文字以外的内容，主要有表格、文字框、文字字段、书签、内容目录等。

有关如何建立、打开、保存和打印文档的内容在第 5 章中介绍，因为这些内容不仅适用于文本文档，还适用于其他类型的文档。

文本文档的结构

文本文档主要含有以下四种信息：

- 实际的文字
- 用于格式化字符、段落和页面的文档样式
- 非文字元素，如表格、图形和绘图对象
- 文本文档的全局设定

本节集中讨论文字及相关的格式化选项。

> 用于 StarSuite Writer 的 StarSuite 6.x API 的设计与以前版本有所不同。在以前版本的 API 中，主要使用 `Selection` 对象，此对象主要面向针对最终使用者的使用者界面，所以旧的 API 专注于处理通过鼠标控制的突出显示。
>
> StarSuite 6.x API 替换了使用者界面和程序员接口之间的这些连接。这样，程序员可以并行访问应用程序的所有部分，可以同时处理文档不同部分的对象。原有的 `Selection` 对象不再可用。

段落和段落部分

文本文档的核心包含一系列段落。因为既没有为段落命名，也没有为段落建立索引，所以不能**直接**访问单个的段落。但是，可以借助第 4 章中介绍的 Enumeration 对象来顺序遍历段落。这样就可以编辑段落了。

但是，在使用 Enumeration 对象时，需要注意一个特别的情形：
它不仅返回段落，而且还返回表格 (严格地讲，在 StarSuite Writer 中，表格是一种特殊类型的段落)。因此，在访问返回的对象之前，应该检查返回的对象是支持用于段落的 com.sun.star.text. Paragraph 服务，还是支持用于表格的 com.sun.star.text. TextTable 服务。

以下示例使用循环对文本文档的内容进行遍历，并在每个实例中使用一条信息来告知使用者，所查询的对象是段落还是表格。

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

' 建立文本对象
Doc = StarDesktop.CurrentComponent

' 建立枚举对象
Enum = Doc.Text.createEnumeration

' 对所有文字元素进行循环
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
```

此示例建立了一个 Doc 文档对象，用它来引用当前的 StarSuite 文档。借助 Doc **对象，**示例随后建立了一个 Enumeration 对象，用该对象来遍历文字 (段落和表格) 的各个部分，并将当前元素指定到 TextElement 对象。示例使用了 supportsService 方法检查 TextElement 是段落还是表格，以发送相应的消息。

段落

com.sun.star.text.Paragraph 服务授权对段落内容的访问。可以使用 String 属性检索和更改段落中的文字：

```
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
```

示例打开当前的文本文档，并使用 `Enumeration` 对象遍历文档。它对所有段落使用 `TextElement.String` 属性以访问相关的段落，并将其中的字串 `you`、`too` 和 `for` 替换为字符 `U`、`2` 和 `4`。
用于替换的 `Replace` 函数不属于标准的 StarSuite Basic 语言范畴。这就是第 3 章搜寻和替换一节中介绍的示例函数的一个实例。

此处介绍的用于访问文本文档段落的程序与 VBA 中使用的 `Paragraphs` 列单类似，后者由 VBA 中的 `Range` 和 `Document` 对象提供。在 VBA 中，要通过段落编号来访问段落 (例如，调用 `Paragraph(1)`)，而在 StarSuite Basic 中，应该使用以上所述的 `Enumeration` 对象。

StarSuite Basic 中没有与 VBA 中提供的 `Characters`、`Sentences` 和 `Words` 列单相对应的内容。但是，您可以切换到 `TextCursor`，利用它实现字符、句子和字词级的浏览 (请参阅 *TextCursor*)。

段落部分

前面的示例可以根据需要更改文字，但是有时可能会破坏格式。

这是因为段落是由单独的子对象组成的。这些子对象都包含自己的格式信息。例如，如果段落中心包含要以黑体打印的词，则在 StarSuite 中要用三个段落部分来表示：粗体类型前面的部分，然后是粗体词，最后是粗体类型后面的部分 (又变为正常字体)。

如果使用段落的 `String` 属性更改了段落的文字，那么 StarSuite 将首先删除旧的段落部分，然后插入新的段落部分。这样一来，旧的段落部分的格式就丢失了。

为了防止发生这种情况，使用者可以访问相关的段落部分，而不要访问整个段落。使用段落的 `Enumeration` 对象即可达到此目的。以下示例使用了一个双循环，来遍历一个文本文档的所有段落和段落含有的段落部分，并采用了前面示例中的替换过程：

```
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

' 对所有段落进行循环
While Enum1.hasMoreElements
	TextElement = Enum1.nextElement
	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = TextElement.createEnumeration

		' 对所有子段落进行循环
		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
```

此示例通过一个双循环遍历了文本文档。外层循环针对文字段落； 内层循环处理这些段落中的段落部分。示例代码使用字串的 `String` 属性更改了这些段落部分的内容，所做更改与前面示例中对段落的更改相同。由于可以直接编辑段落部分，因此在替换字串时，字串的格式信息仍然保留。

格式化

格式化文字的方法有多种。最简单的方法是将格式属性直接指定到文字序列，称为**直接格式化**。直接格式化主要用在较短的文档中，因为使用者可以用鼠标来指定格式。例如，可以使用粗体类型或加入中心线，以突出显示文字中的某个词。

除了直接格式化，还可以使用文档样式来格式化文字，这称为**间接格式化**。使用间接格式化，使用者可以将预设的文档样式指定到相关的文字部分。如果以后要更改文字的版式，只需更改文档样式即可。StarSuite 会更改所有使用该文档样式的文字部分的描述方式。

在 VBA 中，对象的格式化属性通常分布在一系列子对象中 (例如，`Range.Font`、`Range.Borders`、`Range.Shading` 和 `Range.ParagraphFormat`)。这些属性是通过层叠表达式 (例如，`Range.Font.AllCaps`) 访问的。在 StarSuite Basic 中，可以直接使用格式属性，但要使用相关的对象 (`TextCursor`、`Paragraph` 等)。以下两节将概要介绍 StarSuite 中的字符和段落属性。

在旧版本的 StarSuite API 中，主要使用 `Selection` 对象及其下级对象 (例如，`Selection.Font`、`Selection.Paragraph` 和 `Selection.Border`) 对文字进行格式化。在新的 API 中，各种对象中都有格式化属性 (`Paragraph`、`TextCursor` 等等)，并且可以

直接采用。下面的段落提供了可用的字符和段落属性的列单。

字符属性

针对单个字符的格式属性称为字符属性，其中包括粗体类型和字体类型。可以设定字符属性的对象必须支持 `com.sun.star.style.CharacterProperties` 服务。StarSuite 可以识别支持该服务的一整套服务，包括上面提到的用于段落的 `com.sun.star.text.Paragraph` 服务和用于段落部分的 `com.sun.star.text.TextPortion` 服务。

`com.sun.star.style.CharacterProperties` 服务不提供任何接口，但提供了一系列属性，通过这些属性可以定义和调用字符属性。StarSuite API 参考中提供了完整的字符属性列单。以下列单介绍了最重要的属性：

- **`CharFontName (字串)`** - 选定字体类型的名称。
- **`CharColor (长整数)`** - 文字颜色。
- **`CharHeight (浮点数)`** - 字符高度，以点 (pt) 为单位。
- **`CharUnderline (常数组)`** - 下划线类型 (常数与 `com.sun.star.awt.FontUnderline` 一致)。
- **`CharWeight (常数组)`** - 字体的线宽 (常数与 `com.sun.star.awt.FontWeight` 一致)。
- **`CharBackColor (长整数)`** - 背景颜色。
- **`CharKeepTogether (布尔)`** - 禁止自动换行。
- **`CharStyleName (字串)`** - 字符样式的名称。

段落属性

不针对单个字符而是应用到整个段落的格式信息称为段落属性，其中包括段落与页边距的间隔以及段落的行距。可以通过 `com.sun.star.style.ParagraphProperties` 服务来使用段落属性。

可以在各种对象中使用段落属性。所有支持 `com.sun.star.text.Paragraph` 服务的对象同时也支持 `com.sun.star.style.ParagraphProperties` 中的段落属性。

StarSuite API 参考中提供了完整的段落属性列单。最常用的段落属性包括：

- **`ParaAdjust (枚举)`** - 垂直文字方向 (常数与 `com.sun.star.style.ParagraphAdjust` 一致)。
- **`ParaLineSpacing (结构)`** - 行距 (结构与 `com.sun.star.style.LineSpacing` 一致)。
- **`ParaBackColor (长整数)`** - 背景颜色。
- **`ParaLeftMargin (长整数)`** - 左边距，以百分之一毫米为单位。
- **`ParaRightMargin (长整数)`** - 右边距，以百分之一毫米为单位。
- **`ParaTopMargin (长整数)`** - 上边距，以百分之一毫米为单位。
- **`ParaBottomMargin (长整数)`** - 下边距，以百分之一毫米为单位。

- **ParaTabStops (结构数组)** - 制表符的类型和位置 (`com.sun.star.style.TabStop` 结构的数组)。
- **ParaStyleName (字串)** - 段落样式的名称。

示例：简单的 HTML 输出

以下示例说明了如何使用格式信息。示例遍历了一个文本文档，建立了一个简单的 HTML 文件。为此，所有段落都记录为相应的 HTML 元素 <P>。以粗体类型显示的段落部分在输出时使用 <B> HTML 元素进行标记。

```
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

' 对所有段落进行循环
While Enum1.hasMoreElements
      TextElement = Enum1.nextElement
      If TextElement.supportsService("com.sun.star.text.Paragraph") Then
            Enum2 = TextElement.createEnumeration
            CurLine = "<P>"

            ' 对所有段落部分进行循环
            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

            ' 输出行
            CurLine = CurLine & "</P>"
            Print #FileNo, CurLine

      End If
Wend

' 写入 HTML 页脚
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

此示例的基本结构基于前面讨论的用于遍历文本文档中段落部分的示例。但这个示例中加入了写入 HTML 文件的函数，以及检查相应文字部分字体线宽的测试代码，并为粗体类型的段落部分提供了对应的 HTML 标记。

字符和段落属性的默认值

直接格式化的优先度始终要高于间接格式化。换句话说，在文本文档中，使用文档样式进行格式化比直接格式化的优先度要低。

要确定文档的某个部分是采用了直接格式化还是采用了间接格式化并不容易。StarSuite 提供的图标栏显示了常用的文字属性，如字体类型、线宽和大小。但是，文字中相应的设定是基于文档样式，还是采用了直接格式化仍然不清楚。

StarSuite Basic 提供了 `getPropertyState` 方法，程序员可以使用该方法来检查某个属性是如何进行格式化的。该方法以属性名称作为参数，返回一个可以提供有关格式化来源信息的常数。可能出现以下在 `com.sun.star.beans.PropertyState` 枚举中定义的响应：

- **`com.sun.star.beans.PropertyState.DIRECT_VALUE`** - 属性是直接在文字中定义的 (直接格式化)
- **`com.sun.star.beans.PropertyState.DEFAULT_VALUE`** - 属性是由文档样式定义的 (间接格式化)
- **`com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE`** - 属性不明确。
  例如，对于既包括粗体又包括正常字体的段落，在查询这种段落的粗体类型属性时，就会出现这种状态。

以下示例显示了如何在 StarSuite 中编辑格式属性。示例将搜寻文本文档中那些直接格式化为粗体类型的段落部分。如果示例找到了相应的段落部分，将使用 `setPropertyToDefault` 方法删除直接格式化，并将字符样式 `MyBold` 指定到这些段落部分。

```
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

' 对所有段落进行循环
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration

              ' 对所有段落部分进行循环
              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
```

编辑文本文档

前一节已经讨论了用于编辑文本文档的一系列选项，重点介绍了 com.sun.star.text.TextPortion 和 com.sun.star.text.Paragraph 服务，它们分别可以授权对段落部分和段落的访问。这些服务适用于那些在一次循环中对文本文档内容进行编辑的应用程序。但是，对于很多问题，这样是不够的。StarSuite 提供了 com.sun.star.text.TextCursor 服务，用以解决更为复杂的任务，包括在文档中向后浏览，或者基于句子或字词而不是基于 TextPortions 进行浏览。

TextCursor

StarSuite API 中的 TextCursor 与 StarSuite 文档中使用的可见光标类似。它在文本文档中标记出某一点，通过使用命令向不同的方向进行浏览。但是，不要将 StarSuite Basic 中的 TextCursor 对象与可见光标混淆。它们是两种不同的事物。

警告! 与 VBA 中使用的术语不同：就功能范围而言，VBA 的 Range 对象相当于 StarSuite 中的 TextCursor 对象，却与 StarSuite 中的 Range 不同 (尽管它们的名称相同)。

例如，StarSuite 中的 TextCursor 对象提供了浏览和更改文字的方法，而 VBA 中的 Range 对象也提供了这些方法 (例如，MoveStart、MoveEnd、InsertBefore 和 InsertAfter)。以下各节将介绍 StarSuite 中 TextCursor 对象的相关内容。

在文本文档中浏览

StarSuite Basic 中的 TextCursor 对象与文本文档中的可见光标是相互独立的。在任何情况下，通过编程更改 TextCursor 对象的位置都不会影响可见光标。您甚至可以为同一个文档打开多个 TextCursor 对象并在相互独立的不同位置使用。

TextCursor 对象是通过调用 createTextCursor 建立的：

```
Dim Doc As Object
Dim Cursor As Object

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

用这种方法建立的 Cursor 对象支持 com.sun.star.text.TextCursor 服务，而该服务又提供了一整套浏览文本文档的方法。以下示例首先将 TextCursor 向左移动十个字符，然后再向右移动三个字符：

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

TextCursor 可以突出显示一个完整的区域。这与使用鼠标突出显示文字中的某一点相类似。以上函数调用中的 False 参数指定了光标移动所覆盖的区域是否突出显示。例如，下面示例中的 TextCursor

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

首先向右移动十个字符但不突出显示覆盖的区域，然后再向左移动三个字符并突出显示区域。因此，由 TextCursor 突出显示的区域从文字中的第七个字符开始，到第十个字符为止。

以下是 com.sun.star.text.TextCursor 服务提供的、用于浏览的主要方法：

- **goLeft (Count, Expand)** - 向左跳过 Count 个字符。
- **goRight (Count, Expand)** - 向右跳过 Count 个字符。
- **gotoStart (Expand)** - 跳转到文本文档的开头。
- **gotoEnd (Expand)** - 跳转到文本文档的结尾。
- **gotoRange (TextRange, Expand)** - 跳转到指定的 TextRange 对象。
- **gotoStartOfWord (Expand)** - 跳转到当前词的开头。
- **gotoEndOfWord (Expand)** - 跳转到当前词的结尾。
- **gotoNextWord (Expand)** - 跳转到下一个词的开头。
- **gotoPreviousWord (Expand)** - 跳转到前一个词的开头。
- **isStartOfWord ()** - 如果 TextCursorTrue 位于词的开头，则返回 True。
- **isEndOfWord ()** - 如果 TextCursor 位于词的结尾，则返回 True。
- **gotoStartOfSentence (Expand)** - 跳转到当前句子的开头。
- **gotoEndOfSentence (Expand)** - 跳转到当前句子的结尾。
- **gotoNextSentence (Expand)** - 跳转到下一句的开头。
- **gotoPreviousSentence (Expand)** - 跳转到前一句的开头。
- **isStartOfSentence ()** - 如果 TextCursor 位于句子的开头，则返回 True。
- **isEndOfSentence ()** - 如果 TextCursor 位于句子的结尾，则返回 True。
- **gotoStartOfParagraph (Expand)** - 跳转到当前段落的开头。
- **gotoEndOfParagraph (Expand)** - 跳转到当前段落的结尾。
- **gotoNextParagraph (Expand)** - 跳转到下一个段落的开头。
- **gotoPreviousParagraph (Expand)** - 跳转到前一个段落的开头。
- **isStartOfParagraph ()** - 如果 TextCursor 位于段落的开头，则返回 True。
- **isEndOfParagraph ()** - 如果 TextCursor 位于段落的结尾，则返回 True。

文本文档根据句子符号被划分为句子。例如，句号被解释为指示句子结尾的符号。

Expand 参数是一个布尔值，用来指定是否要突出显示浏览期间覆盖的区域。所有的浏览方法都将返回一个参数，以指定浏览是否成功，或者指定操作是否由于缺少文字而终止。

以下列单中的几种方法，用于编辑使用 TextCursor 突出显示的区域，这些方法也支持 com.sun.star.text.TextCursor 服务：

- **collapseToStart ()** - 重新设定突出显示，并将 TextCursor 置于前一个突出显示的区域的开头。
- **collapseToEnd ()** - 重新设定突出显示，并将 TextCursor 置于前一个突出显示的区域的结尾。
- **isCollapsed ()** - 如果 TextCursor 当前未突出显示任何区域，则返回 True。

使用 TextCursor 格式化文本文档

com.sun.star.text.TextCursor 服务支持本章开头提到的所有字符属性和段落属性。

以下示例显示了如何将这些属性与 TextCursor 一起使用。
示例将遍历整个文档，并将每句的第一个词格式化为粗体类型。

```
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
```

示例首先为已打开的文本文档建立了一个文档对象，然后一句一句地遍历整个文档，并突出显示各句的第一个词，将其格式化为粗体。

检索并更改文字内容

如果 TextCursor 含有突出显示的区域，则可以通过 TextCursor 对象的 String 属性来获得这些文字。以下示例使用 String 属性在消息框中显示句子的第一个词：

```
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 属性以相同的方法对每个句子的第一个词进行更改：

```
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 含有突出显示的区域，则指定到 String 属性的新文字将替换该区域。如果没有突出显示的区域，文字被将插入到 TextCursor 当前所在的位置。

插入控制代码

有时，不需要更改文档的实际文字，而要更改文档的结构。为此，StarSuite 提供了控制代码。这些代码可以插入到文本文档中，并影响文档的结构。控制代码在 `com.sun.star.text.ControlCharacter` 常数组中定义。StarSuite 中可用的控制代码包括：

- **`PARAGRAPH_BREAK`** - 段落分隔符。
- **`LINE_BREAK`** - 段落内的换行符。
- **`SOFT_HYPHEN`** - 可能的音节点。
- **`HARD_HYPHEN`** - 必需的音节点。
- **`HARD_SPACE`** - 对齐文字中不能调整间隔的受保护的空格字符。

要插入控制代码，不仅需要光标还需要相关的文本文档对象。以下示例在文字中第 20 个字符的后面插入一个段落：

```
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)
```

调用 `insertControlCharacter` 方法时用到的参数 `False` 可以确保当前由 `TextCursor` 突出显示的区域在插入操作之后仍然存在。如果这里传递的是参数 `True`，那么 `insertControlCharacter` 将替换当前文字。

搜寻文字部分

在很多情况下，需要在文本文档中搜寻特定的项，并需要对相应的位置进行编辑。所有的 StarSuite 文档都为实现此功能提供了一个特殊的接口，该接口始终遵循相同的原则：在开始搜寻之前，必须首先建立一个 `SearchDescriptor`，它定义了 StarSuite 要在文档中搜寻的内容。`SearchDescriptor` 是一个对象，支持 `com.sun.star.util.SearchDescriptor` 服务，可以通过文档的 `createSearchDescriptor` 方法建立：

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

建立 `SearchDescriptor` 之后，它就会收到要搜寻的文字：

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

就其功能而言，`SearchDescriptor` 与 StarSuite 的搜寻对话框十分相似。与搜寻视窗一样，搜寻所需的设定可以在 `SearchDescriptor` 对象中设定。

由 com.sun.star.util.SearchDescriptor 服务提供的属性包括：

- **SearchBackwards (布尔)** - 向后而不是向前搜寻文字。
- **SearchCaseSensitive (布尔)** - 搜寻时考虑字符的大小写。
- **SearchRegularExpression (布尔)** - 处理搜寻表达式的方式与处理常规表达式的方式相同。
- **SearchStyles (布尔)** - 在文本文档中搜寻特定的段落样式。
- **SearchWords (布尔)** - 只搜寻完整的词。

StarSuite Basic 还提供了 SearchSimilarity (或“模糊匹配”) 功能。使用此功能，StarSuite 将搜寻与搜寻表达式类似但不完全相同的表达式。可以为这些表达式分别定义要附加的、删除的和更改的字符数量。以下是 com.sun.star.util.SearchDescriptor **服务的相关**属性：

- **SearchSimilarity (布尔)** - 执行类似搜寻。
- **SearchSimilarityAdd (短整数)** - 向类似搜寻加入的字符数量。
- **SearchSimilarityExchange (短整数)** - 类似搜寻中被替换部分的字符数量。
- **SearchSimilarityRemove (短整数)** - 类似搜寻中被删除部分的字符数量。
- **SearchSimilarityRelax (布尔)** - 在搜寻表达式中考虑所有偏差规则。

按要求准备好 SearchDescriptor 之后，就可以在文本文档中采用了。为此，StarSuite 文档提供了 findFirst 和 findNext 方法：

```
Found = Doc.findFirst (SearchDesc)

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

示例通过一次循环查找所有的匹配项，并返回一个 TextRange 对象，该对象用于引用找到的文字段。

示例：类似搜寻

此示例显示了如何在文本文档中搜寻词“turnover”，并将搜寻结果格式化为粗体类型。示例中使用了类似搜寻，这样，不仅可以找到词“turnover”，还可以找到其复数形式“turnovers”，以及有偏差的内容，如“turnover's”。找到的表达式与搜寻表达式最多有两个字母不同：

```
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，搜寻和替换的基本思想与 VBA 中的相同。这两种接口都为您提供了一个对象，通过该对象可以定义搜寻和替换的属性。然后将该对象用于所需的文字区域以执行操作。在 VBA 中，可以通过 `Range` 对象的 `Find` 属性来使用相关的辅助对象，而在 StarSuite Basic 中则要调用文档对象的 `createSearchDescriptor` 或 `createReplaceDescriptor` 来建立此辅助对象。它们的搜寻属性和方法也不同。

与 StarSuite 旧版本的 API 一样，新 API 中对文字的搜寻和替换也是使用文档对象来执行的。以前有一个称为 `SearchSettings` 的对象专门用于定义搜寻选项，而在新版本中，使用 `SearchDescriptor` 来搜寻对象，或者使用 `ReplaceDescriptor` 对象来自动替换文字。这些对象不仅涵盖选项，而且包括当前的搜寻文字，必要时，还可以包括相关的替换文字。描述符对象是使用文档对象建立的，根据相关的请求完成，然后作为搜寻方法的参数传送回文档对象。

替换文字部分

与搜寻功能一样，StarSuite Basic 中也提供了 StarSuite 的替换功能。 这两种功能的处理方法相同。替换过程首先也需要一个用于记录过程参数的特殊对象。这个对象称为 ReplaceDescriptor，它支持 com.sun.star.util.ReplaceDescriptor 服务。以上段落介绍的 SearchDescriptor 的所有属性 ReplaceDescriptor 也支持。例如，在替换过程中，可以启动或关闭区分大小写，也可以执行类似搜寻。

以下示例示范了如何使用 ReplaceDescriptors 在 StarSuite 文档中进行搜寻。

```
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
```

搜寻和替换表达式是使用 ReplaceDescriptors 的 SearchString 和 ReplaceString 属性设定的。而实际的替换过程最终是使用文档对象的 replaceAll 方法实现的，此方法对所有搜寻到的搜寻表达式进行替换。

示例：使用常规表达式搜寻和替换文字

StarSuite 的替换功能在与常规表达式结合使用时尤其有效。这时，可以使用通配符和特殊字符来定义变量搜寻表达式，而不必使用固定的值。

StarSuite 支持的常规表达式在 StarSuite 联机帮助中有详细介绍。以下是一些示例：

- 搜寻表达式中的句号表示任意字符。因此，搜寻表达式 sh.rt 既可以表示 shirt，又可以表示 short。
- 字符 ^ 用于标记段落的开头。因此，要搜寻所有出现在段落开头的名字 Peter，可以使用搜寻表达式 ^Peter。
- 字符 $ 用于标记段落的结尾。因此，要搜寻所有出现在段落结尾的名字 Peter，可以使用搜寻表达式 Peter$。
- * 表示前面的字符可以重复出现若干次。它可以与句号结合使用，作为代表任意字符的通配符。例如，表达式 temper.*e 可以表示表达式 temperance 和 temperature。

以下示例显示了如何借助常规表达式 ^$ 来删除文本文档中的所有空行。

```
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)
```

文本文档：文字以外的内容

到目前为止，本章仅讨论了文字段落及段落部分。但是，文本文档中还含有其他对象，包括表格、绘图、文字字段和目录。这些对象可以锁定到文本文档中的任意位置。

由于有了这些常用功能，StarSuite 中的所有这些对象都支持一个名为 com.sun.star.text.TextContent 的常用基本服务。此服务提供了以下属性：

- **AnchorType (枚举)** - 确定 TextContent 对象的锁定类型 (默认值与 com.sun.star.text.TextContentAnchorType 枚举 一致)。
- **AnchorTypes (枚举序列)** - 所有支持特殊的 TextContent 对象的 AnchorTypes 的枚举。
- **TextWrap (枚举)** - 确定 TextContent 对象周围文字的环绕类型 (默认值与 com.sun.star.text.WrapTextMode 枚举 一致)。

TextContent 对象还共享一些方法，尤其是建立、插入和删除对象的方法。

- 使用文档对象的 createInstance 方法建立新的 TextContent 对象。
- 使用文本对象的 insertTextContent 方法插入对象。
- 使用 removeTextContent 方法删除 TextContent 对象。

下面各节将介绍一些使用这些方法的示例。

表格

以下示例将使用前面介绍的 createInstance 方法建立一个表格。

```
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)
```

建立表格之后，调用 `initialize` 将表格设定为要求的行数和列数，然后使用 `insertTextContent` 将其插入到文本文档中。

正如示例所示，`insertTextContent` 方法不仅要求插入 `Content` 对象，还需要使用另外两个参数：

- 一个 `Cursor` 对象，用于确定插入位置
- 一个布尔变量，用于指定用 `Content` 对象替换光标当前选定的内容 (`True` 值)，或者在当前选定的文字之前插入 `Content` 对象 (`False` 值）

当在文本文档中建立和插入表格时，StarSuite Basic 使用的对象与 VBA 中的对象相似：文档对象和 StarSuite Basic 中的 `TextCursor` 对象 (在 VBA 中是 `Range` 对象)。在 VBA 中，`Document.Tables.Add` 方法用于建立和设定表格，而在 StarSuite Basic 中，则要使用 `createInstance` 建立表格，通过 `insertTextContent` 进行初始化并插入到文档中，上面的示例即说明了这一点。

可以使用一个简单的循环来确定插入到文本文档中的表格。使用文本文档对象的 `getTextTables()` 方法即可实现此目的：

```
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)

      ' 编辑表格
Next I
```

StarSuite 6 中的文字表格是通过文档对象的 `TextTables` 列单获得的。此列单取代了以前由 `Selection` 对象提供的表格列单。以上的示例显示了建立文字表格的方法。用于访问文字表格的选项将在下一节介绍。

编辑表格

表格是由各个行组成的，而行是由许多单元格组成的。严格地讲，StarSuite 中并没有表格列。列是由相邻的行 (一个接一个) 隐含地组合而成。但是，为了简化对表格的访问，StarSuite 提供了一些使用列进行操作的方法。如果表格中没有合并的单元格，这些方法将很有用。

让我们先来了解一下表格本身的属性。这些属性是在 `com.sun.star.text.TextTable` 服务中定义的。以下列单包含了表格对象最重要的属性：

- **BackColor (长整数)** - 表格的背景颜色。
- **BottomMargin (长整数)** - 下边距，以百分之一毫米为单位。
- **LeftMargin (长整数)** - 左边距，以百分之一毫米为单位。
- **RightMargin (长整数)** - 右边距，以百分之一毫米为单位。
- **TopMargin (长整数)** - 上边距，以百分之一毫米为单位。
- **RepeatHeadline (布尔)** - 在每个页面上重复表格标题。
- **Width (长整数)** - 表格的绝对宽度，以百分之一毫米为单位。

行

表格是由含有行的列单组成的。以下示例显示了如何对表格的行进行检索和格式化。

```
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
```

示例首先调用 `Table.getRows` 建立了一个含有所有行的列单。利用 `getCount` 和 `getByIndex` 方法可以进一步处理列单，它们都属于 `com.sun.star.table.XtableRows` 接口。`getByIndex` 方法返回一个行对象，该对象支持 `com.sun.star.text.TextTableRow` 服务。

以下是 `com.sun.star.table.XtableRows` 接口的主要方法：

- **getByIndex(Integer)** - 返回指定的索引对应的行对象。
- **getCount()** - 返回行对象的数量。
- **insertByIndex(Index, Count)** - 从 `Index` 位置开始在表格中插入 Count 个行。
- **removeByIndex(Index, Count)** - 从 `Index` 位置开始从表中删除 Count 个行。

`getByIndex` 和 `getCount` 方法可用于所有表格，而 `insertByIndex` 和 `removeByIndex` 方法只能用于不包含合并的单元格的表格。

`com.sun.star.text.TextTableRow` 服务提供了以下属性：

- **BackColor (长整数)** - 行的背景颜色。
- **Height (长整数)** - 行的高度，以百分之一毫米为单位。
- **IsAutoHeight (布尔)** - 表格高度根据内容动态调整。
- **VertOrient (常数)** - 文字框的垂直方向，即有关表格中文字垂直方向的细节 (值与 `com.sun.star.text.VertOrientation` 一致)

列

访问列的方法与行相同，即对 `Column` 对象使用 `getByIndex`、`getCount`、`insertByIndex` 和 `removeByIndex` 方法，而此对象是通过 `getColumns` 获得的。但是，这些方法只能用于不含合并的单元格的表格。在 StarSuite Basic 中，不能按列格式化单元格，而必须使用对单个表格单元格进行格式化的方法。

单元格

StarSuite 文档中的每个单元格都有唯一的名称。当 StarSuite 光标位于单元格内时，可以在状态栏上看到该单元格的名称。左上角的单元格一般称为 A1，右下角的单元格称为 Xn，其中 X 表示最高一列的字母，n 表示最后一行的编号。单元格对象可以通过表格对象的 getCellByName() 方法获得。以下示例显示了一个循环，它遍历了表格的所有单元格，并将对应的行、列编号输入到单元格中。

```
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
```

表格单元格与标准文字一样，它支持用于建立相关的 TextCursor 对象的 createTextCursor 接口。

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

因此，所有用于单个字符和段落的格式化选项都可用于表格单元格。

以下示例将对文本文档中的所有表格进行搜寻，并通过相应的段落属性对含有数字值的单元格采用右对齐的格式。

```
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
```

此示例建立了一个 `TextTables` 列单，它含有在一个循环中遍历的所有文字表格。然后，StarSuite 建立了含有这些表格的单元格名称的列单，又在另一个循环中遍历这些单元格。如果单元格中含有数字值，示例就会更改其格式。为此，示例首先建立了一个 `TextCursor` 对象，用它来引用表格单元格的内容，然后调整表格单元格的段落属性。

文字框

文字框和表格、图形一样，也是 `TextContent` 对象。文字框主要由标准文字组成，但可以置于页面上的任意位置而不包括在文字流中。

与所有 `TextContent` 对象一样，在文档中建立文字框和插入文字框也是有区别的。

```
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)
```

建立文字框要使用文档对象的 `createInstance` 方法。以这种方法建立的文字框，可以使用 `Text` 对象的 `insertTextContent` 方法插入到文档中。为此，应该指定正确的 `com.sun.star.text.TextFrame` 服务的名称。

文字框的插入位置由 `Cursor` 对象确定，而且在插入时也会执行此对象。

StarSuite 中的文字框与 Word 中的位置框相对应。而 VBA 使用 `Document.Frames.Add` 方法来建立位置框，上述过程则使用了 TextCursor 和文档对象的 `createInstance` 方法。

文字框对象提供了一系列可以影响框的位置和行为的属性。这些属性大部分是在 `com.sun.star.text.BaseFrameProperties` 服务中定义的，各个 TextFrame 服务也支持这些属性。主要的属性包括：

- **`BackColor` (长整数)** - 文字框的背景颜色。
- **`BottomMargin` (长整数)** - 下边距，以百分之一毫米为单位。
- **`LeftMargin` (长整数)** - 左边距，以百分之一毫米为单位。
- **`RightMargin` (长整数)** - 右边距，以百分之一毫米为单位。
- **`TopMargin` (长整数)** - 上边距，以百分之一毫米为单位。
- **`Height` (长整数)** - 文字框的高度，以百分之一毫米为单位。
- **`Width` (长整数)** - 文字框的宽度，以百分之一毫米为单位。
- **`HoriOrient` (常数)** - 文字框的水平方向 (与 `com.sun.star.text.HoriOrientation` 一致)。
- **`VertOrient` (常数)** - 文字框的垂直方向 (与 `com.sun.star.text.VertOrientation` 一致)。

以下示例使用以上属性建立了一个文字框：

```
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)
```

示例建立了一个 `TextCursor`，作为文字框的插入标记。它位于文本文档的第一个词和第二词之间。文字框是使用 `Doc.createInstance` 建立的。文字框对象的属性设定为需要的起始值。

在这里应该注意 `AnchorType` 属性 (来自 `TextContent` 服务) 和 `VertOrient` (来自 `BaseFrameProperties` 服务) 属性之间的交互。`AnchorType` 接收 `AS_CHARACTER` 值，因此，文字框将直接插入到文字流中，其行为也与字符类似。例如，如果遇到换行符，文字框将移到下一行。`VertOrient` 属性的 `LINE_TOP` 值则确保文本框的上边缘与字符的上边缘高度相同。

完成初始化之后，调用 `insertTextContent` 将文字框插入到文本文档中。

要编辑文字框中的内容，使用者应该使用 TextCursor。该对象已经介绍过多次，它也可用于文字框。

```
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!"
```

示例建立了一个文字框，将其插入到当前文档中，并为文字框打开一个 TextCursor。此光标用于将框中的字体设定为粗体类型，将段落的方向设定为居中。最后，向文字框指定字串“This is a small test!”。

文字字段

文字字段是 TextContent 对象，因为它除提供纯文字外，还提供了其他逻辑。要将文字字段插入到文本文档中，可以使用与处理其他 TextContent 对象相同的方法：

```
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)
```

此示例将含有当前日期的文字字段插入到当前文本文档的开头。IsDate 属性设定为 True 值，以确保仅显示日期而不显示时间。IsFixed 设定为 False 值，可以确保在打开文档时自动更新日期。

在 VBA 中，字段类型由 Document.Fields.Add 方法的参数指定，而在 StarSuite Basic 中，用于处理字段类型的服务名称就定义了字段的类型。

以前，文字字段是通过使用 StarSuite 中旧的 `Selection` 对象提供的方法 (如 `InsertField`、`DeleteUserField` 和 `SetCurField`) 来访问的。

在 StarSuite 6 中，字段的管理运用了面向对象的概念。要建立文字字段，首先要建立所需类型的字段，并使用所需的属性进行初始化。然后，使用 `insertTextContent` 方法将文字字段插入到文档中。前面的示例就验证了这个过程。下面各节将介绍最重要的字段类型及其属性。

除插入文字字段以外，在文档中搜寻字段也是一项很重要的任务。以下示例显示了如何通过循环来遍历文本文档中的所有文字字段，并检查它们的相关类型。

```
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
```

建立文字字段的起点是文档对象的 `TextFields` 列单。示例根据此列单建立了一个 `Enumeration` 对象，利用该对象，可以在一次循环中查询所有的文字字段。对于找到的文字字段，使用 `supportsService` 方法检查字段支持的服务。如果字段被证明是日期/ 时间字段或是备注，则在信息框中显示对应的字段类型。另一方面，如果示例遇到其他字段，则显示信息 “unknown”。

下面，将为您介绍最重要的文字字段及其相关属性。`com.sun.star.text.TextField` 模块中的 API 参考提供了完整的文字字段列单。(在 StarSuite Basic 中，列举文字字段的服务名称时，应该使用大写和小写字符，如前例所示。)

页数、字数和字符数

文字字段

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

可以返回文本文档的页数、字数和字符数。它们支持以下属性：

- **`NumberingType (常数)`** - 编号格式 (规则与 `com.sun.star.style.NumberingType` 中的常数一致)。

当前页面

可以使用 `com.sun.star.text.TextField.PageNumber` 文字字段将当前页面的页码插入到文档中。可以指定以下属性：

- **`NumberingType (常数)`** - 编号格式 (规则与 `com.sun.star.style.NumberingType` 中的常数一致)。
- **`Offset (短整数)`** - 为页码加入的偏移 (可以指定负数)。

以下示例显示了如何将页码插入到文档的页脚中。

```
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)
```

示例首先建立了一个支持 `com.sun.star.text.TextField.PageNumber` 服务的文字字段。由于页眉行和页脚行被定义为 StarSuite 页面样式的一部分，因此首先使用所有 `PageStyles` 的列单来建立页面样式。

为了确保页脚行可见，`FooterIsOn` 属性设定为 `True`。然后，使用左侧页脚行的相关对象将文字字段插入到文档中。

备注

备注字段 (com.sun.star.text.TextField.Annotation) 在文本文档中以小的黄色符号表示。单击该符号可以打开文字字段，其中记录了对当前位置的文字的注解。备注字段具有以下属性。

- **Author (字串)** - 作者的姓名。
- **Content (字串)** - 注解文字。
- **Date (日期)** - 书写备注的日期。

日期/ 时间

日期/ 时间字段 (com.sun.star.text.TextField.DateTime) 表示当前日期或当前时间。它支持以下属性：

- **IsFixed (布尔)** - 如果为 True,则保持插入时的时间细节；否则，则在每次打开文档时进行更新。
- **IsDate (布尔)** - 如果为 True，字段显示当前日期；否则显示当前时间。
- **DateTimeValue (结构)** - 字段的当前内容 (com.sun.star.util.DateTime 结构)
- **NumberFormat (常数)** - 描述时间或日期所用的格式。

章节名称/ 章节编号

当前章节的名称可以通过 com.sun.star.text.TextField.Chapter 类型的字段获得。可以使用两个属性来定义表单。

- **ChapterFormat (常数)** - 确定是否描述章节名称或章节编号 (与 com.sun.star.text.ChapterFormat 一致)
- **Level (整数)** - 确定要显示的章节名称和/ 或章节编号的章节级。0 表示可用的最高级。

书签

书签 (服务 com.sun.star.text.Bookmark) 是 TextContent 对象。使用前面介绍的概念即可建立和插入书签：

```
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)
```

示例建立了一个 Cursor，它标记书签的插入位置，然后标记实际的书签对象 (Bookmark)。然后，为书签指定了名称，并通过 insertTextContent 将书签插入到文档中光标所在的位置。

文本文档的书签通过一个名为 Bookmarks 的列单来访问。可以通过编号或名称访问书签。

以下示例显示了如何在文本文档中查找书签，并在书签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 方法以按名称查找所需的书签。createTextCursorByRange 调用随后建立了一个 Cursor，它位于书签的锁定位置。然后，光标将所需的文字插入到该点。

第 **7** 章

工作表文档

StarSuite Basic 为通过编程建立和编辑工作表提供了丰富的接口。本章介绍如何控制工作表文档的相关服务、方法和属性。

第一节介绍工作表文档的基本结构，说明如何访问和编辑各个单元格的内容。

第二节通过重点介绍单元格区域以及用于搜寻和替换单元格内容的选项，来集中说明如何高效地编辑工作表。

Range 对象允许您处理任意的表格区域，目前已包含在新的 API 中。

基于表格的文档 (工作表) 的结构

工作表的文档对象是基于 `com.sun.star.sheet.SpreadsheetDocument` 服务的。对于这种文档，每个文档中都可能含有多个工作表。在本书中，基于表格的文档或工作表文档是指整个文档，而工作表是指文档中的一个表格。

对于工作表及其内容，VBA 和 SatrOffice Basic 中分别使用了不同的术语。文档对象在 VBA 中称为 *Workbook*，其含有的各个页面称为 *Worksheet*，而在 StarSuite Basic 中，这两者分别称为 *SpreadsheetDocument* 和 *Sheet*。

工作表文档

您可以通过 `Sheets` 列单来访问工作表文档中的各个工作表。

以下示例显示了如何通过工作表编号或工作表名称来访问工作表。

示例 **1**：通过编号访问 **(**编号从 **0** 开始**)**

```
Dim Doc As Object
Dim Sheet As Object

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

示例 **2**：通过名称访问

```
Dim Doc As Object
Dim Sheet As Object

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

第一个示例是通过编号 (从 0 开始) 访问工作表的。第二个示例是通过工作表名称和 `getByName` 方法访问工作表的。

通过 `getByName` 方法获得的 `Sheet` 对象支持 `com.sun.star.sheet.Spreadsheet` 服务。而该服务除提供了若干用于编辑内容的接口外，还提供了以下属性：

- **`IsVisible` (布尔)** - 工作表是可见的。
- **`PageStyle` (字串)** - 工作表的页面样式名称。

建立、删除和重命名工作表

还可以通过工作表文档的 `Sheets` 列单来建立、删除和重命名各个工作表。以下示例使用 `hasByName` 方法检查名为 *MySheet* 的工作表是否存在。如果存在，方法就使用 `getByName` 方法确定相应的对象引用，然后将引用存盘到 `Sheet` 的变量中。如果相应的工作表不存在，则调用 `createInstance` 建立它，并使用 `insertByName` 方法将其插入到工作表文档中。

```
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` 和 `insertByName` 方法来自 `com.sun.star.container.XnameContainer` 接口，第 4 章中介绍了此接口。

行和列

工作表由行和列组成，可以通过工作表对象的 Rows 和 Columns 属性获得它们。列和行支持 com.sun.star.table.TableColumns 服务和/ 或 com.sun.star.table.TableRows 服务。

以下示例建立了两个对象来引用工作表的第一行和第一列，并将引用保存在 FirstCol 和 FirstRow 对象变量中。

```
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)
```

列对象支持 com.sun.star.table.TableColumn 服务，此服务具有以下属性：

- **Width (长整数)** - 列宽，以百分之一毫米为单位。
- **OptimalWidth (布尔)** - 将列设定为最佳宽度。
- **IsVisible (布尔)** - 显示列。
- **IsStartOfNewPage (布尔)** - 打印时，在列前建立换页符。

只有当 OptimalWidth 属性设定为 True 时，才能获得最佳列宽。这时，如果更改一个单元格的宽度，则此单元格所在列的宽度不会改变。就功能而言，OptimalWidth 更像是一个方法而不是属性。

行对象是基于 com.sun.star.table.RowColumn 服务的，具有以下属性：

- **Height (长整数)** - 行的高度，以百分之一毫米为单位。
- **OptimalHeight (布尔)** - 将行设定为最佳高度。
- **IsVisible (布尔)** - 显示行。
- **IsStartOfNewPage (布尔)** - 打印时，在行前建立分页符。

如果行的 OptimalHeight 属性设定为 True，则在更改单元格的高度时，其所在行的高度将自动进行更改。除非通过 Height 属性指定行的绝对高度，否则将一直自动采用最佳行高。

以下示例使工作表中的前五行自动获得最佳高度，并使第二列不可见。

```
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，可以通过索引访问 Rows 和 Columns 列单。但与 VBA 不同，第一列的索引是 0 而不是 1。

插入和删除行和列

工作表的 Rows 和 Columns 对象可以访问现有的行和列，也可以插入、删除行和列。

```
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 方法在工作表第四列的位置 (索引为 3，编号从 0 开始) 插入一个新列。第二个参数指定要插入的列的数目 (在本例中为 1)。

removeByIndex 方法删除了第六列 (索引为 5)。同样，第二个参数指定要删除的列的数目。

使用 Rows 对象插入和删除行的方法与这里介绍的使用 Columns 对象编辑列的方法一样。

单元格

工作表由含有单元格的二维列单组成。每个单元格都是用相对于左上角单元格 (位置为 0,0) 的 X 位置和 Y 位置来定义的。

以下示例建立了一个引用左上角单元格的对象，并在单元格中插入文字：

```
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"
```

除数字坐标外，工作表的单元格还有名称，例如，工作表左上角的单元格 (0,0) 称为 A1。字母 A 表示列，数字 1 表示行。请勿混淆单元格的名称和位置，因为名称的行计数是从 1 开始的，而位置计数是从 0 开始的。

在 StarSuite 中，表格单元格可以为空，也可以含有文字、数字或公式。单元格的类型不是由存盘在单元格中的内容决定的，而是由用于其条目的对象属性决定的。使用 Value 属性可以插入并调用数字,使用 String 属性和 Formula 属性可以分别插入并调用文字和公式。

```
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"
```

此示例在单元格 A1 到 A3 中分别插入了一个数字、一个词和一个公式。

Value、String 和 Formula 属性取代了用于设定表格单元格的值的 PutCell 方法。

对于使用 String 属性输入的单元格内容，StarSuite 会将其作为文字处理，即使输入的是数字也是如此。用这种方法输入的数字会在单元格中左对齐，而不是右对齐。使用公式时，也要注意文字和数字的区别：

```
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
```

尽管单元格 A1 中含有的数值是 100，单元格 A2 中含有的数值是 1000，但是公式 A1+A2 却返回 100。这是因为单元格 A2 中的内容是作为字串输入的，并不是数字。

要检查单元格的内容是字串还是数字，请使用 Type 属性：

```
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 属性返回的 com.sun.star.table.CellContentType 枚举值可以标识单元格内容的类型。可能的值包括：

- **EMPTY** - 无值
- **VALUE** - 数字
- **TEXT** - 字串
- **FORMULA** - 公式

插入、删除、复制和移动单元格

除了可以直接更改单元格的内容，StarSuite Calc 还提供了一个接口，用于插入、删除、复制或合并单元格。可以通过工作表对象使用此接口 (com.sun.star.sheet.XRangeMovement)，它提供了四种更改单元格内容的方法。

insertCell 方法用于向工作表中插入单元格。

```
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)
```

此示例在工作表文档中第一个工作表 (编号为 0) 的第二列、第二行交叉处 (行、列编号都为 1) 插入一个大小为两行乘两列的单元格区域。指定的单元格区域内现有的数值被移到区域之下。

要定义插入的单元格区域，请使用 com.sun.star.table.CellRangeAddress 结构。该结构包括以下值：

- **Sheet (短整数)** - 工作表编号 (从 0 开始)。
- **StartColumn (长整数)** - 单元格区域中的第一列 (编号从 0 开始)。
- **StartRow (长整数)** - 单元格区域中的第一行 (编号从 0 开始)。
- **EndColumn (长整数)** - 单元格区域中的最后一列 (编号从 0 开始)。
- **EndRow (长整数)** - 单元格区域中的最后一行 (编号从 0 开始)。

完整的 CellRangeAddress 结构必须传递到 insertCells 方法，作为此方法的第一个参数。而 insertCells 的第二个参数含有一个 com.sun.star.sheet.CellInsertMode 枚举值，用来定义如何处理插入位置之前的值。CellInsertMode 枚举可以识别以下值：

- **NONE** - 当前值保留在当前位置。
- **DOWN** - 插入位置及插入位置以下的单元格向下移动。
- **RIGHT** - 插入位置及插入位置右侧的单元格向右移动。
- **ROWS** - 插入位置之后的行向下移动。
- **COLUMNS** - 插入位置之后的列向右移动。

removeRange 方法与 insertCells 方法相对应，它可以从工作表中删除 CellRangeAddress 结构定义的区域。

```
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)
```

此示例从工作表中删除了单元格区域 B2:C3，然后将下面的单元格向上移动两行。删除类型由以下某个 `com.sun.star.sheet.CellDeleteMode` 枚举值定义：

- **NONE** - 当前值保留在当前位置。
- **UP** - 插入位置及插入位置以下的单元格向上移动。
- **LEFT** - 插入位置及插入位置右侧的单元格向左移动。
- **ROWS** - 插入位置之后的行向上移动。
- **COLUMNS** - 插入位置之后的列向左移动。

`XRangeMovement` 接口提供了另外两种方法，用于移动 (`moveRange`) 和复制 (`copyRange`) 单元格区域。以下示例将移动 B2:C3 区域，以使该区域从 A6 开始：

```
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 `CellRangeAdress` 结构以外，`moveRange` 方法还需要使用 `com.sun.star.table.CellAddress` 结构来定义要移动到的目标区域的起始位置。`CellAddress` 方法提供了以下值：

- **Sheet (短整数)** - 工作表编号 (编号从 0 开始)。
- **Column (长整数)** - 目标列的编号 (编号从 0 开始)。
- **Row (长整数)** - 目标行的编号 (编号从 0 开始)。

目标区域中的单元格内容将被 moveRange 方法改写。
与 InsertCells 方法不同的是，removeRange 方法未提供执行自动移动的参数。

copyRange 的使用方法与 moveRange 方法相同，只是 copyRange 是插入单元格区域的复制件而不是移动区域。

就其功能而言，StarSuite Basic 中的 insertCell、removeRange 和 copyRange 方法与 VBA 中的 Range.Insert、Range.Delete 和 Range.Copy 方法相同。但在 VBA 中，这些方法用于相应的 Range 对象，而在 StarSuite Basic 中，则用于相关的 Sheet 对象。

格式化

工作表文档提供了用于格式化单元格和页面的属性和方法。

单元格属性

用于格式化单元格的选项有很多，如指定字体类型和文字大小等。所有单元格都支持 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties` 服务，这两种服务的主要属性已经在第 6 章 (文本文档) 中做了介绍。特殊的单元格格式化由 `com.sun.star.table.CellProperties` 服务来处理。该服务的主要属性将在以下各节中介绍。

所有介绍的属性既可用于单个的单元格，也可用于单元格区域。

> StarSuite API 中的 `CellProperties` 对象与 `VBA` 中的 `Interior` 对象相同，也是用于定义单元格专用的属性。

背景颜色和阴影

`com.sun.star.table.CellProperties` 服务提供了以下属性，用于定义背景颜色和阴影：

- **`CellBackColor (长整数)`** - 表格单元格的背景颜色。
- **`IsCellBackgroundTransparent (布尔)`** - 将背景颜色设定为透明。
- **`ShadowFormat (结构)`** - 指定单元格的阴影 (结构与 `com.sun.star.table.ShadowFormat` 一致)。

用于单元格阴影的 `com.sun.star.table.ShadowFormat` 结构和详细规范具有以下结构：

- **`Location (枚举)`** - 阴影的位置 (值来自 `com.sun.star.table.ShadowLocation` 结构)。
- **`ShadowWidth (短整数)`** - 阴影的大小，以百分之一毫米为单位。
- **`IsTransparent (布尔)`** - 将阴影设定为透明。
- **`Color (长整数)`** - 阴影的颜色。

以下示例将数字 1000 写到单元格 B2 中，使用 `CellBackColor` 属性将背景颜色更改为红色，然后为该单元格建立一个向左下方偏移 1 毫米的浅灰色阴影。

```
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
```

对齐

StarSuite 提供了很多功能，用于更改单元格中文字的对齐方式。

以下属性定义了文字的水平和垂直对齐方式：

- **HoriJustify (枚举)** - 文字的水平对齐方式 (值来自 `com.sun.star.table.CellHoriJustify`)
- **VertJustify (枚举)** - 文字的垂直对齐方式 (值来自 `com.sun.star.table.CellVertJustify`)
- **Orientation (枚举)** - 文字的方向 (值与 `com.sun.star.table.CellOrientation` 一致)
- **IsTextWrapped (布尔)** - 允许在单元格中自动换行
- **RotateAngle (长整数)** - 文字的旋转角度，以百分之一度为单位

以下示例显示了如何排列单元格的内容，以使各个字符从单元格的左上角开始逐个排列。字符并未旋转。

```
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
```

数字、日期和文字格式

StarSuite 提供了一整套预设的日期和时间格式。每种格式都有一个内部编号，通过它使用 `NumberFormat` 属性为单元格指定格式。StarSuite 提供了 `queryKey` 和 `addNew` 方法，从而使您可以访问现有的数字格式，也可以建立您自己的数字格式。通过以下的对象调用可以访问这些方法：

```
NumberFormats = Doc.NumberFormats
```

格式是使用格式字串指定的，格式字串的结构与 StarSuite Basic 格式函数的结构相同。但它们之间有一个重要区别：命令格式要求使用英文缩写、小数点或作为千位分隔符的字符，而 NumberFormats 对象的命令格式结构必须使用针对国家/ 地区的缩写。

以下示例对单元格 B2 进行格式化，以使数字显示三位小数，并用逗号作为千位分隔符。

```
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
```

StarSuite Calc 中的 **[**单元格属性**]** 对话框提供了用于格式化单元格的各种选项的摘要。

页面属性

页面属性是一些格式化选项，用于确定页面中的文档内容、每页重复出现的可视元素的位置。这些属性包括

- 纸张格式
- 页边距

- 页眉和页脚。

定义页面格式的过程与其他格式化形式不同。单元格、段落和字符元素可以直接定义并采用，而页面格式也可以使用页面样式定义，但不能直接采用。例如，将页眉和页脚加入到页面样式中。

下面各节介绍了用于工作表页面的主要格式化选项。这里介绍的许多样式同样适用于文本文档。对两种文档都有效的页面属性是在 `com.sun.star.style.PageProperties` 服务中定义。仅用于工作表文档的页面属性则是在 `com.sun.star.sheet.TablePageStyle` 服务中定义的。

Microsoft Office 文档的页面属性 (页边距、边框等) 是使用 `PageSetup` 对象在 `Worksheet` 对象级 (Excel) 或 `Document` 对象级 (Word) 定义的。
在 StarSuite 中，这些属性是使用页面样式来定义的，而这些页面样式又链接到相关的文档。

页面背景

`com.sun.star.style.PageProperties` 服务定义了以下页面背景属性：

- **`BackColor` (长整数)** - 背景颜色
- **`BackGraphicURL` (字串)** - 要使用的背景图形的 URL
- **`BackGraphicFilter` (字串)** - 用来解释背景图形的筛选的名称
- **`BackGraphicLocation` (枚举)** - 背景图形的位置 (值由 `com.sun.star.style.GraphicLocation` 枚举确定)
- **`BackTransparent` (布尔)** - 使背景透明

页面格式

页面格式是使用 `com.sun.star.style.PageProperties` 服务的以下属性定义的：

- **`IsLandscape` (布尔)** - 横向格式
- **`Width` (长整数)** - 页面宽度，以百分之一毫米为单位
- **`Height` (长整数)** - 页面高度，以百分之一毫米为单位
- **`PrinterPaperTray` (字串)** - 要使用的打印机纸张来源的名称

以下示例将“标准”页面样式的页面大小设定为 DIN A5 横向格式 (高 14.8 厘米，宽 21 厘米)：

```
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
```

页边距、边框和阴影

com.sun.star.style.PageProperties 服务提供了以下属性，用于调整页边距、边框和阴影：

- **LeftMargin (长整数)** - 页面左边距的宽度，以百分之一毫米为单位
- **RightMargin (长整数)** - 页面右边距的宽度，以百分之一毫米为单位
- **TopMargin (长整数)** - 页面上边距的宽度，以百分之一毫米为单位
- **BottomMargin (长整数)** - 页面下边距的宽度，以百分之一毫米为单位
- **LeftBorder (结构)** - 页面边框左侧线条的规范 (com.sun.star.table.BorderLine 结构)
- **RightBorder (结构)** - 页面边框右侧线条的规范 (com.sun.star.table.BorderLine 结构)
- **TopBorder (结构)** - 页面边框上方线条的规范 (com.sun.star.table.BorderLine 结构)
- **BottomBorder (结构)** - 页面边框下方线条的规范 > (com.sun.star.table.BorderLine 结构)
- **LeftBorderDistance (长整数)** - 页面左边框与页面内容之间的间隔，以百分之一毫米为单位
- **RightBorderDistance (长整数)** - 页面右边框与页面内容之间的间隔，以百分之一毫米为单位
- **TopBorderDistance (长整数)** - 页面上边框与页面内容之间的间隔，以百分之一毫米为单位
- **BottomBorderDistance (长整数)** - 页面下边框与页面内容之间的间隔，以百分之一毫米为单位
- **ShadowFormat (结构)** - 页面中内容区域的阴影规范 (com.sun.star.table.ShadowFormat 结构)

以下示例将“标准”页面样式的左、右边距设定为 1 厘米。

```
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
```

页眉和页脚

文档的页眉和页脚是页面属性的一部分，使用 `com.sun.star.style.PageProperties` 服务定义。用于格式化页眉的属性包括：

- **HeaderIsOn (布尔)** - 已启动页眉
- **HeaderLeftMargin (长整数)** - 页眉与左侧页边距之间的间隔，以百分之一毫米为单位
- **HeaderRightMargin (长整数)** - 页眉与右侧页边距之间的间隔，以百分之一毫米为单位
- **HeaderBodyDistance (长整数)** - 页眉与文档正文之间的间隔，以百分之一毫米为单位
- **HeaderHeight (长整数)** - 页眉的高度，以百分之一毫米为单位
- **HeaderIsDynamicHeight (布尔)** - 页眉高度根据内容自动调整
- **HeaderLeftBorder (结构)** - 页眉框左边框的细节 (`com.sun.star.table.BorderLine` 结构)
- **HeaderRightBorder (结构)** - 页眉框右边框的细节 (`com.sun.star.table.BorderLine` 结构)
- **HeaderTopBorder (结构)** - 页眉框上边框的细节 (`com.sun.star.table.BorderLine` 结构)
- **HeaderBottomBorder (结构)** - 页眉框下边框的细节 (`com.sun.star.table.BorderLine` 结构)
- **HeaderLeftBorderDistance (长整数)** - 页眉内容与左边框之间的间隔，以百分之一毫米为单位
- **HeaderRightBorderDistance (长整数)** - 页眉内容与右边框之间的间隔，以百分之一毫米为单位
- **HeaderTopBorderDistance (长整数)** - 页眉内容与上边框之间的间隔，以百分之一毫米为单位
- **HeaderBottomBorderDistance (长整数)** - 页眉内容与下边框之间的间隔，以百分之一毫米为单位
- **HeaderIsShared (布尔)** - 奇数页和偶数页的页眉内容相同 (请参阅 `HeaderText`、`HeaderTextLeft` 和 `HeaderTextRight`)
- **HeaderBackColor (长整数)** - 页眉的背景颜色
- **HeaderBackGraphicURL (字串)** - 要使用的背景图形的 URL
- **HeaderBackGraphicFilter (字串)** - 用于解释页眉背景图形的筛选的名称
- **HeaderBackGraphicLocation (枚举)** - 页眉背景图形的位置 (值由 `com.sun.star.style.GraphicLocation` 枚举确定)

- **HeaderBackTransparent (布尔)** - 将页眉背景显示为透明
- **HeaderShadowFormat (结构)** - 页眉阴影的细节 (com.sun.star.table.ShadowFormat 结构)

用于格式化页脚的属性包括：

- **FooterIsOn (布尔)** - 已启动页脚
- **FooterLeftMargin (长整数)** - 页脚与左侧页边距之间的间隔，以百分之一毫米为单位
- **FooterRightMargin (长整数)** - 页脚与右侧页边距之间的间隔，以百分之一毫米为单位
- **FooterBodyDistance (长整数)** - 页脚与文档正文之间的间隔，以百分之一毫米为单位
- **FooterHeight (长整数)** - 页脚的高度，以百分之一毫米为单位
- **FooterIsDynamicHeight (布尔)** - 页脚高度根据内容自动调整
- **FooterLeftBorder (结构)** - 页脚框左边框的细节 (com.sun.star.table.BorderLine 结构)
- **FooterRightBorder (结构)** - 页脚框右边框的细节 (com.sun.star.table.BorderLine 结构)
- **FooterTopBorder (结构)** - 页脚框上边框的细节 (com.sun.star.table.BorderLine 结构)
- **FooterBottomBorder (结构)** - 页脚框下边框的细节 (com.sun.star.table.BorderLine 结构)
- **FooterLeftBorderDistance (长整数)** - 页脚内容与左边框之间的间隔，以百分之一毫米为单位
- **FooterRightBorderDistance (长整数)** - 页脚内容与右边框之间的间隔，以百分之一毫米为单位
- **FooterTopBorderDistance (长整数)** - 页脚内容与上边框之间的距离，以百分之一毫米为单位
- **FooterBottomBorderDistance (长整数)** - 页脚内容与下边框之间的间隔，以百分之一毫米为单位
- **FooterIsShared (布尔)** - 奇数页和偶数页的页脚内容相同 (请参阅 FooterText、FooterTextLeft 和 FooterTextRight)
- **FooterBackColor (长整数)** - 页脚的背景颜色
- **FooterBackGraphicURL (字串)** - 要使用的页脚背景图形的 URL
- **FooterBackGraphicFilter (字串)** - 用于解释页脚背景图形的筛选的名称
- **FooterBackGraphicLocation (枚举)** - 页脚背景图形的位置 (值由 com.sun.star.style.GraphicLocation 枚举确定)
- **FooterBackTransparent (布尔)** - 将页脚背景显示为透明

- **FooterShadowFormat (结构)** - 页脚阴影的细节 (com.sun.star.table.ShadowFormat 结构)

更改页眉和页脚的文字

可以通过以下属性访问工作表中页眉和页脚的内容：

- **LeftPageHeaderContent (**对象**)** - 偶数页中页眉的内容 (com.sun.star.sheet.HeaderFooterContent 服务)
- **RightPageHeaderContent (**对象**)** - 奇数页中页眉的内容 (com.sun.star.sheet.HeaderFooterContent 服务)
- **LeftPageFooterContent (**对象**)** - 偶数页中页脚的内容 (com.sun.star.sheet.HeaderFooterContent 服务)
- **RightPageFooterContent (**对象**)** - 奇数页中页脚的内容 (com.sun.star.sheet.HeaderFooterContent 服务)

如果不希望区分奇数页和偶数页的页眉或页脚 (FooterIsShared 属性为 False)，则请设定奇数页中页眉和页脚的属性。

以上介绍的所有对象都将返回一个支持 com.sun.star.sheet.HeaderFooterContent 服务的对象。通过 (非真) 属性 LeftText、CenterText 和 RightText，此服务为 StarSuite Calc 的页眉和页脚提供了三种文字元素。

以下示例在“标准”文档样式中页眉的左侧文字字段内写入了一个“Just a Test.”值。

```
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
```

请注意示例的最后一行：如果更改了文字，则必须再次将 TextContent 对象指定给页眉，这样更改才能生效。

对于文本文档 (StarSuite Writer)，还可以使用另一种机制来更改页眉和页脚的文字，因为这些页眉和页脚内容是由一个单独的文字块构成的。以下属性是在 com.sun.star.style.PageProperties 服务中定义的：

- **HeaderText (对象)** - 包含页眉内容的文字对象 (com.sun.star.text.XText 服务)

- **HeaderTextLeft (对象)** - 包含左侧页面中页眉内容的文字对象 (com.sun.star.text.XText 服务)
- **HeaderTextRight (对象)** - 包含右侧页面中页眉内容的文字对象 (com.sun.star.text.XText 服务)
- **FooterText (对象)** - 包含页脚内容的文字对象 (com.sun.star.text.XText 服务)
- **FooterTextLeft (对象)** - 包含左侧页面中页脚内容的文字对象 (com.sun.star.text.XText 服务)
- **FooterTextRight (对象)** - 包含右侧页面中页脚内容的对象 (com.sun.star.text.XText 服务)

以下示例在文本文档的“标准”页面样式中建立了一个页眉，并将文字“Just a Test.”加入到页眉中。

```
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."
```

在这个示例中，可以通过页面样式的 `HeaderText` 属性直接进行访问，而不必使用 `HeaderFooterContent` 对象。

居中 (仅适用于工作表)

`com.sun.star.sheet.TablePageStyle` 服务仅用于 StarSuite Calc 页面样式，它允许您将要打印的单元格区域居中放置在页面上。该服务提供了以下属性：

- **CenterHorizontally (布尔)** - 表格内容水平居中
- **CenterVertically (布尔)** - 表格内容垂直居中

定义要打印的元素 (仅适用于工作表)

格式化工作表时，可以定义页面元素是否可见。为此，`com.sun.star.sheet.TablePageStyle` 服务提供了以下属性：

- **PrintAnnotations (布尔)** - 打印单元格的备注
- **PrintGrid (布尔)** - 打印单元格的网格线
- **PrintHeaders (布尔)** - 打印行标题和列标题
- **PrintCharts (布尔)** - 打印工作表中含有的图表
- **PrintObjects (布尔)** - 打印嵌入的对象
- **PrintDrawing (布尔)** - 打印绘图对象
- **PrintDownFirst (布尔)** - 如果工作表的内容分布在多个页面上，则首先按垂直方向从上向下打印，然后再打印右边的页面
- **PrintFormulas (布尔)** - 打印公式而不打印计算得到的值
- **PrintZeroValues (布尔)** - 打印零值

高效地编辑工作表文档

前面已经介绍了工作表文档的主体结构，本节将为您介绍一些服务，利用这些服务可以轻松地访问各个单元格或单元格区域。

单元格区域

StarSuite 不但提供了用于单个单元格的对象 (com.sun.star.table.Cell 服务)，而且还提供了表示单元格区域的对象。通过调用工作表对象的 getCellRangeByName 可以建立这样的 CellRange 对象：

```
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")
```

在工作表文档中使用冒号 (:) 来指定单元格区域。例如，A1:C15 表示 1 到 15 行、A 到 C 列的所有单元格。

可以使用 getCellByPosition 方法确定单元格区域中各个单元格的位置，其中，区域左上角的单元格的坐标是 (0,0)。以下示例使用该方法建立了单元格 C3 的对象。

```
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)
```

格式化单元格区域

与单个单元格一样，您可以使用 com.sun.star.table.CellProperties 服务来格式化单元格区域。如果需要有关该服务的更多信息和示例，请参阅格式化一节。

使用单元格区域进行计算

您可以使用 computeFunction 方法对单元格区域执行数学运算。computeFunction 要使用一个常数作为参数，以说明要使用的数学函数。相关的常数是在 com.sun.star.sheet.GeneralFunction 枚举中定义的。可以使用以下值：

- **SUM** - 所有数字值的和
- **COUNT** - 所有值的总个数 (包括非数字值)
- **COUNTNUMS** - 所有数字值的总个数
- **AVERAGE** - 所有数字值的平均值

- **MAX** - 最大的数字值
- **MIN** - 最小的数字值
- **PRODUCT** - 所有数字值的乘积
- **STDEV** - 标准偏差
- **VAR** - 方差
- **STDEVP** - 基于总体样本的标准偏差
- **VARP** - 基于总体样本的方差

以下示例计算了 A1:C3 区域的平均值，并在一个消息框中显示结果：

```
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)
```

删除单元格内容

`clearContents` 方法简化了删除单元格和单元格区域的内容的过程，因为它只从单元格区域中删除特定类型的内容。

以下示例从 B2:C3 区域中删除了所有字串和直接格式化信息。

```
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` 中指定的标记来自 `com.sun.star.sheet.CellFlags` 常数列单。此列单提供了以下元素：

- **VALUE** - 未格式化为日期或时间的数字值
- **DATETIME** - 格式化为日期或时间的数字值
- **STRING** - 字串

- **ANNOTATION** - 链接到单元格的注解
- **FORMULA** - 公式
- **HARDATTR** - 单元格的直接格式化
- **STYLES** - 单元格的间接格式化
- **OBJECTS** - 连接到单元格的绘图对象
- **EDITATTR** - 只适用于部分单元格的字符格式

您还可以将常数相加，以调用 `clearContents` 来删除不同的信息。

搜寻和替换单元格内容

与文本文档一样，工作表文档也提供了用于搜寻和替换的功能。

在工作表文档中，用于搜寻和替换的描述符对象不是通过文档对象直接建立的，而是通过 `Sheets` 列单建立的。以下示例介绍了一个搜寻和替换过程：

```
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
```

此示例使用文档的第一页建立一个 `ReplaceDescriptor`，然后通过循环将其用于所有的页面。

第 **8** 章

绘图和演示文稿

本章简要介绍了如何通过宏建立和编辑绘图。第一节介绍绘图的结构，包括构成绘图的基本元素。第二节进一步介绍相对复杂的编辑功能，如对象的分组、旋转和调整显示比例。

如果需要有关建立、打开和保存绘图的信息，请参阅第 5 章使用 *StarSuite* 文档。

绘图的结构

StarSuite 对绘图文档的页数没有限制。用户可以分别设计每个页面。而且可以加入到每一页的绘图元素的数量也没有限制。

由于使用了分层，才使情况稍微复杂了一些。默认情况下，每个绘图文档都含有“版式”、“控件”和“定量线”分层，所有的绘图元素都会加入到“版式”分层中。也可以选择加入新的分层。如果需要有关绘图分层的更多信息，请参阅《StarSuite 开发者指南》。

页面

`DrawPages` 列单提供了绘图文档的所有页面。可以通过编号或名称来访问每个页面。如果一个文档只有一个名为 *Slide 1* 的页面，则以下两个示例完全相同。

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，页面由编号 (从 0 开始计算) 定位。在示例 2 中，页面由名称和 `getByName` 方法定位。

```
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
```

以上调用会返回一个支持 `com.sun.star.drawing.DrawPage` 服务的页面对象。该服务能够识别以下属性：

- **BorderLeft (长整数)** - 左边框，以百分之一毫米为单位
- **BorderRight (长整数)** - 右边框，以百分之一毫米为单位
- **BorderTop (长整数)** - 上边框，以百分之一毫米为单位
- **BorderBottom (长整数)** - 下边框，以百分之一毫米为单位
- **Width (长整数)** - 页面宽度，以百分之一毫米为单位
- **Height (长整数)** - 页面高度，以百分之一毫米为单位
- **Number (短整数)** - 页码 (从 1 开始编号)，只读
- **Orientation (枚举)** - 页面方向 (与 `com.sun.star.view.PaperOrientation` 枚举一致)

如果更改了这些设定，则文档中的所有页面都会受到影响。

以下示例会将刚才打开的绘图文档的页面大小设定为 20 × 20 cm，页边距设定为 0.5 cm：

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

Page.BorderLeft = 500
Page.BorderRight = 500
Page.BorderTop = 500
Page.BorderBottom = 500

Page.Width = 20000
Page.Height = 20000
```

绘图对象的基本属性

绘图对象包括形状对象 (矩形、圆等)、线条对象和文字对象。所有这些对象都有一些共同的特征，并且都支持 `com.sun.star.drawing.Shape` 服务。此服务用于定义绘图对象的 `Size` 和 `Position` 属性。

StarSuite Basic 还提供了一些其他的服务，用于修改格式或采用充填等属性。格式选项是否可用取决于绘图对象的类型。

以下示例会在绘图文档中建立并插入一个矩形：

```
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` 调用来确定要打开的文档。使用这种方法确定的文档对象可以通过 `drawPages(0)` 调用返回绘图的第一页。

接着对表时绘图对象的原点 (左角点) 和大小的 `Point` 和 `Size` 结构进行了初始化。指定长度时以百分之一毫米为单位。

然后，此程序代码使用 Doc.createInstance 调用来建立由 com.sun.star.drawing.RectangleShape 服务指定的矩形绘图对象。最后，使用 Page.add 调用将绘图对象指定到页面。

充填属性

本节介绍了四种服务，每种服务的示例程序代码都使用了一个含有多种格式的矩形元素。充填属性包含在 com.sun.star.drawing.FillProperties 服务中。

对于一个充填区域，StarSuite 能够识别四种主要格式。最简单的格式是单色充填。使用那些用于定义彩色图案和阴影线的选项可建立其他颜色。第四种格式是将已有图形投影到充填区域。

绘图对象的充填模式由 FillStyle 属性定义。有效值在 com.sun.star.drawing.FillStyle 中定义。

单色充填

单色充填的主要属性为：

- **FillColor (长整数)** - 区域的充填色。

要使用充填模式，必须将 FillStyle 属性设定为 SOLID 充填模式。

以下示例会建立一个矩形，并且用红色进行充填 (RGB 值为 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)
```

彩色图案

如果将 `FillStyle` 属性设定为 `GRADIENT`，就可以在 StarSuite 文档的任意充填区域中采用彩色图案。

如果要采用预设的彩色图案，只需指定与 `FillTransparenceGradientName` 属性关联的名称即可。要定义自己的彩色图案，需要创建一个完整的 `com.sun.star.awt.Gradient` 结构，并将其赋值给 `FillGradient` 属性。此属性提供了以下选项：

- **`Style` (枚举)** - 彩色图案类型，例如，线性或放射状 (默认值与 `com.sun.star.awt.GradientStyle` 一致)
- **StartColor (**长整数**)** - 彩色图案的开始颜色
- **EndColor (**长整数**)** - 彩色图案的结束颜色
- **Angle (**短整数**)** - 彩色图案的角度，以十分之一度为单位
- **XOffset (**短整数**)** - 彩色图案的起点 X 坐标，以百分之一毫米为单位
- **YOffset (**短整数**)** - 彩色图案的起点 Y 坐标，以百分之一毫米为单位
- **StartIntensity (**短整数**)** - `StartColor` 的亮度，以百分比表示 (在 StarSuite Basic 中，也可以指定大于 100% 的值)
- **EndIntensity (**短整数**)** - `EndColor` 的亮度，以百分比表示 (在 StarSuite Basic 中，也可以指定大于 100% 的值)
- **StepCount (**短整数**)** - StarSuite 用来计算彩色图案的彩色等级数值。

以下示例通过 com.sun.star.awt.Gradient 结构来说明彩色图案的用法：

```
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)
```

此示例建立了一个线性彩色图案 (= LINEAR)。此彩色图案在左上角由红色 (StartColor) 开始，按 45 度角 (Angle) 延伸到右下角变为绿色 (EndColor)。开始和结束颜色的亮度都为 150% (StartIntensity 和 EndIntensity)，所生成的颜色看起来好象比在 StartColor 和 EndColor 属性中指定的值要亮一些。本彩色图案的渐变过程使用了一百种颜色 (StepCount)。

阴影线

要建立阴影线充填，必须将 FillStyle 属性设定为 HATCH。用于定义阴影线的程序代码与定义彩色图案的代码十分相似。同样是辅助结构，此处使用 com.sun.star.drawing.Hatch 来定义阴影线的外观。阴影线结构具有以下属性：

- **Style (枚举)** - 阴影线的类型：简单、方形或有对角线的方形 (默认值与 com.sun.star.awt.HatchStyle 一致)
- **Color (长整数)** - 线条的颜色
- **Distance (长整数)** - 线条间的距离，以百分之一毫米为单位
- **Angle (短整数)** - 阴影线的角度，以十分之一度为单位

以下示例说明了阴影线结构的用法：

```
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)
```

此代码建立了一个简单阴影线结构 (`HatchStyle` = `SINGLE`)，其线条呈 45 度旋转 (`Angle`)。线条为暗灰色 (`Color`)，其间距为 0.2 毫米 (`Distance`)。

位图

要使用位图投影进行充填，必须将 `FillStyle` 属性设定为 `BITMAP`。如果 StarSuite 中已有需要的位图，则只需在 `FillBitMapName` 属性中指定此位图的名字以及在 `FillBitmapMode` 属性中指定显示样式即可，例如简单、平铺或拉伸 (默认值与 `com.sun.star.drawing.BitmapMode` 一致)。

如果要使用外部的位图文件，可以在 `FillBitmapURL` 属性中指定其 URL。

以下示例会建立一个矩形，并通过平铺 StarSuite 中已有的 Sky 位图对矩形区域进行充填。

```
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

可以调整充填的透明度。要改变绘图元素的透明度，最简单的方法就是使用 `FillTransparence` 属性

以下示例会建立一个透明度为 50% 的红色矩形。

```
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)
```

要使充填透明，需将 `FillTransparence` 属性的值设定为 100。

除了 `FillTransparence` 属性以外，`com.sun.star.drawing.FillProperties` 服务还可提供 `FillTransparenceGradient` 属性。此属性用来定义用于指定充填区域透明度的彩色图案。

线条属性

所有具有边框线的绘图对象均支持 `com.sun.star.drawing.LineStyle` 服务。以下是此服务支持的一部分属性：

- **`LineStyle (枚举)`** - 线条类型 (默认值与 `com.sun.star.drawing.LineStyle` 一致)
- **`LineColor (长整数)`** - 线条颜色
- **`LineTransparence (短整数)`** - 线条透明度
- **`LineWidth (长整数)`** - 线条宽度，以百分之一毫米为单位
- **`LineJoint (枚举)`** - 转换至连接点 (默认值与 `com.sun.star.drawing.LineJoint` 一致)

以下示例会建立一个宽度为 5 毫米 (LineWidth)，透明度为 50% 的实体边框矩形 (LineStyle = SOLID)。线条的左右边延长并相交 (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)
```

除以上列出的属性外，com.sun.star.drawing.LineStyle 服务还提供绘制点划线和短划线的选项。如果需要更多的信息，请参阅 StarSuite API 参考。

文字属性 (绘图对象)

com.sun.star.style.CharacterProperties 和 com.sun.star.style.ParagraphProperties 服务可以格式化绘图对象中的文字。如果需要有关与这些字符和段落相关的服务的更多信息，请参阅第 6 章文本文档。

以下示例会在矩形中插入文字，并格式化 com.sun.star.style.CharacterProperties 服务的字体。

```
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"
```

此代码使用矩形的 String 属性来插入文字，使用 com.sun.star.style.CharacterProperties 服务的 CharWeight 和 CharFontName 属性来格式化文字字体。

只有将绘图对象加入到绘图页面中后，才能插入文字。也可以使用 com.sun.star.drawing.Text 服务来定位并格式化绘图对象中的文字。以下为此服务的部分重要属性：

- TextAutoGrowHeight (布尔) - 将绘图元素的高度调整为与它所含有的文字相符
- **TextAutoGrowWidth (布尔)** - 将绘图元素的宽度调整为与它所含有的文字相符
- **TextHorizontalAdjust (枚举)** - 绘图元素中文字的水平位置 (默认值与 com.sun.star.drawing.TextHorizontalAdjust 一致)
- **TextVerticalAdjust (枚举)** - 绘图元素中文字的垂直位置 (默认值与 com.sun.star.drawing.TextVerticalAdjust 一致)
- **TextLeftDistance (长整数)** - 绘图元素左侧与文字左侧之间的距离，以百分之一毫米为单位
- **TextRightDistance (长整数)** - 绘图元素右侧与文字右侧之间的距离，以百分之一毫米为单位
- TextUpperDistance (长整数) - 绘图元素顶部与文字顶部之间的距离，以百分之一毫米为单位
- **TextLowerDistance (长整数)** - 绘图元素底部与文字底部之间的距离，以百分之一毫米为单位

以下示例说明了命名属性的使用方法。

```
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"  ' 只能在运行 Page.add 之后运行!

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
```

此代码将一个绘图元素插入到页面中，然后使用 `TextVerticalAdjust` 和 `TextHorizontalAdjust` 属性将文字加入到绘图对象的左上角处。绘图对象的文字边缘间的最小距离设定为 3 毫米。

阴影属性

可以使用 `com.sun.star.drawing.ShadowProperties` 服务为大多数绘图对象加入阴影。此服务具有以下属性：

- **`Shadow (布尔)`** - 启动阴影
- **`ShadowColor (长整数)`** - 阴影颜色
- **`ShadowTransparence (短整数)`** - 阴影的透明度
- **`ShadowXDistance (长整数)`** - 阴影与绘图对象之间的垂直距离，以百分之一毫米为单位
- **`ShadowYDistance (长整数)`** - 阴影与绘图对象之间的水平距离，以百分之一毫米为单位

以下示例会建立一个带阴影的矩形，阴影在矩形的水平方向和垂直方向各偏移 2 毫米。阴影渲染为深灰色，透明度为 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)
```

各种绘图对象概览

矩形

矩形对象 (com.sun.star.drawing.RectangleShape) 支持以下服务以便格式化对象：

- 充填属性 - com.sun.star.drawing.FillProperties
- 线条属性 - com.sun.star.drawing.LineProperties
- 文字属性 - com.sun.star.drawing.Text (包括 com.sun.star.style.CharacterProperties 和 com.sun.star.style.ParagraphProperties)
- 阴影属性 - com.sun.star.drawing.ShadowProperties
- **CornerRadius (**长整数**)** - 圆角半径，以百分之一毫米为单位

圆和椭圆

服务 com.sun.star.drawing.EllipseShape 负责圆和椭圆，它所支持的服务如下：

- 充填属性 - com.sun.star.drawing.FillProperties
- 线条属性 - com.sun.star.drawing.LineProperties
- 文字属性 - com.sun.star.drawing.Text (包括 com.sun.star.style.CharacterProperties 和 com.sun.star.style.ParagraphProperties)

- 阴影属性 - com.sun.star.drawing.ShadowProperties

除上述服务外，圆和椭圆还提供下列属性：

- **CircleKind (枚举)** - 圆或椭圆的类型 (默认值与 com.sun.star.drawing.CircleKind 一致)
- **CircleStartAngle (长整数)** - 起始角度，以十分之一度为单位 (仅用于圆或椭圆段)
- **CircleEndAngle (长整数)** - 结束角度，以十分之一度为单位 (仅用于圆或椭圆段)

CircleKind 属性确定对象是整圆、圆缺或圆段。可提供以下值：

- **com.sun.star.drawing.CircleKind.FULL** - 完整的圆或椭圆
- **com.sun.star.drawing.CircleKind.CUT** - 圆段 (其接合部位直接相互链接的部分圆)
- **com.sun.star.drawing.CircleKind.SECTION** - 圆缺
- **com.sun.star.drawing.CircleKind.ARC** - 角度 (不包括圆线条)

以下示例会建立一个 70 度的圆缺 (起始角度为 20 度，结束角度为 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)
```

线条

StarSuite 为线条对象提供了 `com.sun.star.drawing.LineShape` 服务。线条对象支持除面积外的所有通用格式化服务。以下是与 `LineShape` 服务有关的所有属性：

- 线条属性 - `com.sun.star.drawing.LineProperties`
- 文字属性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)
- 阴影属性 - `com.sun.star.drawing.ShadowProperties`

以下示例借助命名属性建立一个线条并进行格式化。线条的起点在 `Location` 属性中指定，而 `Size` 属性中列出的坐标指定了线条的终点。

```
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)
```

多重多边形

StarSuite 也可以支持复杂的多边形，只需采用 com.sun.star.drawing.PolyPolygonShape 服务。严格地讲，多重多边形不是简单的多边形，而是由多个多边形组成的多边形。因此，可以指定并组合包含角点的多个独立列单以构成一个完整的对象。

就像矩形一样，所有绘图对象的格式化属性也可以供多重多边形使用：

- 充填属性 - com.sun.star.drawing.FillProperties
- 线条属性 - com.sun.star.drawing.LineProperties
- _文字属性 - com.sun.star.drawing.Text (包括 com.sun.star.style.CharacterProperties 和 com.sun.star.style.ParagraphProperties)
- 阴影属性 - com.sun.star.drawing.ShadowProperties

PolyPolygonShape 服务还具有一个用于定义多边形坐标的属性：

- PolyPolygon (数组) - 含有多边形坐标的字段 (具有类型为 com.sun.star.awt.Point 的点的双精度数组)

以下示例显示了用 PolyPolygonShape 服务定义三角形的方法。

```
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)   ' 设定坐标前，必须先运行 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())
```

由于多边形的点已定义为绝对数值，所以不需要指定多边形的大小或起点位置。而是需要建立一个点数组，并将此数组包含到第二个数组中 (使用 Array(Coordinates() 调用)，然后将此数组赋值给多边形。执行相应的调用前，必须先将多边形插入到文档中。

双精度数组的定义允许通过合并多边形来建立复杂的形状。例如，用户可以建立一个矩形，然后在其中插入另一个矩形，以便在第一个矩形中建立一个孔：

```
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)   ' 设定坐标前，必须先运行 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())
```

对于区分哪些区域为充填区域及哪些区域为孔，StarSuite 采用了一条简单的原则：外部形状的边始终为多重多边形的边界。向内的下一条线是此形状的内边框，表示转换到第一个孔。如果内部还有其他线条，则表示转换到充填区域。

图形

此处讲述的最后一个绘图元素是基于 `com.sun.star.drawing.GraphicObjectShape` 服务的图形对象。这些对象可用于 StarSuite 的任意图形中，而且可以使用各种属性对它们的外观进行调整。

图形对象支持的通用格式化属性有两种：

- 文字属性 - `com.sun.star.drawing.Text` (包括 `com.sun.star.style.CharacterProperties` 和 `com.sun.star.style.ParagraphProperties`)
- 阴影属性 - `com.sun.star.drawing.ShadowProperties`

图形对象还支持一些其他的属性：

- **GraphicURL (字串)** - 图形的 URL
- **AdjustLuminance (短整数)** - 色彩的亮度，以百分比表示 (允许使用负数)
- **AdjustContrast (短整数)** - 对比度，以百分比表示 (允许使用负数)
- **AdjustRed (短整数)** - 红色的值，以百分比表示 (允许使用负数)
- **AdjustGreen (短整数)** - 绿色的值，以百分比表示 (允许使用负数)
- **AdjustBlue (短整数)** - 蓝色的值，以百分比表示 (允许使用负数)
- **Gamma (短整数)** - 图形的灰色系数
- **Transparency (短整数)** - 图形的透明度，以百分比表示
- **GraphicColorMode (枚举)** - 颜色模式，例如，标准、灰度、黑白 (默认值与 `com.sun.star.drawing.ColorMode` 一致)

以下示例显示了如何将页面插入到图形对象中。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                        ' 规范，没有实际意义，因为后面的坐标已绑定

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)
```

此代码插入了 `test.jpg` 图形，并使用 `Adjust` 属性调整其外观。在本例中，图形的透明度为 40%，未发生其他色彩转换 (`GraphicColorMode` = `STANDARD`)。

编辑绘图对象

分组对象

在许多情况下，可以将多个图形对象组合起来作为一个大的对象进行处理，此操作非常有用。

以下示例将两个绘图对象组合为一个对象：

```
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
' 建立正方形绘图元素
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' 建立圆形绘图元素
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)
' 将正方形和圆组合起来
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' 使组合的绘图元素居中
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
```

此代码建立了一个矩形和一个圆，然后将它们插入到页面中。接着，又建立了一个支持 com.sun.star.drawing.ShapeCollection 服务的对象，并使用 Add 方法将矩形和圆加入到此对象中。使用 Group 方法将 ShapeCollection 加入到页面后，将返回可以像编辑单个 Shape 对象那样编辑的实际的 Group 对象。

如果要格式化对象组中的单个对象，请在将其加入组之前进行。对象一旦进入对象组，就不能编辑了。

旋转和修剪绘图对象

以上介绍的所有绘图对象都可以使用 com.sun.star.drawing.RotationDescriptor 服务进行旋转和修剪。

此服务提供以下属性：

- **RotateAngle (长整数)** - 旋转角度，以百分之一度为单位
- **ShearAngle (长整数)** - 修剪角度，以百分之一度为单位

以下示例会建立一个矩形，然后使用 RotateAngle 属性将其旋转 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)
```

接下来的这个示例也建立一个和前例一样的矩形，但是使用 ShearAngle 属性将其修剪 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)
```

搜寻和替换

与文本文档一样，绘图文档也提供搜寻和替换功能。此功能与第 6 章文本文档中所介绍的文本文档的相应功能类似。 但是，在绘图文档中，用于搜寻和替换的描述符对象不是直接通过文本对象建立的，而是要通过相关的字符级来建立。以下示例演示了绘图中的替换过程：

```
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
```

此代码使用文档的第一个 DrawPage 来建立一个 ReplaceDescriptor，然后通过循环将此描述符采用到绘图文档的所有页面中。

演示文稿

StarSuite 的演示文稿是基于绘图文档建立的。每一页演示文稿就是一张幻灯片。可以用访问标准绘图对象的方法来访问这些幻灯片，具体方法为使用文档对象的 `DrawPages` 列单。`com.sun.star.presentation.PresentationDocument` 服务负责演示文稿的文档，而且还能提供完整的 `com.sun.star.drawing.DrawingDocument` 服务。

使用演示文稿

除 `Presentation` 属性提供的绘图函数以外，每个演示文稿文档都具有一个演示文稿对象，用于访问主要属性及演示控制机制。例如，此对象可提供一个 `start` 方法，用来启动演示文稿。

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

此示例中使用的代码建立了一个引用当前演示文档的 `Doc` 对象，此对象也用于建立相关的演示文稿对象。对象的 `start()` 方法用于启动示例及运行屏幕演示文稿。

为演示文稿的对象提供了以下方法：

- **start** - 启动演示文稿
- **end** - 结束演示文稿
- **rehearseTimings** - 从头启动演示文稿并建立其运行时

也可以使用以下属性：

- **AllowAnimations (布尔)** - 在演示文稿中运行动画
- **CustomShow (字串)** - 用于指定演示文稿的名称，以便在演示中引用此名称
- **FirstPage (字串)** - 用于启动演示文稿的幻灯片的名称
- **IsAlwaysOnTop (布尔)** - 演示文稿窗口始终在屏幕的最前显示
- **IsAutomatic (布尔)** - 自动运行整个演示文稿
- **IsEndless (布尔)** - 结束后自动从头开始运行演示文稿
- **IsFullScreen (布尔)** - 自动以全屏幕模式启动演示文稿
- **IsMouseVisible (布尔)** - 演示过程中显示鼠标
- **Pause (长整数)** - 演示结束时黑屏的显示时间
- **StartWithNavigator (布尔)** - 演示开始时显示 [助手] 视窗
- **UsePn (布尔)** - 演示过程中显示指针

第 **9** 章

图表

StarSuite 可以将数据显示为图表，通过条形、饼形、折线或其他元素等形式，在图形与数据之间建立链接。数据可以显示为 2 维或 3 维图形，并可以对图表元素的外观分别进行调整，方法与绘图元素所采用的过程类似。

如果数据是以工作表的形式提供的，则可以动态链接至图表。在这种情况下，对基础数据所做的更改能够立即反映在相应的图表中。本章概要介绍了 StarSuite 图表模块的编程接口，着重说明了图表在工作表文档中图表的使用。

在工作表文档中使用图表

在 StarSuite 中，图表不是被视为独立的文档，而是视为嵌入现有文档中的对象。

文本文档和绘图文档中的图表与文档内容是孤立的，而工作表中提供了一种能够将文档数据与嵌入的图表链接起来的机制。下面的示例解释了工作表文档和图表之间的交互：

```
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)
```

尽管示例中使用的代码看起来有点复杂，但是主要过程只有三行：第一行建立 Doc 文档变量，它引用当前的工作表文档 (Doc line = StarDesktop.CurrentComponent)。然后，示例中所使用的代码建立一个含有第一个工作表的所有图表的列单 (Charts line = Doc.Sheets(0).Charts)。最后，使用 addNewByName 方法向此列单的最后一行添加一个新图表。这时，使用者就可以看到新图表了。

最后一行初始化 Rect 和 RangeAddress 辅助结构，而 addNewByName 用作其中一个参数。Rect 确定图表在工作表中的位置。RangeAddress 确定要链接到图表的数据区域。

上一个示例建立了一个条形图。如果需要不同类型的图形，必须显式替换条形图：

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

第一行定义相应的图表对象。第二行将当前图表替换为新图表 - 在本示例中替换为折线图。

在 Excel 中，作为独立页插入到 Excel 文档中的图表与嵌入到表格页中的图表是有区别的。因此，这两种图表的访问方法也不同。而在 StarSuite Calc 中并没有这种区别，因为 StarSuite Calc 中的图表始终是作为表格页的嵌入对象而建立的。始终使用相关 Sheet 对象的 Charts 列单来访问图表。

图表的结构

图表的结构 (及其支持的服务和接口的列单) 取决于其类型。例如，Z 轴的方法和属性只能在 3 维图表中使用，而不能在 2 维图表中使用。在饼图中，没有用于轴的接口。

图表的各个元素

标题、副标题和图例

标题、副标题和图例是构成图表基本元素的一部分。图表对各元素分别提供了相应的对象。为了管理这些元素，Chart 对象提供了以下属性：

- **HasMainTitle (布尔)** - 启动标题。
- **Title (对象)** - 含有图表标题详细信息的对象 (支持 com.sun.star.chart.ChartTitle 服务)。
- **HasSubTitle(布尔)** - 启动副标题。
- **Subtitle (对象)** - 含有图表副标题详细信息的对象 (支持 com.sun.star.chart.ChartTitle 服务)。
- **HasLegend (布尔)** - 启动图例。
- **Legend (对象)** - 含有图表图例详细信息的对象 (支持 com.sun.star.chart.ChartLegendPosition 服务)。

指定的元素在许多方面都与绘图元素相对应。这是因为 com.sun.star.chart.ChartTitle 服务和 com.sun.star.chart.ChartLegendPosition 都支持构成绘图元素技术程序基础的 com.sun.star.drawing.Shape 服务。

因此，使用者可以使用 Size 和 Position 属性来确定元素的位置和大小。

还提供了用于格式化元素的充填和线条属性 (com.sun.star.drawing.FillProperties 和 com.sun.star.drawing.LineStyle 服务) 及字符属性 (com.sun.star.style.CharacterProperties 服务)。

com.sun.star.chart.ChartTitle 不仅含有已命名的格式属性，还有另外两种属性：

- **TextRotation (长整数)** - 文字的旋转角度，以百分之一度为单位。
- **String (字串)** - 显示为标题或副标题的文字。

图表图例 (com.sun.star.chart.ChartLegend 服务) 具有以下附加属性：

- **Alignment (枚举)** - 图例出现的位置 (默认值与 com.sun.star.chart.ChartLegendPosition 一致)。

以下示例建立一个图表，并为其指定标题“Test”、副标题“Test 2”和图例。图例的背景为灰色，位于图表底部，字符大小为 7 点。

```
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

每个图表都具有一个背景区域。每个区域都有一个对象，可以使用图表对象的以下属性来访问：

- **Area (对象)** - 图表的背景区域 (支持 `com.sun.star.chart.ChartArea` 服务)。

图表的背景覆盖所有区域，包括标题、副标题和图表图例以下的区域。相应的 `com.sun.star.chart.ChartArea` 服务支持线条和充填属性，但不再提供其他属性。

图表背景墙和基底

图表背景覆盖图表的整个区域，而图表背景墙仅限于数据区域正后方的区域。

3 维图表通常具有两个图表背景墙：一个位于数据区域后方，另一个位于 Y 轴左侧。3 维图表通常还具有基底。

- **Floor (对象)** - 图表的基底 (仅限于 3 维图表，支持 com.sun.star.chart.ChartArea 服务)。
- **Wall (对象)** - 图表背景墙 (仅限于 3 维图表，支持 com.sun.star.chart.ChartArea 服务)。

指定的对象支持 com.sun.star.chart.ChartArea 服务，而该服务提供了常见的充填和线条属性 (com.sun.star.drawing.FillProperties 和 com.sun.star.drawing.LineStyle 服务，请参阅第 8 章)。

图表背景墙和基底的访问是通过 Chart 对象来实现的，而该对象又是 Chart 对象的一部分：

```
Chart.Area.FillBitmapName = "Sky"
```

以下示例显示了 StarSuite 中已有的图形 (名为 Sky) 是如何被用作图表背景的。

```
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
```

轴

StarSuite 能够识别可以在图表中使用的五种不同的轴。在最简单的方案中，只有 X 轴和 Y 轴。当使用 3 维图表时，有时还需要 Z 轴。如果图表中不同数据行的值相差很大，StarSuite 还提供了第二条 X 轴和 Y 轴，以便进行第二次设定显示比例的操作。

第一条 X 轴、Y 轴和 Z 轴

除了实际的轴以外，第一条 X 轴、Y 轴和 Z 轴也可以带有标题、说明、网格及辅助网格。可以选择显示或隐藏所有这些元素。为了管理这些功能，图表对象提供了以下属性 (假设 X 轴、Y 轴和 Z 轴的属性结构相同)：

- **HasXAxis (布尔)** - 启动 X 轴。
- **XAxis (对象)** - 含有 X 轴详细信息的对象 (支持 com.sun.star.chart.ChartAxis 服务)。
- **HasXAxisDescription (布尔)** - 启动 X 轴说明。
- **HasXAxisGrid (布尔)** - 启动 X 轴主网格。
- **XMainGrid (对象)** - 含有 X 轴主网格详细信息的对象 (支持 com.sun.star.chart.ChartGrid 服务)。
- **HasXAxisHelpGrid (布尔)** - 启动 X 轴辅助网格。
- **XHelpGrid (对象)** - 含有 X 轴辅助网格详细信息的对象 (支持 com.sun.star.chart.ChartGrid 服务)。
- **HasXAxisTitle (布尔)** - 启动 X 轴的标题。
- **XAxisTitle (对象)** - 含有 X 轴标题详细信息的对象 (支持 com.sun.star.chart.ChartTitle 服务)。

第二条 X 轴和 Y 轴

第二条 X 轴和 Y 轴具有以下属性 (以第二条 X 轴的属性为例)：

- **HasSecondaryXAxis (布尔)** - 启动第二条 X 轴。
- **SecondaryXAxis (对象)** - 含有第二条 X 轴详细信息的对象 (支持 com.sun.star.chart.ChartAxis 服务)。
- **HasSecondaryXAxisDescription (布尔)** - 启动 X 轴的说明。

轴的属性

StarSuite 图表的轴对象支持 com.sun.star.chart.ChartAxis 服务。除了字符 (com.sun.star.style.CharacterProperties 服务，请参阅第 6 章) 和线条 (com.sun.star.drawing.LineStyle 服务，请参阅第 8 章) 的属性以外，还提供了以下属性：

- **Max (双精度)** - 轴的最大值。
- **Min (双精度)** - 轴的最小值。

- **Origin (双精度)** - 相交轴的交点。
- **StepMain (双精度)** - 轴的两条主刻度线之间的间隔。
- **StepHelp (双精度)** - 轴的两条次刻度线之间的间隔。
- **AutoMax (布尔)** - 自动确定轴的最大值。
- **AutoMin (布尔)** - 自动确定轴的最小值。
- **AutoOrigin (布尔)** - 自动确定相交轴的交点。
- **AutoStepMain (布尔)** - 自动确定轴的主刻度线之间的间隔。
- **AutoStepHelp (布尔)** - 自动确定轴的次刻度线之间的间隔。
- **Logarithmic (布尔)** - 以对数方式 (而不是线性方式) 调整轴的显示比例。
- **DisplayLabels (布尔)** - 启动轴的文字标签。
- **TextRotation (长整数)** - 轴的文字标签的旋转角度，以百分之一度为单位。
- **Marks (常数)** - 指定轴的主刻度线是位于图表区域之内还是之外的常数 (默认值与 `com.sun.star.chart.ChartAxisMarks` 一致)
- **HelpMarks (常数)** - 指定轴的次刻度线是位于图表区域之内还是之外的常数 (默认值与 `com.sun.star.chart.ChartAxisMarks` 一致)
- **Overlap (长整数)** - 指定不同数据集的条形相互覆盖程度的百分比 (100% 表示完全重叠，-100% 表示中间有相当于一个条形宽度的间隔)。
- **GapWidth (长整数)** - 指定图表中不同条形组之间间隔的百分比 (100% 表示有一个条形的宽度的间隔)。
- **ArrangeOrder (枚举)** - 标签详细位置，除定位刻度位置选项外，还可以选择每隔两个刻度分隔一次标签 (默认值由 `com.sun.star.chart.ChartAxisArrangeOrderType` 决定)
- **TextBreak (布尔)** - 允许换行。
- **TextCanOverlap (布尔)** - 允许文字重叠。
- **NumberFormat (长整数)** - 数字格式 (请参阅第 7 数字、日期和文字格式一节)

轴网格的属性

轴的网格对象基于 `com.sun.star.chart.ChartGrid` 服务，而该服务支持 `com.sun.star.drawing.LineStyle` 支持服务的线条属性 (请参阅第 8 章)。

轴标题的属性

格式化轴标题的对象基于 `com.sun.star.chart.ChartTitle` 服务，该服务还用于图表标题。

示例

下面的示例建立一个折线图。图表背景墙的颜色设置为白色。X 轴和 Y 轴都有灰色辅助网格，用于识别方向。Y 轴的最小值设定为 0，最大值为 100，所以即便数值发生了变化，图表的分辨率仍然保持不变。

```
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
```

3 维图表

StarSuite 中的大多数图表也可以使用 3 维图形来显示。提供此选项的所有图表类型都支持 `com.sun.star.chart.Dim3DDiagram` 服务。该项服务仅提供一个属性：

- **`Dim3D (布尔)`** - 启动 3 维显示

重叠图表

重叠图表是将几个单值堆积起来形成总值的一种图表。这种视图不仅可以显示单值，还可以显示所有值的概览。

在 StarSuite 中，很多类型的图表都能够以重叠的形式显示。这些图表都支持 com.sun.star.chart.StackableDiagram 服务，而该服务提供以下属性：

- **Stacked (布尔)** - 启动重叠视图模式。
- **Percent (布尔)** - 不显示绝对数值，而是显示相应的百分比分布。

图表类型

折线图

折线图 (服务 com.sun.star.chart.LineDiagram) 支持一个 X 轴、两个 Y 轴和一个 Z 轴。可以显示为 2 维或 3 维图形 (com.sun.star.chart.Dim3Ddiagram 服务)。线条可以重叠 (com.sun.star.chart.StackableDiagram)。

折线图的属性包括：

- **SymbolType (常数)** - 显示数据点的符号 (常数与 com.sun.star.chart.ChartSymbolType 一致)。
- **SymbolSize (长整数)** - 显示数据点的符号的大小，以百分之一毫米为单位。
- **SymbolBitmapURL (字串)** - 显示数据点的图形的文件名。
- **Lines (布尔)** - 通过线条来链接数据点。
- **SplineType (长整数)** - 使线条平滑的样条函数 (0：无样条函数，1：三次样条，2：B 样条)。
- **SplineOrder (长整数)** - 样条的多项式权数 (仅用于 B 样条)。
- **SplineResolution (长整数)** - 样条计算需要的支持点数。

面积图

面积图 (com.sun.star.chart.AreaDiagram 服务) 支持一个 X 轴，两个 Y 轴和一个 Z 轴。它们可以显示为 2 维或 3 维图形 (com.sun.star.chart.Dim3Ddiagram 服务)。面积可以重叠 (com.sun.star.chart.StackableDiagram)。

条形图

条形图 (服务 com.sun.star.chart.BarDiagram) 支持一个 X 轴，两个 Y 轴和一个 Z 轴。可以显示为 2 维或 3 维图形 (com.sun.star.chart.Dim3Ddiagram 服务)。条形可以重叠 (com.sun.star.chart.StackableDiagram)。

条形图的属性包括：

- **Vertical (布尔)** - 以垂直方式显示条形，否则以水平方式显示。

- **Deep (布尔)** - 在 3 维视图模式中，将条形堆叠显示，而不是并排显示。
- **StackedBarsConnected (布尔)** - 通过线条将重叠图表中的相应条形链接起来 (仅用于水平图表)。
- **NumberOfLines (**长整数**)** - 重叠图表中显示为线条 (而不是条形) 的线条数。

饼图

饼图 (`com.sun.star.chart.PieDiagram` 服务) 不含轴，也不能重叠。可以显示为 2 维或 3 维图形 (`com.sun.star.chart.Dim3Ddiagram` 服务)。

第 10 章

访问数据库

StarSuite 具有一个称为 Star 数据库连接 (SDBC) 的集成数据库接口 (独立于任何系统)。开发本接口的目的是为了尽可能多地访问不同的数据源。

为了实现此目的，数据源是由驱动程序访问的。由驱动程序从中获取数据的数据源与 SDBC 使用者无关。有些驱动程序访问基于文件的数据库并直接从中获取数据。其他驱动程序则使用 JDBC 或 ODBC 等标准接口。但是，还有一些特殊驱动程序，它们将 MAPI 地址簿、LDAP 目录或 StarSuite 工作表用作数据源。

由于驱动程序基于 UNO 组件，因此，可以开发其他驱动程序来打开新的数据源。如果需要更多信息，请参阅《StarSuite 开发者指南》。

从概念上讲，SDBC 相当于 VBA 中的 ADO 和 DAO 库。它允许对数据库进行高级访问，而不考虑底层数据库后端。

自从使用 StarSuite 6 以来，StarSuite 的数据库接口逐步得到了发展。过去主要通过 `Application` 对象的一些方法来访问数据库，而 StarSuite 6 的接口被细分为几个对象。`DatabaseContext` 用作数据库函数的根对象。

SQL：一种查询语言

SQL 语言是供 SDBC 使用者使用的一种查询语言。为了比较不同的 SQL 语言之间的区别，StarSuite 中的 SDBC 组件具有自己的 SQL 语言分析器。它使用查询视窗来检查输入的 SQL 命令，并纠正简单的语法错误，如有关字符大小写的错误等。

如果驱动程序允许访问不支持 SQL 的数据源，则必须单独将传送的 SQL 命令转化为所需的本地访问。

从 SDBC 实现 SQL 是为了适应 SQL-ANSI 标准的要求。不支持 Microsoft 专用扩展 (如 `INNER JOIN` 结构)。需要用标准命令进行替换 (例如，`INNER JOIN` 应使用 `WHERE` 子句进行替换)。

数据库的访问类型

StarSuite Writer 和 StarSuite Calc 应用程序和数据库表单中提供了 StarSuite 数据库接口。

在 StarSuite Writer 中，可以借助 SDBC 数据源建立标准信件，并且可以打印出来。还可以选择使用拖放功能将数据从数据库视窗中移到文本文档中。

如果使用者将数据库表格移动到工作表中，StarSuite 将建立一个表格区域，原始数据更改后，只要单击一下鼠标就可以更新该区域。反过来，工作表数据也可以移到数据库表格中，这时执行的是数据库输入。

最后，StarSuite 提供了一种基于数据库的表单机制。为此，使用者首先要建立一个标准 StarSuite Writer 或 StarSuite Calc 表单，然后将字段链接到数据库。

此处指定的所有选项全部基于 StarSuite 使用者界面。使用这些函数不需要编程知识。

但本章很少涉及有关指定函数的信息，而主要讨论 SDBC 的编程接口。SDBC 允许数据库自动查询，因此允许使用更为广泛的应用程序。

但是，要完全理解下面的章节，还需要了解有关数据库的工作方式以及 SQL 查询语言的基本知识。

数据源

通过建立通常所说的数据源将数据库加入到 StarSuite 中。在 **[Extras]** 菜单中，使用者界面提供了用于建立数据源的选项。但也可以使用 StarSuite Basic 建立和使用数据源。

使用 `createUnoService` 函数建立的数据库上下文对象用作访问数据源的起始点。它基于 `com.sun.star.sdb.DatabaseContext` 服务，并且是所有数据库操作的根对象。

下面的示例显示了数据库上下文的建立方法，以及如何使用数据库上下文来确定所有可用数据源的名称。名称显示在消息框中。

```
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
```

单个数据源基于 `com.sun.star.sdb.DataSource` 服务，可以使用 `getByName` 方法通过数据库上下文来确定：

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
```

该示例为名为 *Customers* 的数据源建立一个 `DataSource` 对象。

数据源提供了许多属性，而这些属性又提供了有关数据来源和访问方法的一般信息。这些属性包括：

- **Name (字串)** - 数据源名称。

- **URL (字串)** - 格式为 *jdbc: subprotocol : subname* 或 *sdbc: subprotocol : subname* 的数据源 URL。
- **Info (Array)** - array containing `PropertyValue`-pairs with connection parameters (usually at least user name and password).**Info (数组)** - 含有带有连接参数 (通常至少包含使用名和密码) 的 `PropertyValue` 对的数组。
- **User (字串)** - 使用名。
- **Password (字串)** - 使用者密码 (不保存)。
- **IsPasswordRequired (布尔)** - 需要密码，且密码是由使用者交互请求的。
- **IsReadOnly (布尔)** - 允许只读访问数据库。
- **NumberFormatsSupplier (对象)** - 含有数据库中可用的数字格式的对象 (支持 `com.sun.star.util.XNumberFormatsSupplier` 接口，请参阅第 7 章中的数字、日期和文字格式一节。
- **TableFilter (数组)** - 要显示的表格名称列单。
- **TableTypeFilter (数组)** - 要显示的表格类型列单。可用的数值是 `TABLE`、`VIEW` 和 `SYSTEM TABLE`。
- **SuppressVersionColumns (布尔)** - 不显示用于版本管理的列。

StarSuite 的数据源与 ODBC 中的数据源不完全一致。ODBC 数据源仅包含有关数据来源的信息，而 StarSuite 中的数据源还包含有关数据如何在 StarSuite 的数据库视窗中显示的信息。

查询

可以为数据源指定预设的查询。StarSuite 记录 SQL 的查询命令，这样就可以随时使用这些命令。查询可以用于简化数据库操作，因为只需单击一下鼠标就可以打开查询，查询还为不了解 SQL 的使用者提供了 SQL 命令的选项。

支持 `com.sun.star.sdb.QueryDefinition` 服务的对象隐藏在查询之内。查询是通过数据源的 `QueryDefinitions` 方法来访问的。

以下示例列出了可以在消息框中建立的数据源查询的名称。

```
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
```

除示例所使用的 Name 属性外，com.sun.star.sdb.QueryDefinition 还提供了许多其他属性。它们是：

- **Name (字串)** - 查询名称。
- **Command (字串)** - SQL 命令 (通常是 SELECT 命令)。
- **UpdateTableName (字串)** - 基于多个表格的查询：表格内的数值有可能被更改的表格名称。
- **UpdateCatalogName (字串)** - 更新表格目录的名称。
- **UpdateSchemaName (字串)** - 更新表格图的名称。

以下示例显示了如何通过编程控制方式来建立查询对象并将其指定到数据源。

```
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` 调用建立查询对象，然后将查询对象初始化，并通过 `insertByName` 插入到 `QueryDefinitions` 对象中。

与数据库表单链接

为了简化数据源的使用，StarSuite 提供了将数据源与数据库表单链接起来的选项。链接是通过 `getBookmarks()` 方法来实现的。这会返回一个已命名的容器 (`com.sun.star.sdb.DefinitionContainer`)，其中包含所有数据源链接。可以通过**名称**或**索引**来访问书签。

以下示例确定书签 *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
```

数据库访问

要访问数据库，首先需要连接到数据库。这是一种允许与数据库直接通讯的传送通道。与上一节提到的数据源不同，每次重新启动程序时都必须重新建立数据库连接。

StarSuite 提供了多种建立数据库连接的方法。下面基于现有数据源对此方法进行说明。

```
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
```

示例中的代码首先检查数据库是否采用密码保护。如果没有密码保护，则使用 `GetConnection` 调用建立所需的数据库连接。命令行中的两个空字串表示使用名和密码。

如果数据库采用了密码保护，则建立一个 `InteractionHandler`，然后使用 `ConnectWithCompletion` 方法打开数据库连接。InteractionHandler 确保 StarSuite 要求使用者提供所需的登录数据。

表格迭代

StarSuite 中，通常使用 `ResultSet` 对象来访问表格。`ResultSet` 是一种标记类型，表示使用 `SELECT` 命令得到的结果卷中的当前数据集。

此示例显示了如何使用 ResultSet 在数据库表中查询数值。

```
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
```

建立了数据库连接后，示例中的代码首先使用 Connection.createObject 调用建立一个 Statement 对象。然后，此 Statement 对象使用 executeQuery 调用返回实际的 ResultSet。这时程序将检查是否实际存在 ResultSet，并使用循环来遍历数据条目。所需要的数据 (本示例中为 CustomerNumber 字段中的数据) 使用 getString 方法返回 ResultSet，参数 1 借此确定有关第一列数值的调用。

SDBC 中的 ResultSet 对象也提供对数据库的迭代访问，因此它相当于 DAO 和 ADO 中的 Recordset 对象。

实际上，在 StarSuite 6.x 中是通过 ResultSet 对象访问数据库的。它反映了表格的内容或 SQL-SELECT 命令的结果。过去，ResultSet 对象在 Application 对象中提供了用于浏览数据 (例如 DataNextRecord) 的驻留方法。

用于检索数值的特定类型的方法

从上一节的示例中可以看出，StarSuite 提供了一个 getString 方法，用于访问表格内容。此方法以字串的形式提供了结果。可以使用以下 get 方法：

- getByte() - 支持数字、字符和字串的 SQL 数据类型。

- `getShort()` - 支持数字、字符和字串的 SQL 数据类型。
- `getInt()` - 支持数字、字符和字串的 SQL 数据类型。
- `getLong()` - 支持数字、字符和字串的 SQL 数据类型。
- `getFloat()` - 支持数字、字符和字串的 SQL 数据类型。
- `getDouble()` - 支持数字、字符和字串的 SQL 数据类型。
- `getBoolean()` - 支持数字、字符和字串的 SQL 数据类型。
- `getString()` - 支持所有 SQL 数据类型。
- `getBytes()` - 支持二进制值的 SQL 数据类型。
- `getDate()` - 支持数字、字串、日期和时间戳的 SQL 数据类型。
- `getTime()` - 支持数字、字串、日期和时间戳的 SQL 数据类型。
- `getTimestamp()` - 支持数字、字串、日期和时间戳的 SQL 数据类型。
- `getCharacterStream()` - 支持数字、字串和二进值的 SQL 数据类型。
- `getUnicodeStream()` - 支持数字、字串和二进值的 SQL 数据类型。
- `getBinaryStream()` - 二进制值。
- `getObject()` - 支持所有 SQL 数据类型。

无论哪种情况，都应当将列数作为需要对其值进行查询的参数而列出。

ResultSet 变量

访问数据库的关键问题是速度的快慢。因此，StarSuite 提供了几种用来优化 `ResultSets` 的方法，从而控制访问的速度。`ResultSet` 提供的函数越多，执行起来通常就会越复杂，函数的速度也就越慢。

如“表格迭代”小节所述，简单的 `ResultSet` 提供尽可能少的函数。它只允许向前重复，且只能用于将被询问的值。所以不包括其他浏览选项，如更改值的可能性等。

用于建立 ResultSet 的 Statement 对象提供了一些允许改变 ResultSet 函数的属性：

- **ResultSetConcurrency (常数)** - 有关数据能否更改的规范 (规范与 com.sun.star.sdbc.ResultSetConcurrency 一致)。
- **ResultSetType (常数)** - 有关 ResultSets 类型的规范 (规范与 com.sun.star.sdbc.ResultSetType 一致)。

com.sun.star.sdbc.ResultSetConcurrency 中定义的值包括：

- **UPDATABLE** - ResultSet 允许更改值。
- **READ_ONLY** - ResultSet 不允许更改。

com.sun.star.sdbc.ResultSetConcurrency 常数组提供了以下规范：

- **FORWARD_ONLY** - ResultSet 只允许向前浏览。
- **SCROLL_INSENSITIVE** - ResultSet 允许任意类型的浏览，但是不记录对原始数据的更改。
- **SCROLL_SENSITIVE** - ResultSet 允许任意类型的浏览，原始数据的更改将影响到 ResultSet。

ResultSet 包含 READ_ONLY 和 SCROLL_INSENSITIVE 属性，与 ADO 和 DAO 中的 Snapshot 类型的数据条目集相对应。

当使用 ResultSet 的 UPDATEABLE 属性和 SCROLL_SENSITIVE 属性时，ResultSet 函数的范围与 ADO 和 DAO 中 Dynaset 类型的 Recordset 相对应。

ResultSet 中的浏览方法

如果 ResultSet 属于 SCROLL_INSENSITIVE 或 SCROLL_SENSITIVE 类型，那么它可以支持许多种数据浏览方法。其中，最主要的方法是：

- **next()** - 浏览到下一个数据条目。
- **previous()** - 浏览到上一个数据条目。
- **first()** - 浏览到第一个数据条目。
- **last()** - 浏览到最后一个数据条目。
- **beforeFirst()** - 浏览到第一个数据条目之前。
- **afterLast()** - 浏览到最后一个数据条目之后。

所有方法均返回一个布尔参数，该参数说明浏览是否成功。

为了确定当前的光标位置，提供了以下测试方法，这些方法都将返回一个布尔值：

- **isBeforeFirst()** - ResultSet 位于第一个数据条目之前。
- **isAfterLast()** - ResultSet 位于最后一个数据条目之后。
- **isFirst()** - ResultSet 是第一个数据条目。
- **isLast()** - ResultSet 是最后一个数据条目。

更改数据条目

如果已经使用 ResultSetConcurrency = UPDATEABLE 值建立了 ResultSet，那么就可以对其内容进行编辑。只有当 SQL 命令允许数据重新写入数据库 (根据规定) 时才适用。例如，对于含有链接列或累积值的复杂 SQL 命令，就不适用。

ResultSet 对象提供了用于更改数值的 Update 方法，该方法的结构与用于检索数值的 get 方法相同。例如，updateString 方法允许写入字串。

数据更改后，必须使用 updateRow() 方法将这些数据传送到数据库中。必须在下一个浏览命令之前使用调用，否则这些值将会丢失。

如果在更改过程中发生错误，可以使用 cancelRowUpdates() 方法撤消。只有当未使用 updateRow() 将数据重新写入到数据库时，才可以使用此调用。

第 **11** 章

对话框

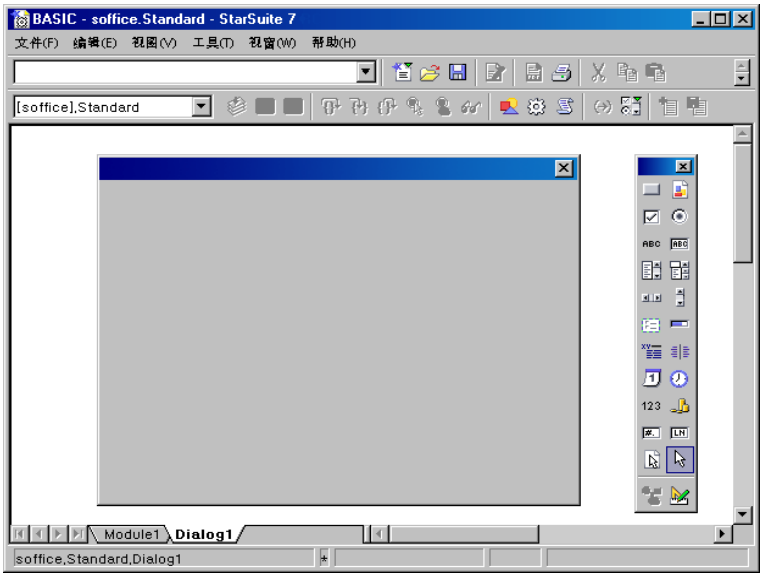
可以将自定义对话视窗和表单加入到 StarSuite 文档中。然后可以把这些视窗和表单链接到 StarSuite Basic 宏，从而极大地扩大 StarSuite Basic 的使用范围。例如，对话框可以显示数据库信息，或以自动文件助理的形式指导用户执行建立新文档的过程。

使用对话框

StarSuite Basic 的对话框由一个可以含有文本字段、列单框、单选按钮以及其他控制元素的对话视窗构成。

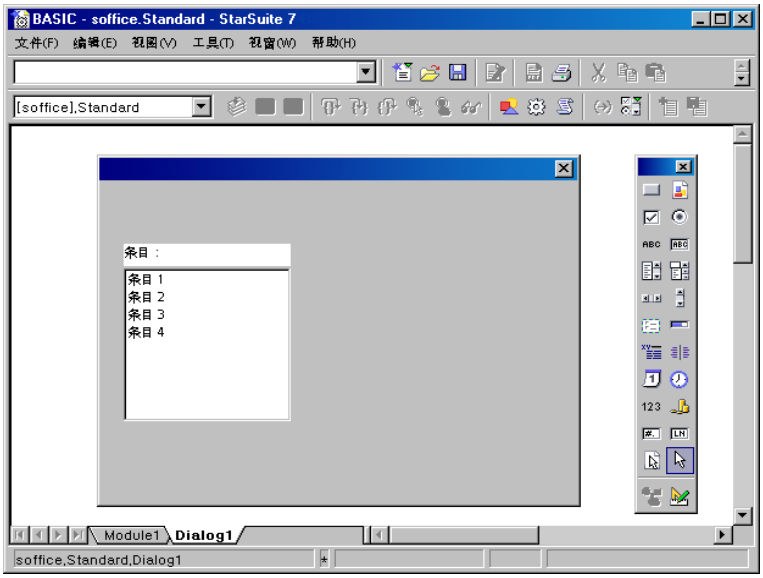
建立对话框

可以使用 StarSuite 对话框编辑器建立和构造对话框，其方法与使用 StarSuite Draw 的方法一样：



实际上，也就是将所需控制元素从设计面板 (右侧) 拖动到对话框区域中，此区域用于定义控制元素的位置和大小。

下面的示例显示了一个含有标签和列单框的对话框。



可以使用以下代码打开对话框：

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` 建立了一个名为 `Dlg` 的对象，该对象引用了一个关联的对话框。建立对话框之前，必须确保已加载对话框所使用的库 (在此示例中，为 `Standard` 库)。如果没有加载，`LoadLibrary` 方法就会执行此任务。

只要对 `Dlg` 对话框对象进行了初始化，就可以使用 `Execute` 方法显示此对话框。这一类的对话框都称为模态对话框，因为只有在关闭时才能对它们执行其他的编程操作。这类对话框打开时，程序会保持处于 `Execute` 调用状态。

一旦程序结束，最后一行代码中的 `dispose` 方法会批准对话框使用的资源。

关闭对话框

使用“确定”或“取消”进行关闭

如果对话框中含有“确定”或“取消”按钮，只需按其中的一个按钮，对话框就会自动关闭。本章的〈对话框控制元素详述〉一节将讨论使用这些按钮方面的更多信息。

如果单击“确定”按钮关闭对话框，`Execute` 方法会返回一个返回值 `1`，否则会返回值 `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
```

使用标题栏中的“关闭”按钮进行关闭

如果需要，可以通过单击对话视窗标题栏上的“关闭”按钮来关闭对话框。在此情况下，对话框的 `Execute` 方法会返回值 0，这与按“取消”按钮时返回的值相同。

使用显式程序调用进行关闭

也可以使用 `endExecute` 方法关闭打开的对话视窗：

```
Dlg.endExecute()
```

访问单个的控制元素

对话框中可以含有任意数目的控制元素。可以通过返回控制元素名称的 `getControl` 方法访问这些元素。

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

此代码首先确定了用作 `MyButton` 控制元素的对象，然后使用对元素的引用初始化 `Ctl` 对象变量。最后，代码将控制元素的 `Label` 属性值设定为 `New Label`。

请注意，对于控制元素，StarSuite Basic 将区分其名称中的大小写字符。

使用对话框和控制元素的模型

StarSuite API 中的许多地方都区分可见程序元素 (视图) 与它们背后的数据或元素 (模型)。除了具有控制元素的方法和属性之外，对话框和控制元素对象都具有一个下级**模型**对象。此对象用于直接访问对话框或控制元素的内容。

在对话框中与在其他 StarSuite API 区域中不一样，数据和描绘之间的区别并不总是非常地清楚。此 API 元素既可以为“视图”元素，也可以为“模型”元素。

`Model` 属性提供了一种由程序控制的访问方式，用于对对话框和控制元素对象的模型进行访问。

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

此示例借助来自 `cmdNtext` 的模型对象关闭了 `Dlg` 对话框中的 `cmdNtext` 按钮。

属性

名称和标题

每个控制元素都有自己的名称，这些名称可以使用以下模型属性查询：

- **`Model.Name (字串)`** - 控制元素的名称

可以通过以下模型属性指定要显示在对话框标题栏中的标题：

- **`Model.Title (字串)`** - 对话框的标题 (仅适用于对话框)。

位置和大小

可以使用模型对象的以下属性查询控制元素的大小和位置：

- **`Model.Height (长整数)`** - 控制元素的高度 (以 ma 为单位)
- **`Model.Width (长整数)`** - 控制元素的宽度 (以 ma 为单位)
- **`Model.PositionX (长整数)`** - 控制元素的 X 位置，从对话框左侧的内部边界算起 (以 ma 为单位)
- **`Model.PositionY (长整数)`** - 控制元素的 Y 位置，从对话框上方的内部边界算起 (以 ma 为单位)

为了确保对话框外观的平台独立性，StarSuite 使用 *Map AppFont (ma)* 内部单位来指定对话框内部的位置和大小。ma 单位的高度定义为操作系统中定义的系统字体字符平均高度的八分之一，宽度为其四分之一。通过使用 ma 单位，StarSuite 确保了其对话框在使用不同系统设置的不同系统上外观都一致。

如果要更改“运行时”的控制元素的大小或位置，请先确定此对话框的总大小，然后再将控制元素的值调整为相应的比率。

Map AppFont (ma) 替换缇单位，以获得更佳的平台独立性。

焦点和制表符序列

可以通过按 Tab 键在任意对话框中浏览所有的控制元素。在此情况下，可以在控制元素模型中使用以下属性：

- **Model.Enabled (布尔)** - 启动控制元素
- **Model.Tabstop (布尔)** - 允许通过 Tab 键访问控制元素
- **Model.TabIndex (长整数)** - 控制元素在启动顺序中的位置

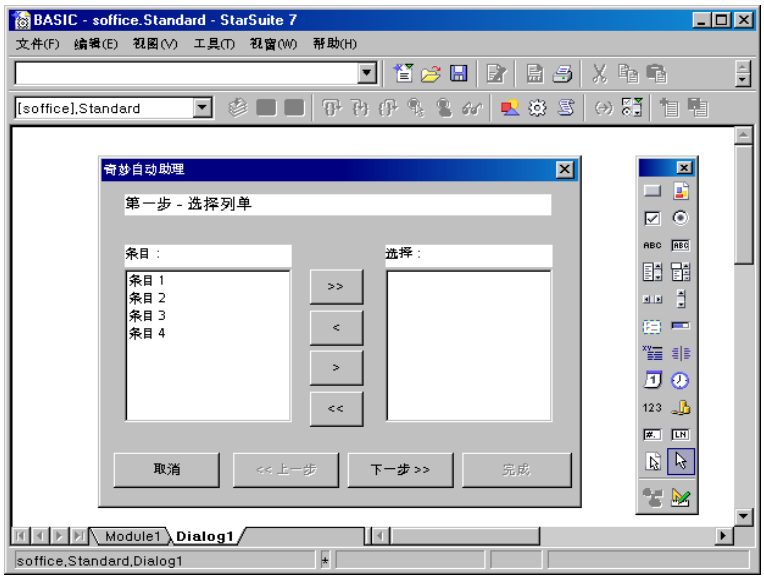
最后，控制元素会提供 getFocus 方法以确保基本的控制元素获得焦点：

- **getFocus** - 控制元素获得焦点 (仅适用于对话框)

多页对话框

StarSuite 中的对话框可以有多个选项卡页面。对话框的 Step 属性用于定义对话框的当前选项卡页面，而控制元素的 Step 属性则用于指定要显示此控制元素的选项卡页面。

Step 值为 0 时情况比较特殊。如果在对话框中将此值设置为零，就会显示所有控制元素，而不管它们具有什么 Step 属性值。同样地，如果将与控制元素对应的这一值设置为零，该元素就会在对话框的所有选项卡页面上显示。



在以上示例中，也可以将 Step 值 0 指定给分隔线以及**“取消”**、**“前一页”**、**“下一页”**和**“完成”**按钮，以便在所有页面上显示这些元素。还可以将这些元素指定到单独的选项卡页面 (例如，第 1 页)。

以下程序代码显示了如何增大或减小**“下一页”**和**“前一页”**按钮的事件处理程序中的 Step 值，以及如何更改这些按钮的状态。

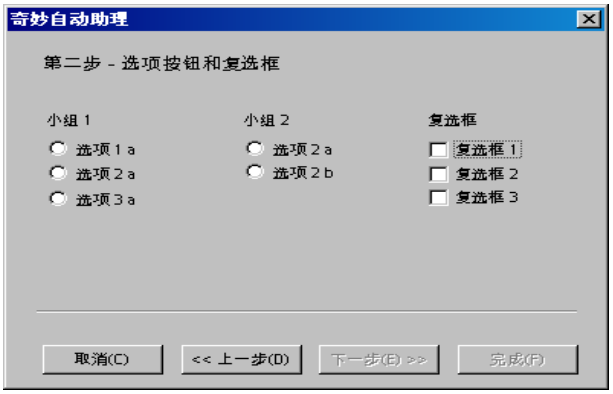
```
Sub cmdNext_Initiated
      Dim cmdNext As Object
      Dim cmdPrev As Object
```

```
      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
```

必须含有一个引用了打开的对话框的全局 `Dlg` 变量，此示例才可能实现。然后，对话框的外观会做出以下更改：

第 **1** 页：

第 **2** 页：



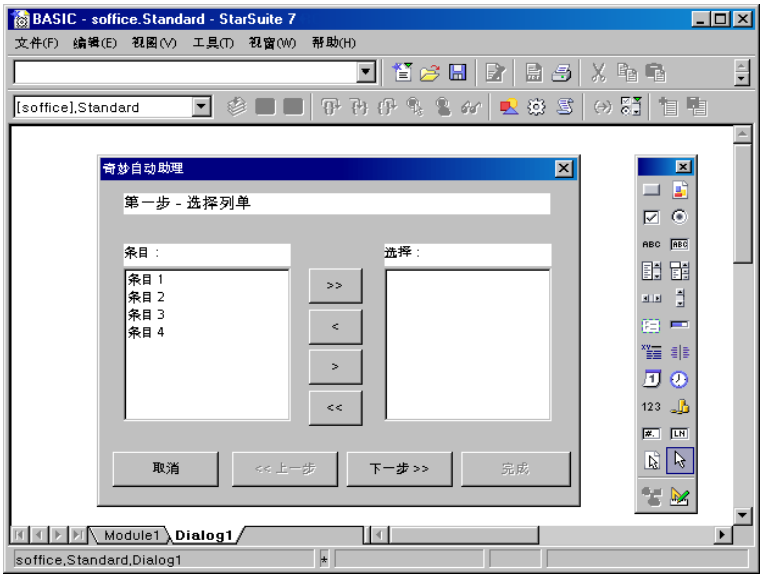
事件

StarSuite 对话框和表单都基于面向事件的编程模型。在这个模型中，可以把**事件处理程序**指定给控制元素。当发生特定的操作时，即使此操作为另一个事件，事件处理程序都会运行一个预设的过程。使用事件处理功能也可以编辑文档、打开数据库或访问其他控制元素。

StarSuite 控制元素能够识别在不同情形下触发的不同事件类型。这些事件类型可以分为四组：

- 鼠标控制：与鼠标操作相对应的事件 (例如，简单的鼠标移动或单击特定的屏幕位置)
- 键盘控制：由键盘击键触发的事件
- 焦点修改：启动或关闭控制元素时 StarSuite 执行的事件
- 控制元素特有的事件：仅在特定的控制元素上才会发生的事件

使用事件时，请确保已在 StarSuite 开发环境中建立了与之关联的对话框，并确保此对话框含有所需控制元素或文档 (如果要将此事件采用到表单中)。



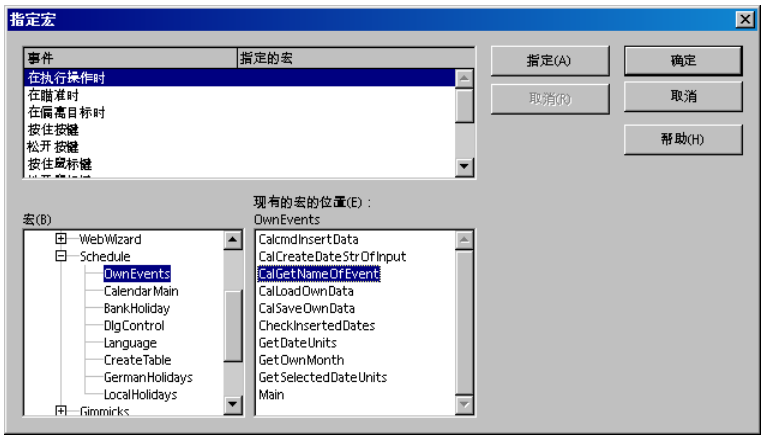
上图显示了在 StarSuite Basic 开发环境中打开的对话框，此对话框含有两个列单框。可以使用两个列单框之间的按钮将数据从一个列单移到另一个列单中。

如果要在屏幕上显示版式，则应建立与之关联的 StarSuite Basic 过程以便事件处理程序调用这些过程。尽管在任何模块中都可以使用这些过程，但最好限于仅供两个模块使用。为了使代码更易于读取，应为这些过程指定带含义的名称。直接从宏跳转到一般编程过程会导致代码不清。因此，为了简化代码维护和疑难解答，就应建立另一个过程，作为事件处理的入口点，即使它只执行对目标过程的单一调用。

以下示例中的代码会将一个条目从对话框的左列单框移到右列单框中。

```
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
```

如果此过程是在 StarSuite Basic 中建立的，就可以使用对话框编辑器的属性视窗将其指定给必需的事件。



指定对话框中列出了所有的 StarSuite Basic 过程。要将一个过程指定给某个事件，请选择该过程，然后单击“指定”。

参数

发生某个特定事件时并不总能获得相应的响应。可能还需要一些其他信息。例如，要进行一次鼠标单击，可能会需要按鼠标键时的屏幕位置。

在 StarSuite Basic 中，可以使用对象参数将一个事件的更多相关信息提供给一个过程，例如：

```
Sub ProcessEvent(Event As Object)

End Sub
```

构建 `Event` 对象时使用的精确度及其属性取决于过程调用触发的事件类型。随后的小节会详细介绍这些事件类型。

不管在何种类型的事件中，都可以访问所有对象的相关控制元素及其模型。要访问控制元素，可以使用

```
Event.Source
```

要访问模型，可以使用

```
Event.Source.Model
```

可以使用这些属性触发事件处理程序中的事件。

鼠标事件

StarSuite Basic 能够识别以下鼠标事件：

- **Mouse moved** - 用户移动了鼠标
- **Mouse moved while key pressed** - 用户按住一个按键并拖动鼠标
- **Mouse button pressed** - 用户按住一个鼠标键
- **Mouse button released** - 用户松开一个鼠标键
- **Mouse outside** - 用户将鼠标移到当前窗口以外

这些关联事件对象的结构由 `com.sun.star.awt.MouseEvent` 结构定义，此结构提供以下信息：

- **Buttons (短整数)** - 按下的按钮 (一个或多个常数与 `com.sun.star.awt.MouseButton` 一致)。
- **X (长整数)** - 从控制元素左上角算起的鼠标的 X 坐标，以像素为单位
- **Y (长整数)** - 从控制元素左上角算起的鼠标的 Y 坐标，以像素为单位
- **ClickCount (长整数)** - 与鼠标事件关联的点击次数 (如果 StarSuite 的响应速度足够快，由于只启动了一个事件，双击的 ClickCount 值也为 1)。

`com.sun.star.awt.MouseButton` 中定义的鼠标键常数为：

- **LEFT** - 鼠标左键
- **RIGHT** - 鼠标右键
- **MIDDLE** - 鼠标中键

以下示例将输出鼠标位置及按下的鼠标键：

```
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` 事件，并通过更改应用程序逻辑模拟 `Doubleclick` 事件。

键盘事件

StarSuite Basic 提供以下键盘事件：

- **`Key pressed`** - 用户按下了一个按键
- **`Key released`** - 用户松开了一个按键

这两个事件都与**逻辑**按键操作相关，而与**物理**操作无关。如果用户要按多个键以输出一个字符 (例如，为字符添加重音符号)，StarSuite Basic 只会建立一个事件。

对修改键 (如 Shift 键或 Alt 键) 执行的单键操作不会建立一个独立的事件。

StarSuite Basic 会将事件对象提供的有关按键的信息提供给过程，以便进行事件处理。它含有以下属性：

- **`KeyCode (短整数)`** - 按下的按键代码 (默认值与 `com.sun.star.awt.Key` 一致)
- **`KeyChar (字串)`** - 输入的字符 (包括修改键)

以下示例使用 KeyCode 属性确定是否按下了 Enter 键、Tab 键或其他控制按键之一。如果按下了上述按键之一，将返回按键的名称，否则将返回输入的字符：

```
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
```

如果需要有关其他键盘常数的信息，请参阅 com.sun.star.awt.Key 常数组下的“API 引用”。

焦点事件

焦点事件表明控制元素是获得还是失去焦点。例如，用户可以使用这些事件确定其他用户是否已处理完某个控制元素，以便更新对话框中的其他元素。可以使用以下焦点事件：

- **When receiving focus** - 元素获得焦点
- **When losing focus** - 元素失去焦点

焦点事件的 Event 对象的构成如下所示：

- **FocusFlags (短整数)** - 焦点变化的原因 (默认值与 com.sun.star.awt.FocusChangeReason 一致)。
- **NextFocus (对象)** - 获得焦点的对象 (仅适用于 When losing focus 事件)
- **Temporary (布尔)** - 暂时丢失焦点

控制元素特有的事件

除了上述所有控制元素都支持的事件之外，还有一些控制元素特有的事件，这些事件仅针对某些控制元素而定义。这些事件中最重要的事件为：

- **When Item Changed** - 控制元素的值发生变化
- **Item Status Changed** - 控制元素的状态发生变化
- **Text modified** - 控制元素的文本发生变化
- **When initiating** - 触发控制元素时可以执行的操作 (例如，按下一个按钮)

使用这些事件时，请注意，每次在一些特定的控制元素上 (例如，在单选按钮上) 单击鼠标时，都会启动某些事件，例如，`When initiating` 事件。此时并不会执行任何操作以检查控制元素的状态是否真的发生了变化。要避免这样的“盲目事件”，请将旧的控制元素值保存在一个全局变量中，然后检查执行事件时此值是否发生变化。

`Item Status Changed` 事件的属性如下：

- **Selected (长整数)** - 当前选定的条目
- **Highlighted (长整数)** - 当前突出显示的条目
- **ItemId (长整数)** - 条目的 ID

对话框控制元素详述

StarSuite Basic 能够识别一系列控制元素，这些元素分为以下几组：

条目字段：

- 文本字段
- 日期字段
- 时间字段
- 数字字段
- 货币字段
- 采用任何格式的字段

按钮：

- 标准按钮
- 复选框
- 单选按钮

选择列单：

- 列单框
- 组合框

其他控制元素：

- 滚动条 (水平和垂直)
- 组字段
- 进展方框
- 分隔线 (水平和垂直)
- 图形
- 文件选择字段

下面介绍这些控制元素中最重要的元素。

按钮

单击按钮时，它会执行一项操作。

最初级的方案就是使按钮在用户单击时触发 `When Initiating` 事件。也可以使用 `PushButtonType` 属性将另一个操作与按钮链接在一起，以便打开一个对话框。 如果单击的按钮此属性值设定为 0，对话框不受影响。如果单击的按钮此属性值设置为 1，对话框将关闭，并且对话框的 `Execute` 方法会返回值 1 (对话框序列已正确结束)。如果 `PushButtonType` 的值为 2，则关闭对话框，并且对话框的 `Execute` 方法会返回值 0 (对话框已关闭)。

下面是可通过按钮模型使用的所有属性：

- **`Model.BackgroundColor (长整数)`** - 背景颜色
- **`Model.DefaultButton (布尔)`** - 该按钮用作默认值，并当此按钮没有焦点时响应 Enter 键。
- **`Model.FontDescriptor (结构)`** - 用于指定要使用的字体细节的结构 (与 `com.sun.star.awt.FontDescriptor` 结构一致)
- **`Model.Label (字串)`** - 按钮上显示的标签
- **`Model.Printable (布尔)`** - 可以打印控制元素
- **`Model.TextColor (长整数)`** - 控制元素的文字颜色
- **`Model.HelpText (字串)`** - 将鼠标光标移到控制元素上方时显示的帮助文字
- **`Model.HelpURL (字串)`** - 与控制元素对应的联机帮助的 URL
- **`PushButtonType (短整数)`** - 与按钮链接的操作 (0：无操作；1：确定；2：取消)

选项字段

使用时，选项字段通常会编组，并且允许从中选择一个选项。当选择了一个选项后，组中的其他所有选项都会关闭。这可确保一次仅设定一个选项字段。

每个选项字段控制元素都提供这两种属性：

- **State (布尔)** - 启动按钮
- **Label (字串)** - 按钮上显示的标签

也可以使用选项字段模型中的以下属性：

- **Model.FontDescriptor (结构)** - 含有所使用字体的细节的结构 (与 `com.sun.star.awt.FontDescriptor` 一致)
- **Model.Label (字串)** - 控制元素上显示的标签
- **Model.Printable (布尔)** - 可以打印控制元素
- **Model.State (短整数)** - 如果此属性为 1，将启动该选项，否则关闭该选项
- **Model.TextColor (长整数)** - 控制元素的文字颜色
- **Model.HelpText (字串)** - 当鼠标光标置于控制元素上方时显示的帮助文字
- **Model.HelpURL (字串)** - 与控制元素对应的联机帮助的 URL

要将多个选项按钮组合到一个组中，必须无间隔地按照启动顺序一个接一个地定位这些选项字段 (`Model.TabIndex` 属性，在对话框编辑器中描述为“Order”)。如果其他控制元素中断了启动顺序，StarSuite 会自动启动一个可启动的新控制元素组，而不管第一个控制元素组。

与 VBA 不同的是，不能在 StarSuite Basic 中的控制元素组内插入选项字段。通过在控制元素的周围画框，StarSuite Basic 的控制元素分组仅用于确保可视分区。

复选框

复选框用于输入值“是”或“否”，根据所处的模式，它们可以调整为两种或三种状态。除了“是”和“否”状态之外，如果相应的“是”或“否”状态具有多重含义或含义不清，复选框可以具有一种**中间状态**。

复选框提供以下属性：

- **State (短整数)** - 复选框的状态 (0：否，1：是，2：中间状态)
- **Label (字串)** - 控制元素的标签
- **enableTriState (布尔)** - 除了启动或关闭状态之外，还可以使用中间状态

复选框的模型对象可提供以下属性：

- **Model.FontDescriptor (结构)** - 含有所使用字体的细节的结构 (与 `com.sun.star.awt.FontDescriptor` 结构一致)
- **Model.Label (字串)** - 控制元素的标签
- **Model.Printable (布尔)** - 可以打印控制元素
- **Model.State (短整数)** - 复选框的状态 (0：否，1：是，2：中间状态)
- **Model.Tabstop (布尔)** - 可通过 Tab 键访问控制元素
- **Model.TextColor (长整数)** - 控制元素的文字颜色
- **Model.HelpText (字串)** - 将鼠标光标置于控制元素上方时显示的帮助文字

- **Model.HelpURL (字串)** - 与控制元素对应的联机帮助的 URL

文本字段

文本字段用于输入数字和文本。com.sun.star.awt.UnoControlEdit. 服务为文本字段的基础。

文本字段可含有一行或多行文本，可以对文本字段进行编辑，或禁止用户输入。文本字段也可以用作特殊货币和数字字段以及用作特殊任务的屏幕字段。由于这些控制元素都基于 UnoControlEdit Uno 服务，用程序控制它们的处理方式非常相似。

文本字段提供以下属性：

- **Text (字串)** - 当前文本
- **SelectedText (字串)** - 当前突出显示的文本
- **Selection (结构)** - 以只读方式突出显示的细节 (结构与 com.sun.star.awt.Selection 一致，通过 Min 和 Max 属性指定当前突出显示的文本的开头和结尾)
- **MaxTextLen (短整数)** - 可以在字段中输入的最大字符数
- **Editable (布尔)** - 值为 True 时启动输入文本的选项，值为 False 时禁止输入选项 (不能直接调用此属性，只能通过 IsEditable 进行调用)
- **IsEditable (布尔)** - 可以更改控制元素的内容，只读。

另外，通过与之关联的模型对象还提供了以下属性：

- **Model.Align (短整数)** - 文本方向 (0：左对齐，1：居中，2：右对齐)
- **Model.BackgroundColor (长整数)** - 控制元素的背景颜色
- **Model.Border (短整数)** - 边框类型 (0：无边框，1：3 维边框，2：简单边框)
- **Model.EchoChar (字串)** - 密码字段的回显字符
- **Model.FontDescriptor (结构)** - 含有所使用字体的细节的结构 (与 com.sun.star.awt.FontDescriptor 结构一致)
- **Model.HardLineBreaks (布尔)** - 在控制元素文本中永久插入自动换行
- **Model.HScroll (布尔)** -
- **Model.MaxTextLen (短整数)** - 最大文本长度，其中，0 对应于无长度限制
- **Model.MultiLine (布尔)** - 允许条目占据多行
- **Model.Printable (布尔)** - 可以打印控制元素
- **Model.ReadOnly (布尔)** - 控制元素的内容只读
- **Model.Tabstop (布尔)** - 可以通过 Tab 键访问控制元素
- **Model.Text (字串)** - 文本与控制元素关联
- **Model.TextColor (长整数)** - 控制元素的文字颜色

- **Model.VScroll (布尔)** - 文本具有垂直的滚动条
- **Model.HelpText (字串)** - 当鼠标光标置于控制元素上方时显示的帮助文字
- **Model.HelpURL (字串)** - 与控制元素对应的联机帮助的 URL

列单框

列单框 (com.sun.star.awt.UnoControlListBox 服务) 支持以下属性：

- **ItemCount (短整数)** - 元素的数目，只读
- **SelectedItem (字串)** - 突出显示条目的文本，只读
- **SelectedItems (字串数组)** - 含有突出显示条目的数据字段，只读
- **SelectedItemPos (短整数)** - 正在突出显示的条目数目，只读
- **SelectedItemsPos (短整数数组)** - 含有突出显示条目数目的数据字段 (适用于支持多重选择的列单)，只读
- **MultipleMode (布尔)** - 值为 True 时会启动选择多重条目的选项，值为 False 时会禁止多重选择 (不能直接调用此属性，只能通过 IsMultipleMode 进行访问)
- **IsMultipleMode (布尔)** - 允许在列单内进行多重选择，只读

列单框提供以下方法：

- **addItem (Item, Pos)** - 将在 Item 中指定的字串输入位于 Pos 位置的列单中
- **addItems (ItemArray, Pos)** - 将在 字串的 ItemArray 数据字段中列出的条目输入位于 Pos 位置的列单中
- **removeItems (Pos, Count)** - 删除多个 Count 条目，从 Pos 位置开始
- **selectItem (Item, SelectMode)** - 根据 SelectMode 布尔变量启动或关闭突出显示的字串 Item 中指定的元素
- **makeVisible (Pos)** - **滚动列单字段，以便显示通过** Pos 指定的条目。

列单框的模型对象提供以下属性：

- **Model.BackgroundColor (长整数)** - 控制元素的背景颜色
- **Model.Border (短整数)** - 边框类型 (0：无边框，1：3 维边框，2：简单边框)
- **Model.FontDescriptor (结构)** - 含有所使用字体的细节的结构 (与 com.sun.star.awt.FontDescriptor 结构一致)
- **Model.LineCount (短整数)** - 控制元素中的行数
- **Model.MultiSelection (布尔)** - 允许多重选择条目
- **Model.SelectedItems (字串数组)** - 突出显示条目的列单
- **Model.StringItemList (字串数组)** - 所有条目的列单
- **Model.Printable (布尔)** - 可以打印控制元素
- **Model.ReadOnly (布尔)** - 控制元素的内容为只读
- **Model.Tabstop (布尔)** - 可以通过 Tab 键访问控制元素。
- **Model.TextColor (长整数)** - 控制元素的文字颜色
- **Model.HelpText (字串)** - 当鼠标光标置于控制元素上方时，自动显示的帮助文字
- **Model.HelpURL (字串)** - 与控制元素对应的联机帮助的 URL

VBA 中有用于发送具有数字附加值的列单条目 (ItemData) 的选项，而 StarSuite Basic 中没有对应的选项。如果除了自然语言文本以外，还需要管理数值 (例如，数据库 ID)，除列单框外还必须建立一个用于管理目的的辅助数据字段。

第 **12** 章

表单

StarSuite 表单的结构在许多方面都与前一章所讨论的对话框相似。然而，它们之间也存在一些关键的不同点：

- 对话框以单个对话视窗的形式显示在文档的上方，并且在对话框结束之前，不能对其执行除对话框处理之外的任何操作。 而表单就像是绘图元素一样直接显示在文档中。
- 具有一个为建立对话框而提供的对话框编辑器，可在 StarSuite Basic 开发环境中找到此编辑器。而表单则使用 **[表单功能]** 图标栏直接在文档内建立。
- 所有的 StarSuite 文档都提供对话框功能，而只有文本和工作表中提供完整的表单功能。
- 表单中的控制元素可以链接到外部数据库表。而对话框中则没有此功能。
- 对话框和表单中的控制元素存在多方面的差异。

如果希望在表单中使用自定义的事件处理方法，请参阅第 11 章对话框。上一章所介绍的这一机制与表单所使用的机制完全一致。

使用表单

StarSuite 表单中可能会包含可以直接插入文本或工作表中的文本字段、列单框、单选按钮以及一系列其他控制元素。**[**表单功能**]** 图标栏用于编辑表单。

StarSuite 表单可能会采用以下两种模式中的任一种模式：草稿模式和显示模式。在草稿模式下，可以更改控制元素的位置，还可以使用属性视窗编辑它们的属性。

[表单功能] 图标栏也用于在模式之间进行切换。

确定对象表单

StarSuite 将表单中的控制元素置于绘图对象级。实际的对象表单可以通过绘图级的 `Forms` 列单进行访问。在文本文档中，可按以下方式访问这些对象：

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

`GetByIndex` 方法返回了 Index 值为 0 的表单。

使用工作表时，需要通过 `Sheets` 列单获得一个中间级别，因为绘图级不是直接位于文档内，而是分布在各个工作表中：

```
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` 方法的名称所暗示的，一个文档可能会含有多个表单。这非常有用，例如，如果要在一个文档内显示不同数据库的内容，或者要在一个表单内显示一种 1:n 的数据库关系。出于此目的，还提供了一个建立子表单的选项。

控制元素表单的三要素

表单的控制元素具有三个要素：

- 第一个是控制元素的**模型**。当使用控制元素表单时，对于 StarSuite Basic 程序员来说，它是关键对象。
- 与此相对应的是控制元素的**视图**，它管理着显示的信息。
- 由于在文档内，控制元素表单被当作一种特殊的绘图元素进行管理，因此还具有一个 *Shape* **对象**，用于反映该控制元素的绘图元素属性 (尤其是位置和大小)。

读取控制元素表单的模型

可通过 Object **表单**的 GetByName 方法获得表单控制元素的模型：

```
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")
```

此示例确定了 MyListBox 控制元素的模型，它位于当前打开的文本文档的第一个表单中。

如果不确定某个控制元素所属的表单，则可以使用搜索选项对所有表单进行搜索，以便找出控制元素所需的表单：

```
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 方法检查文本文档中的所有表单，以确定这些表单是否包含一个名为 MyListBox 的控制元素模型。如果找到一个相应的模型，则此模型的引用将保存在 Ctl 变量中，然后终止搜索。

读取控制元素表单的视图

要访问控制元素表单的视图，首先需要关联模型。然后，控制元素的视图会在模型的帮助下通过文档控制器确定。

```
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
```

此示例中列出的代码与前面的示例中列出的确定控制元素模型的代码非常相似。它不仅使用了 `Doc` 文档对象，而且还使用了引用当前文档窗口的 `DocCrl` 文档控制器对象。在此控制器对象以及控制元素模型的帮助下，本示例接着使用 `GetControl` 方法确定了控制元素表单的视图 (`CtlView` 变量)。

读取控制元素表单的 Shape 对象

用于访问控制元素 Shape 对象的方法也使用文档的相应绘图级。要确定一个特殊的控制元素，就必须搜索绘图级上的所有绘图元素。

```
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
```

此示例检查了所有绘图元素，以确定它们是否支持控制元素表单所需要的 com.sun.star.drawing.XControlShape 接口。如果支持，Control.Name 属性会接着检查此控制元素的名称是否为 MyListBox。如果是，函数将结束搜索。

确定控制元素的大小和位置

如前所述，可以使用关联的 Shape 对象确定控制元素的大小和位置。为此，与所有其他 Shape 对象一样，控制元素的 Shape 对象也提供 Size 和 Position 属性：

- **Size (结构)** - 控制元素的大小 (com.sun.star.awt.Size 数据结构)。
- **Position (结构)** - 控制元素的位置 (com.sun.star.awt.Point 数据结构)。

以下示例显示如何使用关联的 Shape 对象设定控制元素的位置和大小：

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

要使代码生效，控制元素的 Shape 对象必须为已知对象。如果不是已知对象，则必须使用以上代码确定此对象。

控制元素表单详述

可用于表单的控制元素与可用于对话框的控制元素类似。其选择范围包括简单文本字段、列单框和组合框以及各种按钮。

在下面的章节中，将分别介绍控制元素表单的一些最重要的属性。每个属性都是构成关联模型对象的组成部分。

在表单中，除了可以使用标准控制元素，还可以使用表格控制元素，以便并入完整的数据库表。第 11 章的〈数据库表单〉一节中介绍了此内容。

按钮

表单按钮的**模型**对象提供以下属性：

- **BackgroundColor (长整数)** - 背景颜色。
- **DefaultButton (布尔)** - 设定为默认值的按钮。此处，如果该属性没有焦点，它也会响应为输入按钮。
- **Enabled (布尔)** - 可以启动控制元素。
- **Tabstop (布尔)** - 可以通过 Tab 按钮访问控制元素。
- **TabIndex (长整数)** - 控制元素在启动序列中的位置。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **Tag (字串)** - 包含附加信息的字串，可以保存在按钮中用于由程序控制的访问。
- **TargetURL (字串)** - URL 类型按钮所对应的目标 URL。
- **TargetFrame (字串)** - 启动按钮 (表示 URL 类型的按钮) 时，将在其中打开 TargetURL 的视窗 (或框) 的名称。
- **Label (字串)** - 按钮标签。
- **TextColor (长整数)** - 控制元素的文本颜色。
- **HelpText (字串)** - 当将鼠标光标置于控制元素上方时自动显示的帮助文字。
- **HelpURL (字串)** - 与控制元素对应的联机帮助的 URL。
- **ButtonType (枚举)** - 与按钮链接的操作 (来自 com.sun.star.form.FormButtonType 的默认值)。

通过使用 ButtonType 属性，可以自行定义按下按钮时要自动执行的操作。与之关联的 com.sun.star.form.FormButtonType 常数组提供以下值：

- **PUSH** - 标准按钮。
- **SUBMIT** - 结束表单输入 (特别适用于 HTML 表单)。
- **RESET** - 将表单内的所有值重设为原始值。
- **URL** - 对 TargetURL 中定义的 URL 的调用 (在通过 TargetFrame 指定的视窗内打开)。

表单中不支持对话框中提供的“确定”按钮和“取消”按钮。

选项字段

可通过选项字段的**模型**对象使用它的以下属性：

- **Enabled (布尔)** - 可以启动控制元素。
- **Tabstop (布尔)** - 可以通过 Tab 键访问控制元素。
- **TabIndex (长整数)** - 控制元素在启动序列中的位置。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **Tag (字串)** - 含有附加信息的字串，可保存在按钮中用于由程序控制的访问。
- **Label (字串)** - 按钮的标签。
- **Printable (布尔)** - 可以打印控制元素。
- **State (短整数)** - 如果值为 1，则启动该选项，否则关闭该选项。
- **RefValue (字串)** - 用于保存附加信息的字串 (例如，用于管理数据记录 ID)。
- **TextColor (长整数)** - 控制元素的文本颜色。
- **HelpText (字串)** - 当将鼠标光标置于控制元素上方时自动显示的帮助文字。
- **HelpURL (字串)** - 与控制元素对应的联机帮助的 URL。

通过选项字段的分组机制能够区分出用于对话框的控制元素和用于表单的控制元素。在对话框中，控制元素逐个出现并自动组合成一个组，而表单中的分组则是按名称进行的。为此，每个组中的所有选项字段都必须含有同样的名称。StarSuite 将这些已分组的控制元素合并到一个数组中，以便使用以上介绍的方法访问 StarSuite Basic 程序的各个按钮。

以下示例显示如何确定控制元素组的模型。

```
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
```

此代码与之前的用于确定简单控制元素模型的代码类似。它通过循环搜索当前文本文档内的所有表单，并使用 `HasByName` 方法检查相应表单是否含有正在搜索的名为 `MyOptions` 的元素。如果含有此元素，则使用 `GetGroupByName` 方法 (而不是确定简单模型的 `GetByName` 方法) 访问模型**数组**。

复选框

复选框表单的**模型**对象提供以下属性：

- **Enabled (布尔)** - 可以启动控制元素。
- **Tabstop (布尔)** - 可以通过 Tab 键访问控制元素。
- **TabIndex (长整数)** - 控制元素在启动序列中的位置。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **Tag (字串)** - 含有附加信息的字符串，可保存在按钮中用于由程序控制的访问。
- **Label (字串)** - 按钮标签。
- **Printable (布尔)** - 可以打印控制元素。
- **State (短整数)** - 如果值为 1，则启动该选项，否则关闭该选项。
- **RefValue (字串)** - 用于保存附加信息的字串 (例如，用于管理数据记录 ID)。
- **TextColor (长整数)** - 控制元素的文本颜色。
- **HelpText (字串)** - 当将鼠标光标置于控制元素上方时自动显示的帮助文字。
- **HelpURL (字串)** - 与控制元素对应的联机帮助的 URL。

文本字段

文本字段表单的**模型**对象提供以下属性：

- **Align (短整数)** - 文本方向 (0：左对齐，1：居中，2：右对齐)
- **BackgroundColor (长整数)** - 控制元素的背景颜色。
- **Border (短整数)** - 边框类型 (0：无边框，1：3 维边框，2：简单边框)。
- **EchoChar (字串)** - 密码字段的回显字符。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **HardLineBreaks (布尔)** - 在控制元素文本中永久插入自动换行。
- **HScroll (布尔)** - 文本具有一个水平滚动条。
- **MaxTextLen (短整数)** - 最大文本长度；如果指定值为 0，则无长度限制。
- **MultiLine (布尔)** - 允许多行输入。
- **Printable (布尔)** - 可以打印控制元素。
- **ReadOnly (布尔)** - 控制元素的内容为只读。
- **Enabled (布尔)** - 可以启动控制元素。
- **Tabstop (布尔)** - 可以通过 Tab 键访问控制元素。
- **TabIndex (长整数)** - 控制元素在启动序列中的位置。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **Text (字串)** - 控制元素的文本。
- **TextColor (长整数)** - 控制元素的文本颜色。
- **VScroll (布尔) -** 文本具有一个垂直滚动条。
- **HelpText (字串)** - 当将鼠标光标置于控制元素上方时自动显示的帮助文字。
- **HelpURL (字串) -** 与控制元素对应的联机帮助的 URL。

列单框

列单框表单的**模型**对象提供以下属性：

- **BackgroundColor (长整数)** - 控制元素的背景颜色。
- **Border (短整数)** - 边框类型 (0：无边框，1：3 维边框，2：简单边框)。
- **FontDescriptor (结构)** - 含有所使用字体的细节的结构 (与 `com.sun.star.awt.FontDescriptor` 结构一致)。
- **LineCount (短整数)** - 控制元素的行数。
- **MultiSelection (布尔)** - 允许多重选择条目。
- **SelectedItems (字串数组)** - 突出显示条目的列单。
- **StringItemList (字串数组)** - 所有条目的列单。
- **ValueItemList (变体数组)** - 含有各条目的附加信息的列单 (例如，用于管理数据记录 ID)。
- **Printable (布尔)** - 可以打印控制元素。
- **ReadOnly (布尔)** - 控制元素的内容为只读。
- **Enabled (布尔)** - 可以启动控制元素。
- **Tabstop (布尔)** - 可以通过 Tab 键访问控制元素。
- **TabIndex (长整数)** - 控制元素在启动序列中的位置。
- **FontName (字串)** - 字体类型的名称。
- **FontHeight (单精度)** - 以点数 (pt) 为单位的字符高度。
- **Tag (字串)** - 含有附加信息的字串，可保存在按钮中用于由程序控制的访问。
- **TextColor (长整数)** - 控制元素的文本颜色。
- **HelpText (字串)** - 当将鼠标光标置于控制元素上方时自动显示的帮助文字。
- **HelpURL (字串) -** 与控制元素对应的联机帮助的 URL。

通过其 `ValueItemList` 属性，列单框表单提供一种对应 VBA 属性的属性 `ItemData`，用于管理各列单条目的附加信息。

此外，通过列单框中的**视图** 对象，还提供了以下方法：

- **addItem (Item, Pos)** - 将在 `Item` 中指定的字串插入到列单中的 `Pos` 位置。
- **addItems (ItemArray, Pos)** - 将字串的 `ItemArray` 数据字段中列出的条目插入到 `Pos` 位置的列单中
- **removeItems (Pos, Count)** - 删除多个 `Count` 条目，从 `Pos` 位置开始
- **selectItem (Item, SelectMode)** - 根据 `SelectMode` 变量启动或关闭突出显示的字串 `Item` 中指定的元素。
- **makeVisible (Pos)** - 滚动列单字段，以便显示由 `Pos` 指定的条目。

数据库表单

StarSuite 表单可以直接链接到数据库。用此方法建立的表单，既不需要独立的编程工作，而且还为完整的数据库前端提供了所有功能。

用户可以选择翻页或搜索选定的表格和查询，以及更改数据记录或插入新的数据记录。StarSuite 会自动确保从数据库检索到相关的数据，并确保所做的任何更改都将写回到数据库中。

数据库表单基本上相当于一个标准的 StarSuite 表单。除标准属性之外，还必须在表单中设定以下数据库属性：

- **DataSourceName (字串)** - 数据源的名称 (请参阅第 10 章访问数据库; 数据库访问; 必须以全局方式在 StarSuite 中建立数据源)。
- **Command (字串)** - 要为其建立链接的表格、查询或 SQL 选择命令的名称。
- **CommandType (常数)** - 指定 Command 为一个表格、查询还是 SQL 命令 (com.sun.star.sdb.CommandType 枚举中的值)。

com.sun.star.sdb.CommandType 枚举涉及以下值：

- **TABLE** - 表格
- **QUERY** - 查询
- **COMMAND** - SQL 命令

数据库字段通过以下属性指定给各控制元素：

- **DataField (字串)** - 已链接的数据库字段的名称。

表格

为了使用数据库，还提供了一种控制元素：表格控制元素。此元素集中体现了完整的数据库表格或查询的内容。最初级的方案为，表格控制元素通过自动文件助理表单链接到数据库，此操作将根据用户的规定将各个列链接到相关的数据库字段。由于关联的 API 相当复杂，有关 API 的这一方面，本文不提供完整的说明。

附录

VBA 移植提示

StarOffice 5.x 移植提示

索引

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y