StarSuite™ 7 Office Suite

A Sun™ ONE Software Offering

Basic 프로그래밍 설명서

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

Part No. 817-3926-10
2003, Revision A

Copyrights and Trademarks

이 설명서의 내용

1 장

소개

이 설명서에서는 StarSuite Basic 7 프로그래밍을 소개하고 StarSuite 에서 StarSuite Basic 을 사용하여 제공할 수 있는 응용 프로그램에 대해 설명합니다. 이 설명서를 최대한 활용하려면 다른 프로그래밍 언어에 대한 지식이 있어야 합니다.

StarSuite Basic 프로그램을 신속하게 개발하는 데 도움이 될 풍부한 예를 제공합니다.

Microsoft Visual Basic 프로그래머나 StarSuite Basic 이전 버전의 사용자를 위해 이 설명서 전체에 걸쳐 다양한 이전 팁을 제공합니다. 이러한 예는 페이지 가장자리에 작은 기호로 표시합니다. 이 설명서의 부록에는 모든 이전 팁에 대한 색인이 있으므로 이 색인을 통해 원하는 팁을 신속하게 찾을 수 있습니다.

StarSuite Basic 정보

StarSuite Basic 프로그래밍 언어는 StarSuite 용으로 특별히 개발되었으므로 StarSuite 패키지와 완벽하게 통합됩니다.

이름만으로도 알 수 있듯이 StarSuite Basic 은 Basic 제품군의 프로그래밍 언어입니다. Microsoft Visual Basic, VBA(Visual Basic for Applications) 등과 같이 다른 Basic 언어를 사용한 경험이 있는 개발자는 StarSuite Basic 의 경우에도 빨리 익숙해질 수 있습니다. StarSuite Basic 의 기본 구성소 중 상당수는 Visual Basic 과 호환됩니다.

StarSuite Basic 프로그래밍 언어는 4 가지 구성 요소로 구분할 수 있습니다.

- **StarSuite Basic 의 언어**: 변수 선언, 루프, 함수 등과 같은 기본적인 언어 구성소를 지정합니다.
- **런타임 라이브러리**: 숫자, 문자열, 날짜 값, 파일을 편집하는 함수처럼 StarSuite 를 직접 참조하지 않는 표준 함수를 제공합니다.
- **StarSuite API(응용 프로그램 프로그래밍 인터페이스)**: StarSuite 문서에 액세스하여 문서를 작성, 저장, 수정 및 인쇄할 수 있게 합니다.
- **대화 상자 편집기**: 개인 대화 상자 창을 만들고 컨트롤 요소 및 이벤트 처리기를 추가할 범위를 제공합니다.

StarSuite Basic 과 VBA 사이의 호환성은 런타임 라이브러리 및 StarSuite Basic 언어와 관련되어 있습니다. StarSuite API 와 대화 상자 편집기는 VBA 와 호환되지 *않습니다.* 이 인터페이스를 표준화할 경우 StarSuite 개념 중 상당수의 구현이 불가능해 집니다.

StarSuite Basic 의 대상 사용자

StarSuite Basic 응용 프로그램은 StarSuite 표준 함수만으로는 구현할 수 없는 기능을 위해 필요합니다. 따라서 StarSuite Basic 을 사용하여 반복되는 작업을 자동화하고, 데이터베이스 서버 등의 다

른 프로그램과 연결하고, 미리 지정한 스크립트를 사용하는 버튼을 눌러 복잡한 작업을 수행할 수 있습니다.

StarSuite Basic 은 모든 StarSuite 함수에 대해 완전한 액세스를 제공하고, 모든 함수를 지원하며, 문서 유형을 수정하고, 개인 대화 상자 창을 만드는 옵션을 제공합니다.

StarSuite Basic 의 사용

StarSuite 사용자라면 누구나 추가적인 프로그램이나 지원 없이도 StarSuite Basic 을 사용할 수 있습니다.
표준 설치만으로도 StarSuite Basic 은 다음과 같이 고유한 Basic 매크로를 만드는 데 필요한 모든 구성 요소를 갖추고 있습니다.

- **통합 개발 환경**(IDE)은 매크로를 입력하고 테스트할 수 있는 편집기를 제공합니다.
- **인터프리터**는 StarSuite Basic 매크로를 실행하는 데 필요합니다.
- **인터페이스**는 다양한 StarSuite 응용 프로그램과 연결하여 StarSuite 문서를 직접 액세스할 수 있게 합니다.

이 설명서의 구성

처음 세 장에서는 StarSuite Basic 에 대해 소개합니다.

- 2 장: StarSuite Basic 의 언어
- 3 장: StarSuite Basic 의 런타임 라이브러리
- 4 장: StarSuite API 소개

이 장에서는 StarSuite Basic 을 개괄적으로 소개하며 StarSuite Basic 프로그램을 개발하려는 경우 반드시 읽어야 합니다.

나머지 장에서는 StarSuite API 의 각 구성 요소를 자세하게 설명하고 있으며, 필요한 경우 선택적으로 읽을 수 있습니다.

- 5 장: StarSuite 문서 작업
- 6 장: 텍스트 문서
- 7 장: 스프레드시트 문서
- 8 장: 그리기 및 프레젠테이션
- 9 장: 다이어그램(차트)
- 10 장: 데이터베이스 액세스
- 11 장: 대화 상자
- 12 장: 양식

추가 정보

이 설명서에서 다루는 StarSuite API 구성 요소는 실제로 StarSuite Basic 프로그래머에게 제공하는 혜택을 기준으로 선정했습니다. 일반적으로 이 인터페이스의 일부만 다루게 됩니다. 자세한 내용은 다음 인터넷 사이트에서 제공하는 API 참조 설명서를 참조하십시오.

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

Developer's Guide 에서 StarSuite API 에 대해 이 설명서보다 더 자세히 설명하고 있지만 주로 Java 및 C++ 프로그래머를 대상으로 하고 있습니다. StarSuite Basic 에 익숙한 개발자라면 Developer's Guide 에서 StarSuite Basic 및 StarSuite 프로그래밍에 대한 추가 정보를 얻을 수 있습니다. Developer's Guide 는 다음 인터넷 사이트에서 다운로드할 수 있습니다.

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

StarSuite Basic 보다 Java 나 C++를 사용하여 직접 작업하려는 프로그래머는 이 설명서 대신 StarSuite Developer's Guide 를 참조하는 것이 좋습니다. Java 나 C++를 사용한 StarSuite 프로그래밍은 StarSuite Basic 프로그래밍에 비해 훨씬 복잡한 과정입니다.

2 장

StarSuite Basic 의 언어

StarSuite Basic 은 Basic 언어의 제품군입니다. StarSuite Basic 의 상당 부분은 Microsoft Visual Basic for Applications 및 Microsoft Visual Basic 과 같습니다. 이러한 언어를 사용해본 적이 있다면 StarSuite Basic 도 빠르게 익힐 수 있습니다.

Java, C++, Delphi 와 같은 다른 언어의 프로그래머 역시 StarSuite Basic 을 쉽게 사용할 수 있습니다. StarSuite Basic 은 완전하게 개발된 절차식 프로그래밍 언어이기 때문에 GoTo 및 GoSub 와 같은 초보적인 제어 구조를 더 이상 사용하지 않습니다.

또한 StarSuite Basic 인터페이스에서는 외부 개체 라이브러리 사용을 지원하므로 개체 지향 프로그래밍의 장점을 활용할 수 있습니다. 전체 StarSuite API 는 이러한 인터페이스를 기반으로 하며 여기에 관해서는 나중에 더 자세히 설명합니다.

이 장에서는 StarSuite Basic 언어의 핵심 요소 및 구성소, StarSuite Basic 기반 응용 프로그램 및 라이브러리의 프레임워크를 개괄적으로 소개합니다.

StarSuite Basic 프로그램 개요

StarSuite Basic 은 인터프리터 언어입니다. C++나 Turbo Pascal 과 달리 StarSuite 컴파일러는 자동으로 실행되는 실행 파일이나 자동 압축 파일을 만들지 않습니다. 그 대신 버튼을 눌러 StarSuite Basic 프로그램을 실행할 수 있습니다. 먼저 코드에 명백한 오류가 없는지 검사한 다음 줄 단위로 실행합니다.

프로그램 줄

Basic 인터프리터가 줄 기준으로 작동한다는 것은 Basic 과 다른 프로그래밍 언어를 구분짓는 중요한 차이점 중 하나입니다. Java, C++, Delphi 와 같은 프로그램 원본 코드에서는 줄바꿈의 위치가 중요하지 않지만 Basic 프로그램에서 각 줄은 독립적인 단위를 구성합니다. 함수 호출, 수학식 그리고 함수와 루프 헤더와 같은 기타 언어 요소는 시작했던 같은 줄에서 완료해야 합니다.

입력 시 충분한 공간이 없거나 줄이 길어지면 밑줄(_)을 추가하여 여러 줄을 연결할 수 있습니다. 다음 예는 네 줄의 수학식을 연결하는 방법입니다.

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

연결된 줄에서는 항상 밑줄이 마지막 문자이며 그 다음에 공백이나 탭을 사용하지 않아야 합니다. 그렇지 않으면 코드에서 오류가 생성됩니다.

개별 줄을 연결하는 것 외에도 StarSuite Basic 에서는 여러 식이 사용되는 경우에 충분한 공간을 확보하도록 콜론을 사용하여 한 줄을 여러 구역으로 구분할 수 있습니다. 예를 들어 다음 대입식의 경우

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

다음과 같이 작성할 수 있습니다.

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

주석

StarSuite Basic 프로그램은 실행할 프로그램 코드 외에도 주석을 포함할 수 있으며, 이 주석은 프로그램의 각 부분을 설명하고 이후의 문제 해결 등을 위해 프로그램 코드를 재검토할 때 도움이 될 중요한 정보를 담고 있습니다.

StarSuite Basic 은 프로그램 코드에 주석을 삽입하는 2 가지 방법을 제공합니다.

- 아포스트로피 다음에 오는 문자는 모두 주석으로 처리됩니다.

  ```
  Dim A    ' This is a comment for variable A
  ```

- 키워드 `Rem` 다음에도 주석이 옵니다.

  ```
  Rem This comment is introduced by the keyword Rem.
  ```

일반적으로 주석은 해당 줄 끝까지의 모든 문자를 포함합니다. 그리고 나서 StarSuite Basic 은 그 다음 줄부터 일반 명령으로 해석합니다. 여러 줄에 걸쳐 주석을 입력하려면 줄마다 주석으로 표시해야 합니다.

```
Dim B       ' This comment for variable B is relatively long
            ' and stretches over several lines. The
            ' comment character must therefore be repeated
            ' in each line.
```

표시자

StarSuite Basic 프로그램은 수십, 수백, 수천 여 개의 변수, 상수, 함수 등의 이름, 즉 표시자를 포함할 수 있습니다.
표시자 이름을 선택할 때 다음이 적용됩니다.

- 표시자는 영어, 숫자, 밑줄(_)만 포함할 수 있습니다.
- 표시자의 첫 문자는 반드시 문자나 밑줄이어야 합니다.
- 표시자는 ä â î ß 와 같은 특수 문자를 포함할 수 없습니다.

- 표시자의 최대 길이는 255 자입니다.
- 단, 대/소문자는 구분하지 않습니다. 예를 들어 표시자 `OneTestVariable` 은 `onetestVariable` 및 `ONETESTVARIABLE` 과 같은 변수를 지정합니다.

  그러나 이 규칙에 한 가지 예외가 있습니다. UNO-API 상수는 대/소문자를 구분합니다. UNO 에 대한 자세한 내용은 4 장을 참조하십시오.

StarSuite Basic 의 표시자 구성 규칙은 VBA 와 다릅니다. 예를 들어, VBA 와 달리 StarSuite Basic 은 표시자에 특수 문자를 사용하지 않습니다. 국제적인 프로젝트에서 문제가 될 수 있기 때문입니다.

다음은 표시자에 대한 여러 가지 사용 예를 보여 줍니다.

```
Surname              ' 올바름
Surname5             ' 올바름(숫자 5 가 처음에 오지 않았습니다.)
First Name           ' 올바르지 않음(공백은 사용할 수 없습니다.)
DéjàVu               ' 올바르지 않음(é, à 와 같은 문자는 사용할 수 없습니다.)
5Surnames            ' 올바르지 않음(첫 번째 문자로 숫자를 사용할 수 없습니다.)
First,Name           ' 올바르지 않음(쉼표와 마침표는 사용할 수 없습니다.)
```

변수 사용

암시적 변수 선언

Basic 언어는 사용하기 편리하도록 설계되었습니다. 따라서 StarSuite Basic 에서는 변수를 명시적으로 선언하지 않고 사용하더라도 변수가 만들어집니다. 즉 변수를 코드에 포함하는 시점부터 변수가 존재합니다. 변수가 존재하는지 여부에 따라 다음 코드는 최대 3 개의 변수를 새로 선언합니다.

```
a = b + c
```

입력할 때 실수로 원치 않는 새 변수를 만들 수 있기 때문에 변수를 암시적으로 선언하는 것은 그리 좋은 방법이 아닙니다. 이럴 경우 인터프리터는 잘못 입력한 변수에 대해 오류 메시지를 표시하지 않고 이를 새 변수로 간주하여 값을 0 으로 초기화합니다.

명시적 변수 선언

암시적 변수 선언으로 인한 오류를 방지하기 위해 StarSuite Basic 은 다음과 같은 스위치를 제공합니다.

```
Option Explicit
```

이 스위치를 각 모듈의 첫 번째 프로그램 줄에 표시할 경우 선언되지 않은 변수가 있으면 오류 메시지가 표시됩니다. `Option Explicit` 스위치는 모든 Basic 모듈에 포함해야 합니다.

가장 간단한 형태의 명시적 변수 선언은 다음과 같습니다.

```
Dim MyVar
```

이 예에서는 이름이 `MyVar` 이고 유형은 Variant 인 변수를 선언합니다. Variant 는 문자열, 정수, 부동 소수점 숫자, Boolean 값 등 가능한 모든 값을 기록할 수 있는 범용 변수입니다. Variant 변수의 몇 가지 예를 소개합니다.

```
MyVar = "Hello World"                    ' 문자열 할당
MyVar = 1                              ' 정수 할당
MyVar = 1.0                            ' 부동 소수점 숫자 할당
MyVar = True                           ' Boolean 값 할당
```

위에서 선언한 변수는 같은 프로그램에서 다른 변수 유형으로도 사용할 수 있습니다. 따라서 상당한 유연성이 확보되지만 변수는 하나의 변수 유형으로만 사용하는 것이 가장 좋습니다. StarSuite Basic 은 특정 컨텍스트에서 잘못 지정된 변수 유형을 발견하면 오류 메시지를 생성합니다.

입력이 제한된 변수를 선언할 때는 다음과 같은 방식을 사용합니다.

```
Dim MyVar As Integer    ' integer 유형의 변수 선언
```

이 변수는 integer 유형으로 선언되었으므로 정수값을 기록할 수 있습니다. 또한 다음과 같은 방법으로 integer 유형의 변수를 선언할 수 있습니다.

```
Dim MyVar%              ' integer 유형의 변수 선언
```

Dim 명령문으로 여러 변수를 선언할 수 있습니다.

```
Dim MyVar1, MyVar2
```

변수를 영구적인 유형으로 할당하려면 각 변수를 개별적으로 할당해야 합니다.

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

변수의 유형을 선언하지 않으면 StarSuite Basic 은 해당 변수에 Variant 유형을 할당합니다. 예를 들어, 다음 변수 선언에서 MyVar1 은 Variant, MyVar2 는 Integer 가 됩니다.

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

아래에서는 StarSuite Basic 에서 지원하는 변수 유형 그리고 해당 변수 유형의 사용 및 선언 방법을 소개합니다.

문자열

숫자와 더불어 문자열은 StarSuite Basic 을 구성하는 가장 중요한 기본 유형입니다. 문자열은 연속되는 개별 문자들의 시퀀스로 구성됩니다. 컴퓨터는 내부적으로 문자열을 일련의 숫자로 저장하며, 각 숫자가 하나의 문자를 나타냅니다.

ASCII 문자 집합에서 유니코드로

문자 집합은 컴퓨터가 화면 또는 프린터에 문자열을 인쇄하는 방법을 설명하는 테이블상의 해당 코드(숫자와 문자)와 문자열의 문자를 일치시킵니다.

ASCII 문자 집합

ASCII 문자 집합은 숫자, 문자, 특수 기호를 1 바이트로 나타내는 코드의 집합입니다. 0~127 의 ASCII 코드는 알파벳과 마침표, 괄호, 쉼표 등의 일반 기호, 일부 화면 및 프린터 제어용 특수 코드를 나타냅니다. ASCII 문자 집합은 컴퓨터 사이에 텍스트 데이터를 전송하는 표준 서식으로 많이 사용합니다.

그러나 이 문자 집합은 â, ä, î 와 같이 유럽에서 사용하는 모든 특수 문자 그리고 키릴 알파벳과 같은 문자 서식은 포함하지 않습니다.

ANSI 문자 집합

Microsoft Windows 제품의 기반이 되는 ANSI(American National Standards Institute) 문자 집합은 ASCII 문자 집합에서 지원하지 않은 문자까지 포함하도록 점진적으로 확장되어 왔습니다.

코드 페이지

ISO 8859 문자 집합은 오랫동안 요구되어 왔던 국제 표준을 제공합니다. ISO 문자 집합의 첫 128 문자는 ASCII 문자 집합에 해당됩니다. ISO 표준은 보다 많은 언어를 올바르게 표시할 수 있도록 새

로운 문자 집합(*코드 페이지*)을 도입합니다. 그러나 그로 인해 같은 문자값이 언어에 따라 다른 문자를 나타내기도 합니다.

유니코드

유니코드는 문자의 길이를 4 바이트로 늘리고 가급적 많은 세계 언어를 나타낼 수 있는 표준을 만들고자 서로 다른 문자 집합을 결합합니다. 유니코드 버전 2.0 은 현재 StarSuite, StarSuite Basic 을 포함하여 많은 프로그램에서 지원하고 있습니다.

String 변수

StarSuite Basic 은 문자열을 유니코드의 String 변수로 저장합니다. String 변수는 최대 65,535 자의 문자를 저장할 수 있습니다. StarSuite Basic 은 내부적으로 각 문자의 유니코드 값을 저장합니다. String 변수에 필요한 작업 메모리는 문자열의 길이에 따라 달라집니다.

String 변수 선언의 예는 다음과 같습니다.

```
Dim Variable As String
```

또한 다음과 같이 선언할 수 있습니다.

```
Dim Variable$
```

VBA 응용 프로그램을 포팅할 때 StarSuite Basic 에서 허용하는 최대 문자열 길이(65535 문자)를 준수하도록 하십시오.

명시적 문자열 지정

String 변수에 명시적 문자열을 할당하려면 해당 문자열을 따옴표(")로 묶습니다.

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

문자열을 두 줄에 걸쳐 나누어 표시하려면 첫 번째 줄 끝에 더하기 기호를 추가합니다.

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

문자열에 따옴표(")를 포함하려면 해당 위치에서 2 번 입력합니다.

```
Dim MyString As String
MyString = "a ""-quotation mark."   ' 값은 a "-quotation mark 가 됩니다.
```

숫자

StarSuite Basic 은 5 가지 기본 유형을 통해 숫자를 처리합니다.

- Integer
- Long Integer
- Float
- Double
- Currency

Integer 변수

Integer *변수는* -32768~32767 범위의 모든 정수를 저장할 수 있습니다. Integer 변수는 최대 2 바이트의 메모리를 사용합니다. Integer 변수의 유형 선언 기호는 %입니다. Integer 변수를 사용하는 계산은 속도가 매우 빠르며 특히 루프 카운터에 유용합니다. 부동 소수점 숫자를 Integer 변수에 할당하면 가장 가까운 정수로 반올림합니다.

Integer 변수 선언의 예는 다음과 같습니다.

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

Long Integer 변수

Long Integer 변수는 2147483648~2147483647 범위의 모든 정수를 저장할 수 있습니다. Long Integer 변수는 최대 4 바이트의 메모리를 사용합니다. Long Integer 의 유형 선언 기호는 &입니다. Long Integer 변수를 사용하는 계산은 속도가 매우 빠르며 특히 루프 카운터에 유용합니다. Long Integer 변수에 부동 소수점 숫자를 할당하면 가장 가까운 정수로 반올림합니다.

Long integer 변수 선언의 예는 다음과 같습니다.

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

Single 변수

Single 변수는 $3.402823x10^{38}$~$1.401298x10^{-45}$ 범위의 모든 양수 또는 음수 부동 소수점 숫자를 저장할 수 있습니다. Single 변수는 최대 4 바이트의 메모리를 사용합니다. Single 변수의 유형 선언 기호는 ! 입니다.

원래 Single 변수는 보다 정밀한 Double 변수보다 계산 시간을 단축하는 데 사용되어 왔습니다. 그러나 더 이상 속도를 고려할 필요가 없어지면서 Single 변수의 필요성이 줄어들고 있습니다.

Single 변수 선언의 예는 다음과 같습니다.

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

Double 변수

Double 변수는 $1.79769313486232x10^{308}$~$4.94065645841247x10^{-324}$ 범위의 모든 양수 또는 음수 부동 소수점 숫자를 저장할 수 있습니다. Double 변수는 최대 8 바이트의 메모리를 사용합니다. Double 변수는 정밀 계산에 적합합니다. 유형 선언 기호는 #입니다.

Double 변수 선언의 예는 다음과 같습니다.

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

Currency 변수

Currency 변수는 값을 처리하는 방식이 다른 변수 유형과 다릅니다. 소수점이 고정되며 소수점 이하 4 자릿수를 사용합니다. 이 변수는 정수부에 최대 15 자리까지 사용할 수 있습니다. Currency 변수는 -922337203685477.5808~+922337203685477.5807 범위의 모든 값을 저장할 수 있으며 최대 8 바이트의 메모리를 사용합니다. Currency 변수의 유형 선언 기호는 @입니다.

Currency 변수는 부동 소수점 숫자를 사용할 경우 예측 불가능한 반올림 오류가 발생할 수 있는 비즈니스 계산에서 주로 사용합니다.

Currency 변수 선언의 예는 다음과 같습니다.

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

명시적 숫자 지정

숫자는 십진 형식이나 과학적 표기법 또는 십진 체계가 아닌 다른 방식으로 표시할 수 있습니다. StarSuite Basic 의 숫자 문자에는 다음 규칙이 적용됩니다.

정수

가장 간단한 방법은 정수를 사용하는 것입니다. 정수는 원본 텍스트에서 천 단위 쉼표를 사용하지 않고 기록됩니다.

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

숫자 앞에 (+)나 (-) 표시를 붙일 수 있습니다(공백을 넣거나 넣지 않음).

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

10 진수

10 진수 입력 시 소수점으로 마침표(.)를 사용합니다. 이 규칙을 적용하면 국가 사이에 원본 텍스트 전송 시 변환할 필요가 없습니다.

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

A = 1223.53       ' 반올림 처리
B = - 23446.46    ' 반올림 처리
C = + 3532.76323
```

또한 10 진수 앞에 더하기(+) 또는 빼기(-) 기호를 사용할 수 있습니다(공백을 넣거나 넣지 않음).

Integer 변수에 10 진수를 할당하면 StarSuite Basic 은 그 숫자를 반올림합니다.

지수 표기 방식

StarSuite Basic 에서는 숫자의 지수 표기 방식을 지원합니다. 예를 들어 1.5×10^{-10}(0.00000000015)은 1.5e-10 으로 표기할 수 있습니다. 문자 "e"는 대 / 소문자 모두 가능하며, 앞에 (+) 기호가 올 수도, 오지 않을 수도 있습니다.

다음은 지수 형식의 숫자에 대한 여러 가지 사용 예를 보여 줍니다.

```
Dim A As Double

A = 1.43E2           ' 올바름
A = + 1.43E2         ' 올바름(더하기 기호와 숫자 사이에 공백)
A = - 1.43E2         ' 올바름(빼기 기호와 숫자 사이에 공백)
A = 1.43E-2          ' 올바름(음수)

A = 1.43E -2         ' 올바르지 않음(숫자 사이에는 공백 사용 불가)
A = 1,43E-2          ' 올바르지 않음(소수점으로 쉼표 사용 불가)
A = 1.43E2.2         ' 올바르지 않음(지수는 정수만 사용 가능)
```

첫 번째와 세 번째의 틀린 예에서는 변수가 틀린 값을 구하더라도 오류 메시지가 생성되지 않습니다.

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

위 식은 1.43 빼기 2 로 해석되어 -0.57 을 값으로 갖습니다. 그러나 의도했던 값은 $1.43 * 10^2$ (0.0143) 이었습니다.

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

값을 위와 같이 지정하면 StarSuite Basic 은 소수점 이후의 지수부를 무시하고 이 식을 다음과 같이 해석합니다.

```
A = 1.43E2
```

16 진수 값

16 진법(16 진수 체계)는 2 자리 숫자를 정확히 1 바이트로 나타낸다는 장점이 있습니다. 따라서 시스템 아키텍처에 더 근접하는 방식으로 숫자를 처리할 수 있습니다. 16 진법에서는 0~9 의 숫자, A~F 의 문자를 숫자로 사용합니다. A 는 10 진수 10 을, F 는 10 진수 15 를 나타냅니다. StarSuite Basic 에서는 16 진수 정수 앞에 &H 를 표시하여 사용할 수 있습니다.

```
Dim A As Longer

A = &HFF      ' 16 진수 FF, 10 진수 255 에 해당
A = &H10      ' 16 진수 10, 10 진수 16 에 해당
```

8 진수 값

또한 StarSuite Basic 은 0~7 의 숫자를 사용하는 8 진법(8 진수 체계)을 지원하며, 정수 앞에 &O 를 표시하여 사용합니다.

```
Dim A As Longer

A = &O77      ' 8진수 77, 10진수 63에 해당
A = &O10      ' 8진수 10, 10진수 8에 해당
```

True와 False

Boolean 변수

Boolean 변수는 `True`와 `False`의 2가지 값 중 하나만 가질 수 있습니다. 이 변수는 명명된 상태 중 하나만 받아들일 수 있는 바이너리 지정에 적합합니다. Boolean 값은 내부적으로 2바이트 정수 값으로 저장되는데, 0은 `False`, 기타 값은 `True`가 됩니다. Boolean 변수는 유형 선언 기호가 없습니다. 변수 선언 시 *As Boolean*이라고 덧붙이면 됩니다.

Boolean 변수 선언의 예는 다음과 같습니다.

```
Dim Variable As Boolean
```

날짜 및 시간 세부 정보

Date 변수

Date 변수는 날짜와 시간 값을 포함할 수 있습니다. 날짜 값 저장 시 StarSuite Basic은 날짜 및 시간 값에 대한 비교 및 수학적 연산을 지원하는 내부 서식을 사용합니다. Date 변수는 유형 선언 기호가 없습니다. *As Date*라고 덧붙여 선언하면 됩니다.

Date 변수 선언의 예는 다음과 같습니다.

```
Dim Variable As Date
```

데이터 필드

간단한 변수(*스칼라*) 외에도 StarSuite Basic 은 데이터 필드(*배열*)를 지원합니다. 데이터 필드는 색인을 통해 참조되는 몇 가지 변수를 포함합니다.

간단한 배열

배열 선언은 간단한 변수 선언과 유사하지만, 변수 선언과 달리 배열 이름 다음에 요소의 수가 괄호로 묶여 표시됩니다.

```
Dim MyArray(3)
```

위 식은 Variant 데이터 유형의 4 개 변수, 즉 `MyArray(0)`, `MyArray(1)`, `MyArray(2)`, `MyArray(3)`으로 구성된 배열을 선언합니다.

또한 특정 유형의 변수를 배열로 선언할 수 있습니다. 예를 들어 다음 줄은 4 개의 Integer 변수로 구성된 배열을 선언합니다.

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

위의 예에서 배열의 색인은 항상 표준 시작 값인 0 으로 시작합니다. 또는 해당 데이터 필드 선언에 대해 시작 값과 끝 값이 있는 유효 범위를 지정할 수 있습니다. 다음의 예에서는 6 개의 Integer 값으로 구성되며 5~10 의 색인을 사용하여 참조할 수 있는 데이터 필드를 선언합니다.

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

색인이 반드시 양수값일 필요는 없습니다. 다음의 예 또한 정확한 선언을 나타내지만 음수 데이터 필드로 제한됩니다.

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

6 개의 값을 갖고 -10 ~ -5 의 색인을 사용하여 참조할 수 있는 Integer 데이터 필드를 선언합니다.

데이터 필드 색인 지정 시 3 가지 제한 사항을 준수해야 합니다.

- 색인의 최소값은 -32,768 입니다.
- 색인의 최대값은 32,767 입니다.
- (데이터 필드 차원 내의) 최대 요소 개수는 16,368 입니다.

기타 제한값은 VBA 에서의 데이터 필드 색인에 적용되기도 합니다. 또한 차원별 최대 요소 개수에 대해서도 같은 제한이 적용됩니다. 유효한 값에 대해서는 관련 VBA 문서에서 확인할 수 있습니다.

시작 색인의 값 지정

데이터 필드의 시작 색인은 주로 0 부터 시작합니다. 또는 다음과 같이 호출을 사용하여 모든 데이터 필드 선언 시 시작 색인을 1 로 바꿀 수 있습니다.

```
Option Base 1
```

이 호출을 해당 모듈의 모든 배열 선언에 적용하려면 모듈의 헤더에 이 호출을 포함해야 합니다. 그러나 이 호출은 StarSuite API 에서 지정하고 색인이 *항상* 0 부터 시작하는 UNO 시퀀스에는 영향을 미치지 않습니다. 보다 명확성을 기하기 위해 Option Base 1 사용은 삼가해야 합니다.

Option Base 1 을 사용하면 배열의 요소 수는 영향을 받지 않으며 시작 색인만 바뀝니다.

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

위 선언은 `MyInteger(1)`, `MyInteger(2)`, `MyInteger(3)`, `MyInteger(4)` 등의 식으로 설명되는 4 개의 Integer 변수를 만듭니다.

VBA 와 달리 StarSuite Basic 에서 Option Base 1 식은 배열의 요소 수에 영향을 주지 않습니다. StarSuite Basic 에서 이동하는 시작 색인일 뿐입니다. 선언 `MyInteger(3)`이 VBA 에서는 색인 1~3 을 사용하는 3 개의 정수값을 만들지만, 같은 선언이 StarSuite Basic 에서는 색인 1~4 인 4 개의 정수값을 만듭니다.

다차원 데이터 필드

StarSuite Basic 에서는 1 차원 데이터 필드 외에도 다차원 데이터 필드를 사용할 수 있습니다. 차원은 쉼표를 사용하여 구분합니다. 다음의 예

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

는 각각 6 개의 색인을 갖는(0~5 의 색인으로 참조) 2 개의 차원으로 구성된 Integer 배열을 지정합니다. 배열 전체적으로 총 6 × 6 = 36 개의 Integer 값을 기록할 수 있습니다.

StarSuite Basic 배열에서는 수백 개의 차원을 지정할 수 있지만 사용 가능한 메모리의 양에 따라 차원의 수가 제한됩니다.

데이터 필드 차원의 동적 변화

위의 예는 차원이 지정된 데이터 필드를 바탕으로 합니다. 또한 데이터 필드의 차원이 동적으로 바뀌는 배열을 지정할 수 있습니다. 예를 들어 텍스트에서 A 문자로 시작되는 모든 단어를 포함하는 배열을 지정할 수 있습니다. 처음에는 그러한 단어 수를 알 수 없으므로 필드 제한을 향후 조정해야 합니다. StarSuite Basic 에서는 이 작업을 위해 다음과 같이 호출합니다.

```
ReDim MyArray(10)
```

VBA 에서는 오로지 Dim MyArray()를 사용하여 동적 배열의 차원을 결정하지만 StarSuite Basic 에서는 ReDim 을 사용하여 정적 및 동적 배열을 모두 바꿀 수 있습니다.

다음의 예는 최초 배열의 차원을 바꿔 11 또는 21 개의 값을 기록할 수 있게 합니다.

```
Dim MyArray(4) As Integer                ' 5 개 요소로 선언

' ...

ReDim MyArray(10) As Integer        ' 11 개 요소로 증가

' ...

ReDim MyArray(20) As Integer        ' 21 개 요소로 증가
```

배열의 차원을 다시 설정할 때 이전에 간략히 소개한 옵션 중 어떤 것이라도 사용할 수 있습니다. 여기에는 다차원 데이터 필드 선언, 명시적 시작 값 및 끝 값 지정 옵션도 포함됩니다. 데이터 필드의 차원이 바뀌면 모든 내용을 잃게 됩니다. 원래의 값을 유지하려면 `Preserve` 명령을 사용합니다.

```
Dim MyArray(10) As Integer                          ' 초기 차원 지정


' ...

ReDim Preserve MyArray(20)   As Integer             ' 내용을
                                                    ' 유지하면서
                                                    ' 데이터 필드 증가
```

`Preserve` 명령 사용 시 차원의 수 및 변수 유형은 같아야 합니다.

`Preserve` 사용 시 데이터 필드 마지막 차원의 최대값만 바꿀 수 있는 VBA 와 달리 StarSuite Basic 에서는 다른 차원도 바꿀 수 있습니다.

`ReDim` 을 `Preserve` 와 함께 사용하려면 원래의 데이터 필드 선언에서 지정한 데이터 유형을 똑같이 사용해야 합니다.

변수의 범위 및 수명

StarSuite Basic 변수는 수명 그리고 다른 프로그램 코드에서 이 변수를 읽고 사용할 수 있는 범위가 한정되어 있습니다. 변수를 유지하고 액세스할 수 있는 시간은 지정된 위치 및 유형에 따라 달라집니다.

Local 변수

함수나 프로시저에서 선언한 변수는 Local 변수라고 부릅니다.

```
Sub Test
      Dim MyInteger As Integer

      ' ...
End Sub
```

Local 변수는 해당 함수나 프로시저가 실행 중인 동안에만 유효하며, 그 다음에는 0 으로 다시 설정됩니다. 함수를 호출하면 이전에 생성한 값을 사용할 수 없습니다.

이전 값을 유지하려면 해당 변수를 `Static` 으로 지정해야 합니다.

```
Sub Test
      Static MyInteger As Integer

      ' ...

End Sub
```

VBA 와 달리 StarSuite Basic 은 Local 변수의 이름이 모듈 헤더에서 Global 변수나 Private 변수의 이름으로 동시에 사용되지 않게 합니다. VBA 응용 프로그램을 StarSuite Basic 으로 포팅할 때 중복되는 변수 이름은 바꿔 주어야 합니다.

Public Domain 변수

Public Domain 변수는 모듈의 헤더 부분에서 `Dim` 키워드를 사용하여 지정합니다. 이 변수는 라이브러리의 모든 모듈에서 사용할 수 있습니다.

모듈 A:

```
Dim A As Integer

Sub Test
      Flip
      Flop
End Sub

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

모듈 B:

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

변수 `A`의 값은 `Test` 함수에 의해 바뀌지 않지만 `Flip` 함수에 의해 1 만큼 증가하며 `Flop` 함수에 의해서는 1 만큼 감소합니다. 두 변수 바꾸기 모두 전역에 걸쳐 적용됩니다.

또한 키워드 `Public`을 `Dim` 대신 사용하여 Public Domain 변수를 선언할 수 있습니다.

```
Public A As Integer
```

Public Domain 변수는 관련 매크로가 실행 중인 동안에만 사용 가능하며 그 후에는 다시 설정됩니다.

Global 변수

함수 측면에서 Global 변수는 Public Domain 변수와 유사하지만, 관련 매크로 실행이 끝난 후에도 그 값이 유지된다는 점은 다릅니다. Global 변수는 모듈의 헤더 부분에서 키워드 `Global`을 사용하여 선언합니다.

```
Global A As Integer
```

Private 변수

Private 변수는 자신이 지정된 모듈에서만 사용할 수 있습니다. 키워드 Private 을 사용하여 변수를 지정합니다.

```
Private MyInteger As Integer
```

여러 모듈에 같은 이름의 Private 변수가 있으면 StarSuite Basic 은 각 항목마다 다른 변수를 만듭니다. 다음의 예에서 모듈 A 와 B 모두 C 라는 이름의 Private 변수를 가지고 있습니다. Test 함수는 먼저 모듈 A 에서 Private 변수를 설정한 다음, 모듈 B 에서 Private 변수를 설정합니다.

모듈 A:

```
Private C As Integer

Sub Test
      SetModuleA            ' 모듈 A 의 변수 C 설정
      SetModuleB            ' 모듈 B 의 변수 C 설정

      ShowVarA              ' 모듈 A 의 변수 C 표시(= 10)
      ShowVarB              ' 모듈 B 의 변수 C 표시(= 20)
End Sub

Sub SetmoduleeA
      A = 10
End Sub

Sub ShowVarA
      MsgBox C              ' 모듈 A 의 변수 C 표시
End Sub
```

모듈 B:

```
Private C As Integer

Sub SetModuleB
      A = 20
End Sub

Sub ShowVarB
      MsgBox C              ' 모듈 B 의 변수 C 표시
End Sub
```

상수

StarSuite Basic 에서는 키워드 `Const` 를 사용하여 상수를 선언합니다.

```
Const A = 10
```

필요하면 선언에서 상수의 유형도 지정할 수 있습니다.

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

연산자

StarSuite Basic 은 일반적인 수학, 논리, 비교 연산자를 지원합니다.

수학 연산자

수학 연산자는 모든 숫자 유형에 적용할 수 있으며, **+** 연산자는 문자열을 연결하는 데에도 사용할 수 있습니다.

`+` 수와 데이터 값의 덧셈, 문자열 연결

`-` 수와 데이터 값의 뺄셈

`*` 수의 곱셈

`/` 수의 나눗셈

`\` 정수 결과로 표시한 수의 나눗셈(반올림)

`^` 수의 거듭 제곱

`MOD` 모듈 연산(나눗셈의 나머지 계산)

논리 연산자

논리 연산자는 Boolean 대수 규칙에 따라 요소를 연결하도록 지원합니다. Boolean 값에 연산자를 적용하면 연결은 그 결과를 직접 제공합니다. Integer 및 Long Integer 값과 함께 사용할 경우 비트 수준에서 연결이 수행됩니다.

`AND` And 연결

`OR` Or 연결

`XOR` Exclusive Or 연결

`NOT` 부정

`EQV` 동일성 검사(양쪽 모두 `True` 이거나 `False`)

`IMP` 논리적 포함(첫 번째 식이 True 이면 두 번째 식 역시 True 가 되어야 함)

비교 연산자

비교 연산자는 숫자, 날짜 세부 정보, 문자열, Boolean 값 등의 모든 기본 변수 유형에 적용할 수 있습니다.

`=` 숫자, 날짜 값, 문자열의 일치

`<>` 숫자, 날짜 값, 문자열의 불일치

`>` 숫자, 날짜 값, 문자열이 더 큰지 검사

`>=` 숫자, 날짜 값, 문자열이 같거나 큰지 검사

`<` 숫자, 날짜 값, 문자열이 더 작은지 검사

`<=` 숫자, 날짜 값, 문자열이 같거나 작은지 검사

StarSuite Basic 은 VBA 의 `Like` 비교 연산자를 지원하지 않습니다.

분기

분기문을 사용하여 특정 조건을 만족시킬 때까지 코드 블록의 실행을 제한할 수 있습니다.

If...Then...Else

가장 많이 사용하는 분기문은 다음의 예와 같은 `If` 문입니다.

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

`B = 2` 대입은 변수 `A` 의 값이 3 보다 클 경우에만 수행됩니다. `If` 문을 변형한 `If/Else` 절도 사용합니다.

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

위의 예에서 변수 `B` 의 값은 `A` 가 3 보다 클 경우에만 2 가 되며, 그렇지 않으면 `B` 의 값은 0 이 됩니다.

If 문을 단계적으로 적용하여 다음과 같이 더 복잡한 문장을 작성할 수 있습니다.

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

변수 A의 값이 0과 같으면 B의 값은 0이 됩니다. A가 (0이 아니면서) 3보다 작으면 B의 값은 1이 됩니다. 그밖의 모든 경우에(즉 A가 3보다 크거나 같을 때) B 의 값은 2가 됩니다.

Select...Case

Select...Case 명령은 If 종속문 대신 사용할 수 있으며, 다양한 조건에서 임의의 값을 검사할 때 사용합니다.

```
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
```

위의 예에서 요일 이름이 숫자에 대응하기 때문에 DayOfWeek 변수의 값은 Sunday 일 때 1, Monday 일 때 2 와 같은 방식으로 할당됩니다.

Select 명령은 단순한 1:1 대입에 국한되지 않습니다. Case 분기에 비교 연산자나 일련의 식을 지정할 수 있습니다. 다음의 예는 가장 중요한 변형 구문을 소개합니다.

```
Select Case Var
Case 1 To 5

      ' ... Var 은 1~5 의 숫자

Case 6, 7, 8

      ' ... Var 은 6, 7 또는 8

Case Var > 8 And Var < 11

      ' ... Var 은 8 보다 크고 11 보다 작음

Case Else

      ' ... 그 밖의 모든 경우

End Select
```

루프(Loop)

루프는 지정한 패스 수 만큼 코드 블록을 실행합니다. 또한 패스의 수를 지정하지 않고 루프를 사용할 수 있습니다.

For...Next

For...Next 루프는 패스의 수가 고정되어 있습니다. 루프 카운터는 루프가 실행될 횟수를 지정합니다. 다음의 예에서

```
Dim I

For I = 1 To 10

      ' ... 루프 내부

Next I
```

변수 I 는 초기값이 1 인 루프 카운터입니다. 각 패스가 끝날 때마다 카운터는 1 만큼 증가합니다. 변수 I 가 10 이 되면 루프가 중단됩니다.

패스가 끝날 때마다 1 이 아닌 값만큼 루프 카운터를 늘리려면 Step 함수를 사용합니다.

```
Dim I

For I = 1 To 10 Step 0.5

      ' ... 루프 내부

Next I
```

위의 예에서는 패스가 끝날 때마다 카운터가 0.5 만큼 증가하므로 루프는 총 19 회 실행됩니다.

또한 Step 값에도 음수를 사용할 수 있습니다.

```
Dim I

For I = 10 To 1 Step -1

      ' ... 루프 내부

Next I
```

위의 예에서 카운터는 10 부터 시작하여 1 이 될 때까지 패스가 끝날 때마다 1 만큼 감소합니다.

Exit For 명령을 사용하면 For 루프를 미리 종료할 수 있습니다. 다음의 예에서 루프는 5 번째 패스에서 종료합니다.

```
Dim I

For I = 1 To 10

      If I = 5 Then
            Exit For
      End If

      ' ... 루프의 내부

Next I
```

VBA 에서 사용하는 For Each...Next 변형 루프는 StarSuite Basic 에서 지원하지 않습니다.

Do...Loop

`Do...Loop` 에서는 패스의 수가 고정되지 않습니다. `Do...Loop` 는 특정 조건을 만족시킬 때까지 실행됩니다. `Do...Loop` 는 4 가지 변형된 형태가 지원됩니다. 다음의 예에서는 `A > 10` 대신 어떤 조건이라도 사용할 수 있습니다.

1. `Do While...Loop` 형태

```
  Do While A > 10
      ' ... 루프 본문
  Loop
```

   패스하기 전에 해당 조건을 만족시키는지 검사하고 만족시킨 경우에만 루프를 실행합니다.

2. `Do Until...Loop` 형태

```
Do Until A > 10
      ' ... 루프 본문
Loop
```

   해당 조건을 더 이상 만족시킬 수 없을 때까지 루프를 실행합니다.

3. `Do...Loop While` 형태

```
Do
      ' ... 루프 본문
Loop While A > 10
```

   첫 번째 루프 패스 후 조건을 검사하고 해당 조건을 만족시킨 경우에는 루프를 종료합니다.

4. `Do...Loop Until` 형태

```
Do
      ' ... 루프 본문
Loop Until A > 10
```

   또한 첫 번째 패스 후 조건을 검사하지만, 해당 조건을 더 이상 만족시키지 않을 때까지 루프를 수행합니다.

`For...Next` 루프처럼 `Do...Loop` 도 종료 명령을 제공합니다. `Exit Do` 명령은 루프 내부의 어떤 위치에서도 루프를 종료할 수 있습니다.

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

      ' ... loop body
While A > 10
```

프로그래밍 예: 루프 포함 정렬

목록 검색, 값 계산, 복잡한 수학 작업 수행 등 다양한 용도로 루프를 사용할 수 있습니다. 다음 예는 이름별 목록 정렬에 루프를 사용하는 알고리즘입니다.

```
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
```

위에서 값은 오름차순으로 최종 정렬될 때까지 짝을 이루어 여러 번 교환됩니다. 변수가 적절한 위치를 향해 점진적으로 이전하는 모습이 흡사 거품과 같습니다. 이런 이유로 이 알고리즘을 *Bubble Sort* 라고 부르기도 합니다.

프로시저 및 함수

프로시저와 함수는 프로그램 구조상 중요한 위치를 차지합니다. 즉 복잡한 문제를 여러 하위 작업으로 구분하는 프레임워크를 제공합니다.

프로시저

*프로시저*는 명시적 값을 제공하지 않으면서 작업을 수행합니다. 구문은 다음과 같습니다.

```
Sub Test

      ' ... 실제 프로시저 코드 부분

End Sub
```

위의 예는 프로그램의 어떤 위치에서도 액세스 가능한 코드를 포함하는 `Test` 라는 이름의 프로시저를 지정합니다. 프로그램의 해당 위치에 프로시저의 이름을 입력하여 호출합니다.

```
Test
```

함수

프로시저처럼 *함수*는 실행할 프로그램들을 하나의 논리적 단위로 결합합니다. 그러나 프로시저와 달리 함수는 결과값을 제공합니다.

```
Function Test

      ' ... 실제 함수 코드 부분

      Test = 123
End Function
```

결과값은 간단한 대입을 사용하여 할당합니다. 대입은 반드시 함수의 끝에 위치할 필요는 없으며, 함수상의 어떤 위치라도 가능합니다.

위의 함수는 다음과 같이 프로그램 내부에서 호출할 수 있습니다.

```
Dim A

A = Test
```

이 코드에서는 변수 `A`를 지정하고 `Test` 함수의 값을 그 변수에 할당합니다.

결과값은 함수 내부에서 여러 차례 덮어쓸 수 있습니다. 일반적인 변수 대입과 마찬가지로 위의 예에서 함수는 마지막으로 할당된 값을 구합니다.

```
Function Test

      Test = 12

      ' ...

      Test = 123

End Function
```

위의 예에서 함수의 결과값은 123 입니다.

대입이 중단되면 함수는 결과값으로 0 을 구합니다(숫자값에 대해서는 숫자 0, 문자열에 대해서는 비어 있음).

함수의 결과값은 어떤 유형이라도 가능합니다. 결과값의 유형은 변수 선언과 똑같게 선언합니다.

```
Function Test As Integer

      ' ... 실제 함수 코드 부분

End Function
```

명시적 값 지정이 중단되면 결과값은 Variant 로 할당됩니다.

프로시저 및 함수의 조기 종료

StarSuite Basic 에서는 오류 처리 등을 위해 Exit Sub 와 Exit Function 명령을 사용하면 프로시저 또는 함수를 조기에 종료할 수 있습니다. 이 명령은 프로시저나 함수를 중단하고 프로그램에서 해당 프로시저나 함수가 호출했던 위치로 돌아갑니다.

다음의 예는
ErrorOccured 변수가 True 값을 갖게 되면 구현이 중단되는 프로시저를 보여줍니다.

```
Sub Test
      Dim ErrorOccured As Boolean

      ' ...

      If ErrorOccured Then
            Exit Sub
      End If

      ' ...

End Sub
```

매개 변수 전달

함수와 프로시저는 하나 이상의 매개 변수를 받을 수 있습니다. 기본 매개 변수는 함수나 프로시저 이름 다음에 괄호로 묶어 표시해야 합니다. 다음의 예는

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

Integer 값 A 와 String B 를 매개 변수로 사용하도록 프로시저를 지정합니다.

StarSuite Basic 에서는 일반적으로 참조*(Reference)*를 통해 매개 변수를 전달합니다. 프로시저나 함수 종료 시 변수 변경 사항은 보존됩니다.

```
Sub Test
      Dim A As Integer
      A = 10
      ChangeValue(A)
      ' 이제 매개 변수 A 의 값은 20
End Sub
Sub ChangeValue(TheValue As Integer)
       TheValue = 20
End Sub
```

위의 예에서는 `Test` 함수에서 지정한 값 `A` 가 `ChangeValue` 함수에게 매개 변수 형태로 전달됩니다. 그리고 나서 그 값은 20 으로 바꿔 `TheValue` 에게 전달하고, 함수 종료 시 이 값을 보존합니다.

또한 이후 매개 변수가 바뀌더라도 원래 전달된 값에 영향을 주지 않게 하려면 값의 형태로 매개 변수를 전달할 수 있습니다. 매개 변수를 값의 형태로 전달하려면 함수 헤더의 변수 선언 앞에 `ByVal` 키워드를 사용해야 합니다.

위의 예에서 `ChangeValue` 함수를 다음 함수로 바꾸더라도

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

상위 변수 `A` 는 바뀌지 않습니다. `ChangeValue` 함수 호출 후 변수 A 는 `10` 이라는 값을 그대로 유지합니다.

StarSuite Basic 에서 프로시저 및 함수에 매개 변수를 전달하는 방식은 VBA 와 거의 유사합니다. 기본적으로 매개 변수는 참조를 통해 전달합니다. 매개 변수를 값의 형태로 전달하려면 `ByVal` 키워드를 사용합니다. 또한 VBA 에서는 참조를 통해 매개 변수를 전달하도록 강제하는 `ByRef` 키워드를 사용할 수 있습니다. StarSuite Basic 에서는 이미 기본적인 절차로 구현되었기 때문에 이 키워드를 지원하지 않습니다.

원칙적으로 StarSuite Basic 의 함수 및 프로시저는 `Public` 입니다. VBA 에서 사용하는 `Public` 및 `Private` 키워드를 StarSuite Basic 에서는 지원하지 않습니다.

선택적 매개 변수

함수와 프로시저는 필요한 모든 매개 변수를 호출 과정에서 전달해야 호출할 수 있습니다.

StarSuite Basic 에서는 선택적 매개 변수를 지정할 수 있습니다. 즉 호출 시 해당 값이 포함되지 않으면 StarSuite Basic 은 빈 매개 변수를 전달합니다. 다음의 예에서

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

End Sub
```

A 는 필수 매개 변수이지만, B 는 선택적 매개 변수입니다.

IsMissing 함수는 매개 변수를 전달했는지 또는 누락됐는지 여부를 검사합니다.

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

      ' 매개 변수 B 가 실제로 존재하는지 검사
      If Not IsMissing (B) Then
             B_Local = B            ' 매개 변수 B 존재
      Else
             B_Local = 0            ' 매개 변수 B 없음 -> 기본값은 0
      End If

      ' ... 실제 함수 시작

End Sub
```

위의 예는 우선 매개 변수 B 를 전달했는지 검사하고, 필요하면 같은 매개 변수를 내부 B_Local 변수로 전달합니다. 해당 매개 변수가 존재하지 않으면 전달된 매개 변수 대신 기본값(이번 경우에는 값 0)을 B_ Local 에 전달합니다.

StarSuite Basic 에서는 선택적 매개 변수의 기본값을 지정하는 VBA 옵션을 지원하지 않습니다.

StarSuite Basic 에서는 VBA 의 ParamArray 키워드를 지원하지 않습니다.

재귀

이제 StarSuite Basic 에서는 재귀가 허용됩니다. 재귀 프로시저나 함수는 특정 기본 조건이 만족되었음을 확인할 때까지 스스로를 호출할 수 있습니다. 기본 조건을 통해 함수를 호출하면 결과값을 얻습니다.

다음의 예에서는 재귀 함수를 사용하여 `42`, `-42`, `3.14` 의 계승값을 계산합니다.

```
Sub Main
   Msgbox CalculateFactorial(  42 ) ' 1,40500611775288E+51 표시
   Msgbox CalculateFactorial( -42 ) ' "Invalid number for factorial!" 표시
   Msgbox CalculateFactorial( 3.14 )      ' "Invalid number for factorial!" 표시
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
       ' This is the recursive call:
       CalculateFactorial = Number * CalculateFactorial( Number - 1 )
   Endif
End Function
```

위의 예에서는 기본 조건 `0! = 1` 을 만족시킬 때까지 `CalculateFactorial` 함수를 재귀 호출하여 42 의 계승값을 구합니다`. = 1`.

StarSuite Basic 의 재귀 수준은 현재 500 으로 제한되어 있습니다.

오류 처리

프로그래밍에서 오류의 수정 처리는 가장 많은 시간이 필요한 작업 중 하나입니다. StarSuite Basic 은 오류 처리를 간소화하는 다양한 도구를 제공합니다.

On Error 명령

`On Error` 명령은 모든 오류 처리에서 핵심적인 역할을 합니다.

```
Sub Test
      On Error Goto ErrorHandler

      ' ... 오류가 발생하는 동안 작업 수행

Exit Sub

ErrorHandler:

      ' ... 각 오류 처리 코드

End Sub
```

`On Error Goto ErrorHandler` 줄은 오류 발생 시 StarSuite Basic 이 처리하는 방식을 지정합니다. `Goto ErrorHandler` 는 StarSuite Basic 이 현재 프로그램을 종료하고 `ErrorHandler:` 코드를 실행하게 합니다.

Resume 명령

`Resume Next` 명령은 오류 처리기의 코드를 실행한 후 프로그램에서 오류가 발생한 위치 다음 줄부터 프로그램 수행을 계속합니다.

```
ErrorHandler:

      ' ... 개별 오류 처리 코드

      Resume Next
```

`Resume Proceed` 명령을 사용하면 오류 처리 후 바로 이동하여 프로그램 수행을 계속할 위치를 지정할 수 있습니다.

```
ErrorHandler:

      ' ... 개별 오류 처리 코드
      Resume Proceed

Proceed:

      ' ... 오류 발생 후 여기부터 프로그램 수행
```

오류가 발생하더라도 오류 메시지 없이 프로그램 수행을 계속하려면 다음 서식을 사용합니다.

```
Sub Test
      On Error Resume Next

      ' ... 오류가 발생할 때 수행하고 있을 작업

End Sub
```

`On Error Resume Next` 명령은 전역에 영향을 미치기 때문에 사용 시 주의해야 합니다. 자세한 내용은 구조적 오류 처리에 대한 팁을 참조하십시오.

오류 정보에 관한 쿼리

오류 처리 시 오류를 자세히 설명하고 오류가 발생한 위치와 이유를 파악해두면 유용합니다.

- `Err` 변수는 오류 발생 횟수를 나타냅니다
- `Error$` 변수는 오류에 대한 설명을 수록합니다.
- `Erl` 변수는 오류가 발생한 줄 번호를 나타냅니다.

다음과 같이 호출하면

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

메시지 창에 오류 정보가 어떻게 표시되는지 보여 줍니다.

VBA 에서는 `Err` 이라는 통계 개체에 오류 메시지를 요약하지만, StarSuite Basic 은 `Err`, `Error$`, `Erl` 변수를 제공합니다.

상태 정보는 프로그램에서 `Resume` 이나 `On Error` 명령이 나올 때까지 유효하며, 이 명령이 수행되면 상태 정보가 다시 설정됩니다.

VBA 에서는 `Err` 개체의 `Err.Clear` 방식이 오류 발생 후 오류 상태를 다시 설정합니다. StarSuite Basic 에서는 `On Error` 나 `Resume` 명령을 사용하면 됩니다.

구조적 오류 처리에 대한 팁

지정 명령 `On Error`, 그리고 결과 명령 `Resume` 은 모두 `Goto` 문의 변형입니다.

이 구성을 사용할 때 오류가 발생하지않도록 코드를 구성하려면 점프 명령 사용을 모니터해야 합니다.

`On Error Resume Next` 명령은 열려 있는 모든 오류 메시지를 해제하기 때문에 이 명령을 사용할 때 특히 유의해야 합니다.

가장 좋은 방법은 프로그램 내부에서 오류를 처리할 때 한 가지 방법만 사용하는 것입니다. 오류 처리와 실제 프로그램을 구분하고 오류 발생 후 원래의 코드로 바로 이동하지 않아야 합니다.

다음은 오류 처리 절차의 예입니다.

```
Sub Example

	' 함수 시작 시 오류 처리기 지정
	On Error Goto ErrorHandler

	' ... 실제 프로그램 코드 부분

	' 오류 처리 비활성화
	On Error Goto 0

	' 정규 프로그램 구현 끝
Exit Sub

' 오류 처리 시작 지점
ErrorHandler:

	' 오류 예상 여부 검사
	If Err = ExpectedErrorNo Then
		' ... 오류 처리
	Else
		' ... 예기치 않은 오류 경고
	End If

	On Error Goto 0                                ' 오류 처리 비활성화
```

```
End Sub
```

이 절차는 실제 코드 프로그램 다음에 오류 처리기를 지정하면서 시작합니다. 프로그램 코드 마지막 부분에서 `On Error Goto 0` 을 호출하면서 오류 처리를 비활성화하고 `Exit Sub` 명령을 통해 절차 구현을 종료합니다. 이때 주의할 것은 `End Sub` 와 혼동하지 말아야 한다는 것입니다.

위의 예에서는 오류 번호가 가상 `ExpectedErrorNo` 상수에 저장된 예상 번호와 일치하는지 먼저 검사하고 그 결과에 따라 오류를 처리합니다. 또 다른 오류가 발생하면 시스템은 경고를 출력합니다. 예기치 않은 오류를 감지할 수 있으므로 오류 번호를 검사하는 것이 중요합니다.

코드 마지막 부분에서 `On Error Goto 0` 을 호출하면 오류의 상태 정보, 즉 `Err` 시스템 변수의 오류 코드를 다시 설정하여 나중에 발생하는 오류도 확실히 인식할 수 있게 합니다.

3 장

StarSuite Basic 의 런타임 라이브러리

이후 섹션에서는 런타임 라이브러리의 주요 함수를 소개합니다.

변환 함수

특정 유형의 변수를 다른 유형의 변수로 바꿔야 하는 경우가 자주 있습니다.

암시적 및 명시적 유형 변환

변수의 유형을 바꾸는 가장 간단한 방법은 대입을 사용하는 것입니다.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

위의 예에서 변수 A 는 문자열, 변수 B 는 정수입니다. StarSuite Basic 에서는 변수 B 를 변수 A 에 대입하면서 변수 B 를 문자열로 변환합니다. 이 변환은 보기보다 훨씬 복잡한 과정을 거칩니다. 정수 B 는 작업 메모리에서 2 바이트의 Long 숫자 형식을 유지합니다. 한편 변수 A 는 문자열이므로 각 문자 또는 숫자마다 1 또는 2 바이트 길이의 값을 저장합니다. 따라서 B 의 내용을 A 로 복사하기 전에 B 는 A 의 내부 형식으로 변환되어야 합니다.

대부분의 다른 프로그래밍 언어와 달리 Basic 은 유형 변환을 자동으로 수행합니다. 그러나 이는 치명적인 결과를 초래할 수 있습니다. 간단하게 보이는 다음 코드 시퀀스를 면밀히 검토하면

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

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

결국 문제가 발생할 수 있음을 알 수 있습니다. Basic 인터프리터가 먼저 덧셈 결과를 계산한 다음 문자열로 변환하면 결과값은 String 2 가 됩니다.
한편 Basic 인터프리터가 시작 값 B 와 C 를 문자열로 변환하고 그 결과에 덧셈 연산자를 적용한다면 그 결과는 문자열 11 이 됩니다.

Variant 변수를 사용할 때에도 같은 문제가 발생할 수 있습니다.

```
Dim A
Dim B
Dim C

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

Variant 변수는 숫자와 문자열을 모두 포함할 수 있으므로 변수 A 에 숫자 2 또는 문자열 11 이 할당되는지 분명하지 않습니다.

암시적 유형 변환으로 인한 오류 원본은 Variant 데이터 유형을 사용하지 않는 등(재차 권장) 엄격한 프로그래밍 습관을 통해서만 방지할 수 있습니다.

암시적 유형 변환으로 인한 기타 오류를 방지하기 위해 StarSuite Basic 은 연산의 데이터 유형을 변환해야 할 시점을 지정할 때 사용할 수 있는 다양한 변환 함수를 제공합니다.

- **CStr(Var)** – 모든 데이터 유형을 문자열로 변환합니다.
- **CInt(Var)** – 모든 데이터 유형을 정수 값으로 변환합니다.
- **CLng(Var)** – 모든 데이터 유형을 Long 값으로 변환합니다.
- **CSng(Var)** – 모든 데이터 유형을 Single 값으로 변환합니다.
- **CDbl(Var)** – 모든 데이터 유형을 Double 값으로 변환합니다.
- **CBool(Var)** – 모든 데이터 유형을 Boolean 값으로 변환합니다.
- **CDate(Var)** – 모든 데이터 유형을 Date 값으로 변환합니다.

이러한 변환 함수를 사용하여 StarSuite Basic 이 유형 변환 작업을 수행하는 방식을 지정할 수 있습니다.

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

B = 1
C = 1

A = CInt(B + C)             ' B 와 C 를 먼저 더한 다음 변환
                            (숫자 2 생성)
A = CStr(B) + Cstr(C)       ' B 와 C 를 문자열로 변환한 다음
                            ' 결합(String "11" 생성)
```

위 예의 첫 번째 덧셈 과정에서 StarSuite Basic 은 먼저 정수 변수를 더한 다음 그 결과를 일련의 문자로 변환합니다. A 에는 String 2 가 할당됩니다. 두 번째 경우에서는 먼저 Integer 변수가 2 개의 문자열로 변환되고 대입을 통해 서로 연결됩니다. 따라서 A 에는 String 11 이 할당됩니다.

숫자 변환 함수인 CSng 와 CDbl 에서는 실수도 사용할 수 있습니다. 국가별 설정에서 지정한 기호를 소수점 기호로 사용해야 합니다. 한편 CStr 함수는 숫자, 날짜, 시간 세부 정보의 서식을 설정할 때 현재 선택한 국가별 설정을 사용합니다.

`Val` 함수는 `Csng, Cdbl`, `Cstr` 메서드와 다릅니다. 문자열을 숫자로 변환하지만, 항상 소수점 기호로 마침표를 사용해야 합니다.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)        ' 국가별 설정과 상관 없이 정확하게 변환됩니다
```

변수 내용 검사

Date 유형은 변환되지 않는 경우도 있습니다.

```
Dim A As String
Dim B As Date

A = "test"
B = A                    ' 오류 메시지 생성
```

이번 예에서 문자열 `test` 를 Date 유형의 변수에 대입하는 것은 아무런 의미가 없으므로 , Basic 인터프리터는 오류를 보고합니다. 문자열을 Boolean 변수에 할당하는 경우도 마찬가지입니다.

```
Dim A As String
Dim B As Boolean

A = "test"
B = A                    ' 오류 메시지 생성
```

역시 Basic 인터프리터는 오류를 보고합니다.

대입하기 전에 할당할 변수 내용이 대상 변수의 유형과 일치하는지 확인하기 위해 프로그램을 검사하면 그러한 오류 메시지를 방지할 수 있습니다.
StarSuite Basic 은 이러한 용도로 다음과 같은 테스트 함수를 제공합니다.

- **`IsNumeric(Value)`** – 해당 값이 숫자인지 여부를 확인합니다.
- **`IsDate(Value)`** – 값이 날짜인지 여부를 확인합니다.
- **`IsArray(Value)`** – 값이 행렬인지 여부를 확인합니다.

이 함수들은 사용자 입력을 쿼리할 때 특히 유용합니다. 예를 들어 사용자가 유효한 숫자나 날짜를 입력했는지 확인할 수 있습니다.

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

위의 예에서 `UserInput` 변수가 유효한 숫자 값을 갖는다면 `ValidInput` 변수로 할당됩니다. `UserInput` 이 유효한 숫자를 포함하지 않는다면 `ValidInput` 에 값 `0` 이 할당되고 오류 메시지가 표시됩니다.

Basic 에서 숫자, 날짜 세부 정보, 행렬을 검사하기 위해 테스트 함수가 있지만 Boolean 값을 검사하는 함수는 없습니다. 그러나 `IsBoolean` 함수를 사용하여 비슷한 기능을 구현할 수 있습니다.

```
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` 함수는 Boolean 유형의 내부 `Dummy` 도움말 변수를 지정하고 전달된 값에 할당하려고 합니다. 대입이 성공적으로 수행되면 함수는 `True` 를 구합니다. 실패하면 런타임 오류가 발생하여 테스트 함수가 오류를 구하지 않도록 차단합니다.

StarSuite Basic 에서는 숫자가 아닌 값을 포함하는 문자열을 숫자에 할당하면 StarSuite Basic 은 오류 메시지를 구하는 대신 해당 변수에 값 `0` 을 전달합니다. VBA 에서는 절차가 다릅니다. VBA 에서 그와 같은 대입이 실행되면 오류가 트리거되고 프로그램 구현이 종료합니다.

문자열

문자 집합 사용

StarSuite Basic 은 문자열을 관리할 때 유니코드 문자 집합을 사용합니다. Asc 와 Chr 함수는 임의의 문자에 해당되는 유니코드 값을 설정하고 임의의 유니코드 값에 해당하는 문자를 찾습니다. 다음 식은 다양한 유니코드 값을 code 변수에 할당합니다.

```
Code = Asc("A")              ' 라틴 문자 A (유니코드 값 65)
Code = Asc("€")              ' 유로 문자(유니코드 값 8364)
Code = Asc("Л")              ' 키릴 문자Л(유니코드 값 1083)
```

반대로 다음 식은

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

문자열 MyString 이 숫자 값 13 으로 초기화되고, 이는 줄바꿈을 나타냅니다.

Basic 언어에서 Chr 명령은 문자열에 제어 문자를 삽입할 때 자주 사용합니다.
따라서 다음 대입은

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

텍스트 앞에 탭 문자(유니코드 값 9)가 오며 텍스트 다음에 줄바꿈(유니코드 값 13)이 추가됩니다.

부분 문자열 액세스

StarSuite Basic 은 부분 문자열을 구하는 4 가지 함수를 제공합니다.

- **Left(MyString, Length)** – MyString 에서 최초 Length 개 문자를 구합니다 .
- **Right(MyString, Length)** – MyString 에서 마지막 Length 개 문자를 구합니다 .
- **Mid(MyString, Start, Length)** – Start 위치를 기준으로 MyString 에서 최초 Length 개의 문자를 구합니다 .
- **Len(MyString)** – MyString 의 문자 개수를 구합니다.

다음은 위의 함수를 호출하는 예입니다.

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

MyString = " This is a small test"

MyResult = Left(MyString,5)         ' 결과는 문자열 "This "
MyResult = Right(MyString, 5)               ' 결과는 String "test"
MyResult = Mid(MyString, 8, 5)              ' 결과는 String "a sma"
MyLength = Len(MyString)            ' 결과는 값 4
```

찾기 및 바꾸기

StarSuite Basic 은 문자열 내부에서 부분 문자열을 검색하는 `InStr` 함수를 제공합니다.

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

`SearchString` 매개 변수는 `MyString` 내부에서 검색할 문자열을 나타냅니다. 이 함수는 MyString 내부에서 `SearchString` 이 최초로 나타나는 위치를 숫자 값으로 구합니다. 해당 문자열의 또 다른 위치를 찾고 싶을 경우, 이 함수는 StarSuite Basic 이 검색을 시작할 위치를 지정하는 옵션을 제공합니다. 이번 예에서 함수의 구문은 다음과 같습니다.

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

위의 예에서 `InStr` 는 대/소문자를 구분하지 않습니다. `InStr` 이 대/소문자를 구분하도록 검색을 바꾸려면 다음 예와 같이 매개 변수 `0` 을 추가합니다.

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

프로그래머는 임의의 문자열 내부에서 다른 문자열을 찾아 바꾸는 등 위 함수를 사용하여 문자열을 편집할 수 있습니다.

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

  Result = ""
  StartPos = 1
  CurrentPos = 1

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

위 함수는 Source 에서 `Search` 를 찾기 위해 `InStr` 을 루프로 실행합니다. Search 를 찾으면 식을 실행하기 전에 그 부분을 분리한 다음 `Result` 반환 버퍼에 기록합니다. Search 위치에 `NewPart`

를 추가합니다. 더 이상 Search 를 찾을 수 없으면 이 함수는 문자열의 나머지 부분을 반환 버퍼에 추가합니다. 이런 식으로 만들어진 문자열이 바꾸기 프로세스의 결과가 됩니다.

문자 시퀀스의 바꾸기 부분은 가장 많이 사용하는 함수 중 하나이므로 StarSuite Basic 의 `Mid` 함수는 이 작업을 자동 수행하도록 확장되었습니다. 다음 예는

```
Dim MyString As String

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

MyString 의 6 번째 위치부터 3 개의 문자를 `is` 로 바꿉니다.

문자열(String) 형식 설정

`Format` 함수는 숫자를 문자열 형식으로 설정합니다. 이 작업을 위해 이 함수에서는 `Format` 식이 지정되어야 하며, 이 식은 숫자 표기 형식 설정의 서식 파일로 사용됩니다. 서식 파일 내부의 각 자리 표시자는 해당 항목의 서식이 출력 값에 적합하게 설정되게 합니다. 서식 파일에서 가장 중요한 5 가지 자리 표시자는 영(0), 파운드 기호(#), 마침표 (.), 쉼표 (,), 달러 기호($) 문자입니다.

서식 파일 내부의 *영(0)* 문자는 해당 위치에 항상 숫자가 놓이게 합니다. 숫자가 제공되지 않으면 그 위치에 0 이 표시됩니다.

마침표는 운영 체제의 국가별 설정에서 지정된 소수점 기호를 나타냅니다.

아래 예는 식에서 영*(0)*과 마침표 문자를 사용하여 소수점 아래 자릿수를 지정하는 방법을 보여줍니다.

```
MyFormat = "0.00"

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

같은 방식으로 숫자 앞에 영(0)을 추가하여 원하는 길이를 만들 수 있습니다.

```
MyFormat = "0000.00"

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

쉼표는 운영 체제가 천 단위 구분 기호로 사용하는 문자를, 파운드 기호는 입력 문자열이 필요로 할 경우에만 표시되는 자릿수나 위치를 나타냅니다.

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

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

```
MyString = Format(0.434, MyFormat)        ' 결과는 "0,43"
```

Format 함수는 자리 표시자 위치에 시스템이 지정한 통화 기호를 표시합니다.

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

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

VBA 에서 날짜 및 시간 세부 정보의 서식을 설정하는 데 사용하는 명령은 StarSuite Basic 에서 지원하지 않습니다.

Date 와 Time

StarSuite Basic 은 날짜 및 시간 세부 정보를 바이너리 형식으로 저장하는 `Date` 데이터 유형을 제공합니다.

프로그램 코드 상의 날짜 및 시간 세부 정보 지정

간단한 문자열 대입을 통해 날짜를 Date 변수에 할당할 수 있습니다.

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

StarSuite Basic 은 문자열로 지정된 날짜 값을 Date 변수로 자동 변환하므로 위 대입은 올바르게 수행될 수 있습니다. 그러나 이러한 유형의 대입은 오류를 발생시킬 수 있습니다. 날짜와 시간 값이 국가에 따라 다르게 지정되고 표시되기 때문입니다.

StarSuite Basic 은 문자열을 날짜 값으로 변환할 때 운영 체제의 국가별 설정을 사용하므로 위의 식은 국가별 설정이 문자열 식과 일치한 경우에만 올바르게 수행됩니다.

이러한 문제를 방지하기 위해 `DateSerial` 함수를 사용하여 Date 변수에 고정 값을 할당합니다.

```
Dim MyVar As Date

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

함수 매개 변수는 연도, 월, 일의 순서가 되어야 합니다. 이 함수는 국가별 설정과 상관 없이 변수가 정확한 값을 할당받게 합니다.

TimeSerial 함수는 DateSerial 함수가 날짜 서식을 설정하는 것과 같은 방식으로 시간 세부 정보의 서식을 설정합니다.

```
Dim MyVar As Date

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

이 매개 변수는 시, 분, 초의 순서가 되어야 합니다.

날짜 및 시간 세부 정보 추출

다음 함수는 DateSerial 및 TimeSerial 함수에 대응됩니다.

- **Day(MyDate)** – MyDate 에서 월의 일을 구합니다.
- **Month(MyDate)** – MyDate 에서 월을 구합니다.
- **Year(MyDate)** – MyDate 에서 연도를 구합니다.
- **Weekday(MyDate)** – MyDate 에서 평일의 번호를 구합니다.
- **Hour(MyTime)** – MyTime 에서 시간을 구합니다.
- **Minute(MyTime)** – MyTime 에서 분을 구합니다.
- **Second(MyTime)** – MyTime 에서 초를 구합니다.

이 함수는 지정된 Date 변수에서 날짜 및 시간 부분을 추출합니다. 다음 예에서는

```
Dim MyDate As Date

' ... Initialization of MyDate

If Year(MyDate) = 2003 Then

      ' ... 지정된 날짜는 2003 년
End If
```

MyDate 에 저장된 날짜가 2003 년인지 여부를 확인합니다. 이와 비슷하게 다음 예는

```
Dim MyTime As Date

' ... Initialization of MyTime

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

      ' ... 지정된 날짜는 2003 년
```

```
End If
```

`MyTime` 이 `12` ~ `14` 시간인지 여부를 확인합니다.

`Weekday` 함수는 전달된 날짜의 평일 번호를 구합니다.

```
Dim MyDate As Date
Dim MyWeekday As String

' ... initialize 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
```

참고: 일요일을 주의 시작 요일로 간주합니다.

시스템 날짜 및 시간 구하기

다음 함수는 StarSuite Basic 에서 시스템 시간 및 날짜 정보를 구할 때 사용합니다.

- **`Date`** – 현재 날짜를 구합니다.
- **`Time`** – 현재 시간을 구합니다.
- **`Now`** – 현재 시점(날짜와 시간이 결합된 값)을 구합니다.

파일과 디렉토리

파일 작업은 응용 프로그램의 기본적인 작업 중 하나입니다. StarSuite API 는 StarSuite 문서를 만들고 열며 수정할 때 사용할 수 있는 모든 개체를 제공합니다. 이러한 개체에 대해서는 4 장에서 자세히 설명합니다. 그러나 때로는 파일 시스템을 직접 액세스하고 디렉토리를 찾거나 텍스트 파일을 편집해야 하는 경우가 있습니다. StarSuite Basic 의 런타임 라이브러리는 이러한 작업을 위해 몇 가지 기본적인 함수를 제공합니다.

일부 DOS 전용 파일 및 디렉토리 함수는 StarSuite 6.x 에서 더 이상 제공되지 않거나 그 기능이 제한됩니다.
예를 들어 `ChDir`, `ChDrive`, `CurDir` 함수는 지원하지 않습니다.

일부 DOS 전용 등록 정보는 숨김 파일과 시스템 파일을 구분하기 위한 경우처럼 파일 등록 정보를 매개 변수로 사용하는 함수에서 더 이상 사용되지 않습니다. 이러한 변경 조치는 StarSuite 의 플랫폼 독립성 수준을 최대한 높이기 위해 필요합니다.

파일 관리

디렉토리 검색

StarSuite Basic 의 `Dir` 함수는 디렉토리에서 파일 및 하위 디렉토리를 검색하는 기능을 수행합니다. 최초 요청 시 검색 대상 디렉토리 경로를 포함한 문자열이 `Dir` 의 첫 번째 매개 변수로 할당되어야 합니다. `Dir` 의 두 번째 매개 변수는 검색할 파일 또는 디렉토리를 지정합니다. StarSuite Basic 은 발견된 첫 번째 디렉토리 항목의 이름을 구합니다. 다음 항목을 찾으려면 `Dir` 함수를 매개 변수 없이 요청해야 합니다. `Dir` 함수는 더 이상 항목을 찾지 못하면 빈 문자열을 구합니다.

다음 예는 `Dir` 함수를 사용하여 어떤 디렉토리에 위치한 모든 파일을 요청하는 방법을 소개합니다. 이 절차는 개별 파일의 이름을 `AllFiles` 변수에 저장하고 이를 메시지 상자에 표시합니다.

```
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
```

`Dir` 함수에서 두 번째 매개 변수로 영(0)을 사용하면 `Dir` 은 파일의 이름만 구하며 디렉토리는 무시합니다. 다음과 같이 매개 변수를 지정할 수 있습니다.

- `0` : 일반 파일을 구합니다.
- `16` : 하위 디렉토리를 구합니다.

다음 예는 사실상 위의 예와 같지만 `Dir` 함수의 매개 변수로 16 을 전달하므로 파일 이름 대신 해당 폴더의 하위 디렉토리를 구합니다.

```
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
```

StarSuite Basic 에서는 VBA 와 달리 `Dir` 함수가 `16` 을 매개 변수로 사용하면 해당 폴더의 하위 디렉토리만 구합니다 VBA 에서 이 함수는 표준 파일의 이름도 구하기 때문에 디렉토리만 구하기 위해서는 추가적인 검사가 필요합니다.

VBA 에서 *숨김*, *시스템 파일*, *보관* 및 *볼륨 이름* 등록 정보를 갖는 파일을 찾아 디렉토리를 검색하는 옵션은 StarSuite Basic 에서 지원하지 않습니다. 해당되는 파일 시스템 함수가 일부 운영 체제에서 지원되지 않기 때문입니다.

VBA 와 StarSuite Basic 에서는 `Dir` 경로 지정에 `*` 및 `?` 자리 표시자를 사용할 수 있습니다. 그러나 StarSuite Basic 에서는 VBA 와 달리 `*` 자리 표시자를 파일 이름이나 파일 확장자의 마지막 문자로만 사용할 수 있습니다.

디렉토리 생성 및 삭제

StarSuite Basic 은 디렉토리를 만드는 `MkDir` 함수를 제공합니다.

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

이 함수는 디렉토리 및 하위 디렉토리를 만듭니다. 필요하다면 계층 구조 안에 필요한 모든 디렉토리를 만들 수 있습니다. 예를 들어 `C:\SubDir1` 디렉토리만 존재할 경우 다음과 같이 호출하면

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

`C:\SubDir1\SubDir2` 디렉토리 및 `C:\SubDir1\SubDir2\SubDir3` 디렉토리를 모두 만듭니다.

`RmDir` 함수는 디렉토리를 삭제합니다.

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

해당 디렉토리가 하위 디렉토리나 파일을 포함하고 있다면 이들 역시 *삭제됩니다*. 따라서 `RmDir` 을 사용할 때 주의해야 합니다.

VBA 에서는 `MkDir` 및 `RmDir` 함수가 현재 디렉토리만 대상으로 합니다. 한편 StarSuite Basic 에서는 `MkDir` 및 `RmDir` 를 사용하여 여러 단계의 디렉토리를 만들거나 삭제할 수 있습니다.

VBA 에서는 디렉토리가 파일을 포함하고 있으면 `RmDir` 이 오류 메시지를 표시합니다. StarSuite Basic 에서는 해당 디렉토리 및 *그 디렉토리의 모든 파일을* 삭제합니다.

파일 복사, 이름 바꾸기, 삭제 및 존재 확인

다음과 같이 호출하면

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

Source 파일의 복사본을 Destination 이라는 이름으로 만듭니다.

다음 함수를 사용하여

```
Name OldName As NewName
```

OldName 파일의 이름을 NewName 으로 바꿀 수 있습니다. As 키워드 구문 그리고 쉼표를 사용하지 않는다는 점은 Basic 언어의 기본 특성과 같습니다.

다음과 같이 호출하면

```
Kill(Filename)
```

Filename 파일을 삭제합니다. (파일을 포함하여) 디렉토리를 삭제하려면 RmDir 함수를 사용합니다.

FileExists 함수는 파일의 존재 여부를 확인할 때 사용할 수 있습니다.

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

파일 등록 정보 읽기 및 바꾸기

파일 작업 시 파일 등록 정보, 파일의 마지막 변경 시간 및 파일 길이를 설정하는 것이 중요할 때가 있습니다.

다음과 같이 호출하면

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

파일에 대한 몇 가지 등록 정보를 구합니다. 결과값은 다음 값을 사용하는 비트 마스크 형태로 제공됩니다.

- 1 : 읽기 전용 파일
- 16 : 디렉토리 이름

다음 예는

```
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
```

`test.txt` 파일의 비트 마스크를 확인하여 읽기 전용인지 디렉토리인지 식별합니다. 두 가지 중 어느 것도 해당되지 않으면 `FileDescription` 에는 "일반 "문자열이 할당됩니다.

VBA 에서 *숨김*, *시스템 파일*, *보관* 및 *볼륨 이름* 파일 등록 정보를 쿼리할 때 사용하는 플래그는 StarSuite Basic 에서 지원하지 않습니다. 이들은 Windows 전용이며 다른 운영 체제에서는 전혀 사용할 수 없거나 일부만 사용할 수 있기 때문입니다.

`SetAttr` 함수를 사용하여 파일의 등록 정보를 바꿀 수 있습니다. 다음과 같이 호출하면

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

파일을 읽기 전용 상태로 만들 수 있습니다. 다음과 같이 호출하면 기존의 읽기 전용 상태를 삭제할 수 있습니다.

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

파일의 마지막 수정 날짜 및 시간은 `FileDateTime` 함수가 제공합니다. 여기에서 날짜 서식은 시스템의 국가별 설정을 따릅니다.

```
FileDateTime("test.txt")      ' 결과값은 마지막으로 파일을 수정한 날짜 및 시간
```

`FileLen` 함수는 파일의 길이를 바이트 단위로 나타냅니다(정수 값 범위에서).

```
FileLen("test.txt")           ' 결과값은 바이트 단위의 파일 길이
```

텍스트 파일 쓰기 및 읽기

StarSuite Basic 은 파일을 읽고 쓰는 데 필요한 모든 메서드를 제공합니다. 다음 설명은 텍스트 파일 작업에 해당됩니다(텍스트 문서 *아님*).

텍스트 파일 쓰기

텍스트 파일은 액세스하기 전에 먼저 열어야 합니다. 파일을 열기 위해서는 빈 파일 핸들이 필요하며, *이 파일 핸들은* 이후 파일을 액세스할 때 해당 파일을 식별합니다.

`FreeFile` 함수를 사용하여 빈 파일 핸들을 만들 수 있습니다. 이 핸들은 파일 열기를 수행하는 `Open` 명령의 매개 변수로 사용합니다. 파일을 텍스트 파일로 지정하기 위해 열려면 다음과 같이 `Open` 을 호출합니다.:

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

`Filename` 은 파일의 이름을 포함하는 문자열입니다. `FileNo` 는 `FreeFile` 함수가 만든 핸들입니다.

파일이 열리면 `Print` 명령을 사용하여 줄 단위로 기록할 수 있습니다.

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

`FileNo` 는 여기에서도 파일 핸들을 나타냅니다. 두 번째 매개 변수는 텍스트 파일에 한 줄로 저장될 텍스트를 나타냅니다.

쓰기 작업을 완료했으면 `Close` 를 호출하여 파일을 닫아야 합니다.

```
Close #FileNo
```

여기에서도 파일 핸들을 지정해야 합니다.

다음 예는 텍스트 파일을 열고 기록하며 닫는 방법을 소개합니다.

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

Filename = "c:\data.txt"                        ' 파일 이름 지정
FileNo = Freefile                     ' 빈 파일 핸들 설정

Open Filename For Output As #FileNo             ' 파일 열기(쓰기 모드)
Print #FileNo, "This is a line of text"         ' 줄 저장
Print #FileNo, "This is another line of text"   ' 줄 저장
Close #FileNo                                   ' 파일 닫기
```

텍스트 파일 읽기

텍스트 파일은 쓰기와 같은 방식으로 읽습니다. 파일을 여는 `Open` 명령은 `For Input` 식을 `For Output` 식 대신 사용하고, 데이터를 기록하는 `Print` 명령 대신 `Line Input` 명령을 사용하여 데이터를 읽습니다.

마지막으로 텍스트 파일 호출 시 다음 명령을 사용하여

```
eof(FileNo)
```

파일 끝에 도달했는지 여부를 확인합니다.

다음 예는 텍스트 파일을 읽는 방법을 소개합니다.

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

' 파일 이름 지정
Filename = "c:\data.txt"

' 빈 파일 핸들 설정
FileNo = Freefile

' 파일 열기(읽기 모드)
Open Filename For Input As FileNo

' 파일 끝에 도달했는지 여부 확인
Do While not eof(FileNo)

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

Loop

' 파일 닫기
Close #FileNo

Msgbox Msg
```

각 줄은 `Do While` 루프에서 읽혀지고 `Msg` 변수에 저장되며 메시지 상자 끝에 표시됩니다.

메시지 상자 및 입력란

StarSuite Basic 은 간단한 사용자 통신을 지원하기 위해 `MsgBox` 및 `InputBox` 함수를 제공합니다.

메시지 표시

`MsgBox` 는 간단한 정보 상자를 표시하며 하나 이상의 버튼을 포함할 수 있습니다. 가장 간단한 형태는 다음과 같습니다.

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

이 MsgBox 는 텍스트 및 확인 버튼으로만 구성됩니다.

정보 상자의 모양은 매개 변수를 사용하여 바꿀 수 있습니다. 매개 변수는 버튼 추가, 미리 지정된 버튼 지정, 정보 기호 추가와 같은 옵션을 제공합니다. The values for selecting the buttons are:

- 0 – [확인] 버튼
- 1 – [확인] 및 [취소] 버튼
- 2 – [취소] 및 [다시 시도] 버튼
- 3 – [예], [아니오], [취소] 버튼
- 4 – [예] 및 [아니오] 버튼
- 5 – [다시 시도] 및 [취소] 버튼

임의의 버튼을 기본 버튼으로 설정하려면 버튼 선택 목록에서 매개 변수 값에 다음 값 중 하나를 더합니다. 예를 들어 [예], [아니오], [취소] 버튼을 만들고(값 3) [취소] 버튼을 기본으로 설정하려면 (값 512) 매개 변수 값은 3 + 512 = 515 가 됩니다.

- 0 – 첫 번째 버튼이 기본값
- 256 – 두 번째 버튼이 기본값
- 512 – 세 번째 버튼이 기본값

마지막으로 다음과 같은 정보 기호를 사용할 수 있으며, 관련 매개 변수 값을 더하여 표시할 수 있습니다.

- 16 – 중지 기호
- 32 – 물음표
- 48 – 느낌표
- 64 – 팁 아이콘

다음과 같이 호출하면

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

[예], [아니오] 버튼이 있고(값 4) 두 번째 버튼인 [아니오]가 기본값이며(값 256) 물음표를 받는(값 32) 정보 상자가 표시됩니다(4+256+32=292).

정보 상자가 여러 버튼을 포함하고 있다면 어느 버튼을 눌렀는지 확인하기 위해 결과값을 쿼리해야 합니다. 이 경우에는 다음과 같은 결과값을 사용할 수 있습니다.

- 1 – [확인]
- 2 – [취소]
- 4 – [다시 시도]
- 5 – [무시]
- 6 – [예]
- 7 – [아니오]

이전 예에서 다음과 같이 결과값을 확인할 수 있습니다.

```
If MsgBox ("Do you want to continue?", 292) = 6 Then
      ' Yes button pressed
Else
      ' No button pressed
End IF
```

정보 상자를 구성하는 정보 텍스트 및 매개 변수 외에도 MsgBox 에서는 상자 제목의 텍스트를 지정하는 세 번째 매개 변수를 사용할 수 있습니다.

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

상자 제목을 지정하지 않으면 기본값은 “soffice”가 됩니다.

간단한 문자열 쿼리를 위한 입력란

InputBox 함수는 사용자의 간단한 문자열을 쿼리합니다. 따라서 대화 상자 대신 간단하게 사용할 수 있습니다. InputBox 는 3 가지 표준 매개 변수, 즉

- 정보 텍스트,
- 상자 제목,
- 입력 범위에서 덧셈 가능한 기본값을 받습니다.

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

결과값으로 InputBox 는 사용자가 입력한 문자열을 제공합니다.

기타 함수

Beep

Beep 함수는 사용자의 올바르지 않은 작업에 대해 경고할 수 있도록 시스템에서 사운드를 재생하게 합니다. Beep 는 매개 변수를 갖지 않습니다.

```
Beep  ' 알림 톤으로 재생
```

Shell

Shell 함수를 사용하여 외부 프로그램을 시작할 수 있습니다.

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

Pathname 은 실행할 프로그램의 경로를 지정합니다. Windowstyle 은 프로그램을 시작할 창을 지정합니다. 다음 값을 사용할 수 있습니다.

- 0 – 프로그램이 포커스를 받고 숨겨진 창에서 시작합니다.
- 1 – 프로그램이 포커스를 받고 보통 크기의 창에서 시작합니다.
- 2 – 프로그램이 포커스를 받고 최소화된 창에서 시작합니다.
- 3 – 프로그램이 포커스를 받고 최대화된 창에서 시작합니다.
- 4 – 프로그램은 포커스를 받지 않고 보통 크기의 창에서 시작합니다.
- 6 – 프로그램이 최소화된 창에서 시작하며 포커스는 현재 창에 남아 있습니다.
- 10 – 프로그램이 전체 화면 모드에서 시작합니다.

세 번째 매개 변수인 Param 은 시작할 프로그램으로 명령줄 매개 변수를 전달합니다.

Wait

Wait 함수는 지정된 시간 동안 프로그램 실행을 중단합니다. 대기 시간은 밀리초 단위로 지정합니다.

```
Wait 2000
```

명령은 2 초(2000 밀리초) 동안 중단합니다.

Environ

Environ 함수는 운영 체제의 환경 변수를 구합니다. 시스템 및 구성에 따라 다양한 데이터 유형이 저장됩니다. 다음과 같이 호출하면

```
Dim TempDir

TempDir=Environ ("TEMP")
```

운영 체제 임시 디렉토리의 환경 변수를 확인합니다.

4 장

StarSuite API 소개

StarSuite API 는 StarSuite 에 액세스하는 범용 프로그래밍 인터페이스입니다. StarSuite API 를 사용하여 StarSuite 문서를 만들고 기존 문서를 열거나 수정하거나 인쇄할 수 있습니다. 개인 매크로를 통해 StarSuite 의 기능 범위를 확장하고 개인 대화 상자를 만드는 옵션을 제공합니다.

StarSuite API 는 StarSuite Basic 뿐 아니라 Java, C++와 같은 프로그래밍 언어와도 함께 사용할 수 있습니다. 이는 다양한 프로그래밍 언어에 대한 인터페이스를 제공하는 *UNO(Universal Network Object)*라는 기술을 통해 가능합니다.

이 장에서는 StarSuite Basic 에서 UNO 를 활용하면서 StarSuite 를 어떻게 사용할 수 있는지 중점적으로 알아 봅니다. StarSuite Basic 프로그래머의 관점에서 UNO 의 주요 개념을 설명합니다. StarSuite API 의 다양한 부분의 사용 방법에 대해서는 다음 장부터 자세히 다룹니다.

UNO(Universal Network Object)

StarSuite 는 UNO 형태의 프로그래밍 인터페이스를 제공합니다. 이는 개체 지향 프로그래밍 인터페이스로서, StarSuite 는 StarSuite 패키지에 대해 프로그램 제어 액세스를 보장하는 다양한 개체로 나뉩니다.

StarSuite Basic 은 절차 프로그래밍 언어이기 때문에 UNO 를 사용하기 위해서는 몇 가지 언어 구성소를 추가해야 합니다.

StarSuite Basic 에서 UNO 를 사용하려면 관련 개체에 대한 변수 선언이 필요합니다. 이 선언은 `Dim` 명령을 사용하여 수행됩니다(2참조). `Object` 유형 지정을 사용하여 개체 변수를 선언해야 합니다.

```
Dim Obj As Object
```

위와 같이 호출하면 `Obj` 라는 이름의 개체 변수를 선언합니다.

생성된 개체 변수는 초기화해야 사용할 수 있습니다. 이 작업은 `createUnoService` 함수를 사용하여 수행할 수 있습니다.

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

위와 같이 호출하면 새로 만든 개체에 대한 참조를 `Obj` 변수에 할당합니다. `com.sun.star.frame.Desktop` 은 개체 유형과 비슷하지만, UNO 용어로는 유형 대신 *"서비스"라고* 부릅니다. UNO 규칙에 따라 `Obj` 는 `com.sun.star.frame.Desktop` 서비스를 지원하는 개체에 대한 참조로 설명합니다. 따라서 StarSuite Basic 에서 사용하는 "서비스"라는 용어는 다른 프로그래밍 언어의 *유형* 및 *클래스*에 해당됩니다.

그러나 한 가지 중요한 차이점이 있습니다. UNO 는 동시에 여러 서비스를 지원할 수 있습니다. 일부 UNO 서비스는 다른 서비스를 지원하기 때문에 하나의 개체를 통해 전체 서비스가 제공됩니다. 예를 들어 이미 언급한 `com.sun.star.frame.Desktop` 서비스 기반의 개체 역시 문서를 로드하고 프로그램을 종료하는 다른 서비스를 포함합니다.

VBA 의 개체 구조는 해당 개체가 속한 클래스를 통해 지정하지만 StarSuite Basic 에서는 개체가 지원하는 서비스를 통해 구조를 지정합니다. VBA 개체는 언제나 정확히 하나의 클래스에 할당됩니다. 그러나 StarSuite Basic 개체는 여러 서비스를 지원합니다.

등록 정보 및 메소드

StarSuite Basic 의 개체는 다양한 등록 정보 및 메소드를 제공하며, 이들은 해당 개체를 통해 호출할 수 있습니다.

등록 정보

등록 정보는 `Document` 개체의 `Filename` 및 `Title` 과 같은 개체의 등록 정보와 유사합니다.

등록 정보는 간단한 대입을 통해 설정합니다.

```
Document.Title = "Programmierhandbuch StarSuite 7"
Document.Filename = "progman.sxv"
```

일반적인 변수와 마찬가지로 등록 정보는 해당 등록 정보가 기록할 수 있는 값을 지정하는 유형을 갖습니다.
위의 `Filename` 및 `Title` 등록 정보는 문자열 유형입니다.

실제 등록 정보 및 모방 등록 정보

StarSuite Basic 에서는 대부분의 개체 등록 정보를 UNO 서비스 설명처럼 지정합니다. 이러한 "실제" 등록 정보 외에도 StarSuite Basic 에는 UNO 수준의 2 가지 메소드로 구성되는 등록 정보를 제공합니다. 그 중 하나인 `get` 메소드는 등록 정보의 값을 쿼리하는 데 사용하고 `set` 메소드는 해당 값을 설정할 때 사용합니다. 이등록 정보는 사실상 두 메소드를 모방한 것입니다. 예를 들어 UNO 의 문자 개체는 해당 키 포인트를 호출하고 바꿀 수 있는 `getPosition` 및 `setPosition` 메소드를 제공합니다. StarSuite Basic 프로그래머는 `Position` 등록 정보를 통해 해당 값에 액세스할 수 있습니다. 그러나 원래의 메소드도 사용할 수 있습니다. 이 예에서는 `getPosition` 및 `setPosition` 이 됩니다.

메소드

메소드는 개체에 직접 적용되고 해당 개체를 호출하는 함수로 이해할 수 있습니다. 예를 들어 위 `Document` 개체는 다음과 같이 호출할 수 있는 `Save` 메소드를 제공합니다.

```
Document.Save()
```

메소드는 함수처럼 매개 변수를 갖고 값을 구할 수 있습니다. 그러한 메소드 호출 구문은 일반적인 함수와 비슷합니다. 다음과 같이 호출하면

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

Save 메소드 요청 시 문서 개체에 대해 `True` 매개 변수도 지정합니다.
메소드를 완료하면 `Save` 는 `Ok` 변수에 결과값을 저장합니다.

모듈, 서비스 및 인터페이스

StarSuite 는 수백 여 개의 서비스를 제공합니다. 이러한 서비스를 간략하게 파악할 수 있도록 모듈별로 분류했습니다. StarSuite Basic 프로그래머에게 모듈은 기능 상 별다른 의미를 갖지 않습니다. 서비스 이름 지정 시 의미가 있는 것은 모듈 이름뿐입니다. 모듈 이름을 함께 표시해야 하기 때문입니다. 서비스의 전체 이름은 StarSuite 서비스임을 나타내는 `com.sun.star` 식을 포함하며, 그 다음에 `frame` 과 같은 모듈 이름이 표시되고 마지막으로 `Desktop` 과 같은 실제 서비스 이름이 사용됩니다. 위의 예에서 전체 이름은 다음과 같습니다.

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

모듈과 서비스 용어 외에도 UNO 는 "인터페이스"라는 용어를 사용합니다. Java 프로그래머에게는 익숙한 용어이지만 Basic 에서는 사용하지 않습니다.

인터페이스는 여러 메소드가 결합한 것입니다. 엄격한 의미에서 UNO 의 서비스는 메소드보다는 인터페이스를 지원하며, 인터페이스가 다양한 메소드를 제공합니다. 즉 인터페이스에서 메소드가 서비스에 결합된 형태로 할당됩니다. 특히 Java 나 C++ 프로그래머에게는 이 세부 정보가 중요합니다. 이 언어에서는 메소드를 요청하는 데 인터페이스가 필요하기 때문입니다. 이 사항은 StarSuite Basic 과는 관련 없습니다. StarSuite Basic 에서는 해당 개체를 통해 직접 메소드를 호출합니다.

그러나 API 를 이해하기 위해서는 다양한 인터페이스에 대한 메소드 할당 방식을 익혀 두는 것이 유용합니다. 각기 다른 서비스에서 여러 인터페이스가 사용되기 때문입니다. 인터페이스에 익숙해지면 여러 서비스에 걸쳐 해당 지식을 활용할 수 있습니다.

몇 가지 중요한 인터페이스는 매우 자주 사용되므로, 이 장의 마지막 부분에서 각기 다른 서비스가 트리거하는 형태로 등장할 것입니다.

UNO 사용 도구

그렇다면 UNO 용어를 계속 사용하고자 할 때 어떤 개체 즉 서비스가 어떤 등록 정보, 메소드, 인터페이스를 지원하며 이를 어떻게 식별할 수 있는가 하는 점이 궁금할 것입니다. 본 설명서 외에도 `supportsService` 메소드, debug 메소드, Developer's Guide, API 참조 설명서 등을 참조하여 개체에 대한 자세한 정보를 얻을 수 있습니다.

supportsService 메소드

수 많은 UNO 개체가 `supportsService` 메소드를 지원하며, 이 메소드를 사용하여 어떤 개체의 특정 서비스 지원 여부를 설정할 수 있습니다. 예를 들어 다음과 같이 호출하면

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

`TextElement` 개체의 `com.sun.star.text.Paragraph` 서비스 지원 여부를 결정할 수 있습니다.

디버그 등록 정보

StarSuite Basic 의 모든 UNO 개체는 자신이 이미 갖추고 있는 등록 정보, 메소드 및 인터페이스에 대해 알고 있습니다. 해당 정보를 목록 형태로 구하는 등록 정보를 제공합니다. 해당 등록 정보는 다음과 같습니다.

`DBG_properties` - 개체의 모든 등록 정보를 포함하는 문자열을 구합니다.

`DBG_methods` - 개체의 모든 메소드를 포함하는 문자열을 구합니다.

`DBG_supportetInterfaces` - 개체를 지원하는 모든 인터페이스를 포함하는 문자열을 구합니다.

다음 프로그램 코드는 실제 응용 프로그램에서 `DBG_properties` 및 `DBG_methods` 를 어떻게 사용할 수 있는지 보여 줍니다. 이 코드에서는 먼저 `com.sun.star.frame.Desktop` 서비스를 만들고 지원 등록 정보 및 메소드를 메시지 상자에 표시합니다.

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

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

`DBG_properties` 사용 시 이 함수는 특정 서비스가 이론 상 지원할 수 있는 모든 등록 정보를 구합니다. 그러나 해당 개체 역시 이를 사용할 수 있는지 여부는 보장하지 않습니다. 따라서 등록 정보를 호출하기 전 `IsEmpty` 함수를 사용하여 실제 사용 가능 여부를 확인해야 합니다.

API 참조 설명서

사용 가능한 서비스, 인터페이스, 등록 정보에 대한 자세한 내용은 StarSuite API 의 API 참조 설명서에서 확인할 수 있습니다. www.openoffice.org 에서 제공합니다.

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

몇 가지 주요 인터페이스 개요

StarSuite 의 일부 인터페이스는 StarSuite API 에서 자주 사용합니다. 이들은 다양한 문제에 적용할 수 있는 추상적인 작업용 메소드 집합을 지정합니다. 그 중 가장 많이 사용하는 인터페이스를 간략하게 살펴 봅니다.

개체의 원본에 대해서는 본 설명서의 이후 부분에서 설명합니다. 여기에서는 StarSuite API 가 중요한 인터페이스를 제공하는 개체의 추상적인 측면 몇 가지를 살펴 보도록 합니다.

컨텍스트 종속 개체 만들기

StarSuite API 에서는 두 가지 옵션으로 개체를 만들 수 있습니다. 두 개의 옵션 중 하나는 이 장의 시작 부분에서 다룬 `createUnoService` 함수입니다. `createUnoService` 는 범용 개체를 만듭니다. 그러한 개체 및 서비스를 *컨텍스트 독립 서비스*라고도 부릅니다.

컨텍스트 독립 서비스 외에도 오직 다른 개체와 함께 사용할 수 있는 *컨텍스트 종속 서비스*가 있습니다. 예를 들면 스프레드시트 문서의 그림 개체는 오로지 이 문서와 함께 존재할 수 있습니다.

com.sun.star.lang.`XMultiServiceFactory` 인터페이스

컨텍스트 종속 개체는 일반적으로 해당 개체가 종속된 개체 메소드에 의해 만들어집니다. `XMultiServiceFactory` 인터페이스에서 지정하는 `createInstance` 메소드는 특히 문서 개체에서 사용합니다.

예를 들어 앞에서 언급한 그림 개체는 다음과 같이 스프레드시트 개체를 사용하여 만들 수 있습니다.

```
Dim RectangleShape As Object

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

텍스트 문서의 단락 서식도 같은 방식으로 만듭니다.

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

하위 개체에 대한 명명된 액세스

XNameAccess 및 XNameContainer 인터페이스는 자연어 이름을 사용하여 참조할 수 있는 하위 개체를 포함하는 개체에서 사용합니다.

XNamedAccess 는 개별 개체에 대한 액세스를 지원하는 데 비해 XNameContainer 는 요소의 삽입, 수정 및 삭제를 가능하게 합니다.

com.sun.star.container.XNameAccess 인터페이스

스프레드시트의 시트 개체를 통해 XNameAccess 를 사용하는 예를 소개합니다. 이 인터페이스는 스프레드시트의 모든 페이지를 결합합니다. 각 페이지는 XNameAccess 의 getByName 메소드를 사용하여 액세스합니다.

```
Dim Sheets As Object
Dim Sheet As Object

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

getElementNames 메소드는 모든 요소 이름에 대한 개요를 제공합니다. 즉 이름을 포함하는 데이터 필드를 결과값으로 제공합니다. 다음 예는 루프를 통해 스프레드시트의 모든 요소 이름을 식별하고 표시하는 방법을 소개합니다.

```
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 인터페이스의 hasByName 메소드는 기본 개체 내부에 특정 이름의 하위 개체가 존재하는지 여부를 나타냅니다. 따라서 다음 예는 Spreadsheet 개체가 Sheet1 이라는 이름의 페이지를 포함하는지 여부를 사용자에게 알리는 메시지를 표시합니다.

```
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 인터페이스

XNameContainer 인터페이스는 기본 개체에서 하위 개체의 삽입, 삭제 및 수정을 담당합니다. 해당 함수는 insertByName, removeByName, replaceByName 입니다.

다음은 그 실제 예를 보여 줍니다. StyleFamilies 개체를 포함하는 텍스트 문서를 호출하고 이 개체를 사용하여 해당 문서의 단락 서식(ParagraphStyles)을 설정합니다.

```
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 줄은 NewStyle 스타일을 같은 이름으로 ParagraphStyles 개체에 삽입합니다. replaceByName 줄은 ChangingStyle 의 해당 개체를 NewStyle 로 바꿉니다. 마지막으로 removeByName 을 호출하면 OldStyle 개체를 ParagraphStyles 에서 제거합니다.

하위 개체에 대한 색인 기반 액세스

XIndexAccess 및 XIndexContainer 인터페이스는 하위 개체를 포함하고 색인을 사용하여 참조할 수 있는 개체에서 사용합니다.

XIndexAccess 는 개별 개체에 액세스하는 메소드를 제공합니다.
XIndexContainer 는 요소를 삽입하고 제거하는 메소드를 제공합니다.

com.sun.star.container.XIndexAccess 인터페이스

XIndexAccess 는 하위 개체를 호출하는 getByIndex 및 getCount 메소드를 제공합니다. getByIndex 는 특정 색인을 갖는 개체를 제공합니다. getCount 는 사용 가능한 개체의 수를 나타냅니다.

```
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)
      ' 시트 편집
Next I
```

위의 예에서는 루프가 모든 시트 요소에 대해 차례로 실행되면서 각 요소에 대한 참조를 Sheet 개체 변수에 저장하는 절차를 보여 줍니다. 색인 사용 시 getCount 는 요소의 수를 구합니다. 그러나

getByIndex 의 요소 번호는 0 부터 시작합니다. 따라서 루프 횟수 변수 범위는 0~getCount()-1 이 됩니다.

com.sun.star.container.XIndexContainer 인터페이스

XIndexContainer 인터페이스는 insertByIndex 및 removeByIndex 함수를 제공합니다. 매개변수 구조는 XNameContainer 의 해당 함수와 같습니다.

하위 개체에 대한 반복 액세스

어떤 개체는 이름 또는 색인으로 참조할 수 없는 하위 개체의 목록을 포함하는 경우가 있습니다. 이 경우에는 XEnumeration 및 XenumerationAccess 인터페이스를 사용하는 것이 적합합니다. 이들은 개체의 모든 하위 개체를 직접 참조하지 않고 단계별로 처리할 수 있는 메커니즘을 제공합니다.

com.sun.star.container.XEnumeration 및 XenumerationAccess 인터페이스

기본 개체는 createEnumeration 메소드만 포함하는 XEnumerationAccess 인터페이스를 제공해야 합니다. 이는 보조 개체를 구하며, 이 보조 개체는 XEnumeration 인터페이스에 hasMoreElements 및 nextElement 메소드를 제공합니다. 이 메소드를 통해 하위 개체에 액세스할 수 있습니다.

다음 예는 텍스트의 모든 단락을 단계별로 처리합니다.

```
Dim ParagraphEnumeration As Object
Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

위의 예는 먼저 ParagraphEnumeration 이라는 보조 개체를 만듭니다. 이 개체는 텍스트의 각 단락을 루프를 통해 점진적으로 제공합니다. hasMoreElements 메소드가 False 값을 구하면서 텍스트의 끝에 도달했음을 알리면 루프는 즉시 종료됩니다.

5 장

StarSuite 문서 작업

StarSuite API 는 서로 다른 작업에서 가능한 많은 요소를 공통적으로 사용할 수 있도록 구성되었습니다. 여기에는 문서 만들기, 열기, 저장, 변환, 인쇄 및 서식 파일 관리를 위한 인터페이스와 서비스가 포함됩니다. 이러한 함수 영역은 모든 유형의 문서에 적용할 수 있으므로 이 장의 첫 부분에서 설명합니다.

StarDesktop

문서 작업 시 다음 두 가지 서비스를 가장 많이 사용합니다.

- `com.sun.star.frame.Desktop` 서비스 - StarSuite 의 핵심 서비스와 유사합니다. 이 서비스는 모든 문서 창이 속하는 StarSuite 프레임 개체의 함수를 제공합니다. 이 서비스를 이용하여 문서를 만들고 열거나 가져올 수 있습니다.
- `com.sun.star.document.OfficeDocument` 서비스 - 개별 문서 개체에 대한 기본 기능을 제공합니다. 이 서비스는 문서 저장, 내보내기, 인쇄용 메소드를 제공합니다.

StarSuite 를 시작하면 `com.sun.star.frame.Desktop` 서비스가 자동으로 실행됩니다. 이를 위해 StarSuite 는 전역 이름 `StarDesktop` 으로 액세스할 수 있는 개체를 만듭니다.

`StarDesktop` 에서 가장 중요한 인터페이스는 `com.sun.star.frame.XComponentLoader` 입니다. 이 서비스는 기본적으로 문서 만들기, 가져오기, 열기를 담당하는 `loadComponentFromURL` 메소드를 제공합니다.

`StarDesktop` 개체의 이름은 StarDesktop 이라는 공통 응용 프로그램에 모든 문서 창이 포함된 StarSuite 5 때부터 사용되었습니다. StarSuite 현재 버전에서는 화면에 표시되는 형태의 StarDesktop 을 더 이상 사용하지 않습니다. 대신 StarDesktop 이라는 이름으로 전체 응용 프로그램을 위한 기본 개체라는 것을 알 수 있으므로 StarSuite 의 프레임 개체용으로 그 이름을 계속 사용하게 되었습니다.

`StarDesktop` 개체는 예전에 루트 개체로 사용하던 StarSuite 5 `Application` 개체의 역할을 맡습니다. 그러나 `Application` 개체와 다르게 주로 새 문서를 여는 작업을 담당합니다. StarSuite 화면 표시를 제어하기 위해 이전 `Application` 개체가 제공했던 `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible` 등의 함수는 더 이상 사용하지 않습니다.

Word 의 활성 문서는 `Application.ActiveDocument` 를 통해, Excel 의 활성 문서는 `Application.ActiveWorkbook` 을 통해 액세스하지만, StarSuite 에서는 `StarDesktop` 이 이 작업을 수행합니다. StarSuite 7 에서 활성화된 문서 개체는 `StarDesktop. CurrentComponent` 등록 정보를 통해 액세스합니다.

StarSuite 문서에 대한 기본 정보

문서 작업 시 StarSuite 문서 관리의 몇 가지 기본 문제를 처리하는 것이 유용합니다. 여기에는 파일 저장 형식과 StarSuite 문서의 파일 이름 구조가 포함됩니다.

URL 표기법의 파일 이름

StarSuite 는 플랫폼 독립적인 응용 프로그램으로 설계되었으므로 Internet Standard RFC 1738 에서 지정한 (플랫폼 독립적) URL 표기법을 파일 이름에 사용합니다. 이 표기법에서 표준 파일 이름은 다음 접두사로 시작합니다.

`file:///`

그 다음에 로컬 경로를 표시합니다. 만일 파일 이름에 하위 디렉토리가 포함되면 일반적으로 Windows 에서 사용하는 백슬래시 대신 하나의 슬래시로 구분합니다. 다음 경로는 `c` 드라이브의 `doc` 디렉토리에 있는 `test.sxw` 파일을 참조합니다.

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

로컬 파일 이름을 URL 로 변환하기 위해 StarSuite 는 `ConvertToUrl` 함수를 제공합니다. URL 을 로컬 파일 이름으로 변환하기 위해 StarSuite 는 `ConvertFromUrl` 함수를 제공합니다.

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

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

위의 예는 로컬 파일 이름을 URL 로 변환하고 메시지 상자에 이 내용을 표시합니다. 그 다음 URL 을 로컬 파일 이름으로 변환하고 이 내용도 표시합니다.

그 기반이 되는 Internet Standard RFC 1738 은 `0-9`, `a-z`, `A-Z` 문자의 사용이 가능합니다. 다른 모든 문자는 이스케이프 코딩으로 URL 에 삽입됩니다. 이를 위해 문자를 ISO 8859-1(ISO-Latin) 문자 집합에서 해당되는 16 진수 값으로 변환하며 값 앞에는 백분율 기호가 붙습니다. 따라서 로컬 파일 이름의 공백은 URL 표기법에서 `%20` 이 됩니다.

XML 파일 형식

StarSuite 는 버전 6.0 부터 XML 기반 파일 형식을 제공합니다. XML 을 사용하여 다른 프로그램에서도 파일을 열고 편집할 수 있습니다.

파일 압축

XML 은 표준 텍스트 파일을 기반으로 하므로 결과 파일의 용량이 매우 큽니다. 따라서 StarSuite 는 파일을 압축하고 ZIP 파일로 저장합니다.

`storeAsURL` 메소드 옵션을 사용하여 원본 XML 파일을 직접 저장할 수 있습니다. 83 페이지의 storeAsURL 메소드 옵션을 참조하십시오.

문서 만들기, 열기 및 가져오기

문서를 열고 가져오거나 만드는 데 메소드를 사용합니다.

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

loadComponentFromURL 의 첫 번째 매개 변수는 해당 파일의 URL 을 지정합니다.

loadComponentFromURL 은 StarSuite 가 관리를 위해 내부적으로 만든 창의 프레임 개체 이름을 두 번째 매개 변수로 사용합니다. 미리 지정된 _blank 이름을 지정하는 것이 일반적이며, 이 경우 StarSuite 는 새 창을 만듭니다. 또는 _hidden 을 지정할 수 있으며, 이 경우에는 해당 문서가 로드되지만 보이지는 않습니다.

이 매개 변수를 사용하면 마지막 두 개의 매개 변수에 자리 표시자(dummy 값)만 할당할 수 있으므로 사용자가 StarSuite 문서를 열 수 있습니다.

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

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

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

위와 같이 호출하면 text.sxw 파일을 열고 새 창에 표시합니다.

StarSuite Basic 에서는 이런 방식으로 개수 제한 없이 문서를 열고 결과로 제공된 문서 개체를 사용하여 편집할 수 있습니다.

StarDesktop.loadComponentFromURL 은 기존 StarSuite API 의 Documents.Add 와 Documents.Open 메소드를 대체합니다.

문서 창의 내용 바꾸기

Frame 매개 변수에 대한 _blank 및 _hidden 값은 loadComponentFromURL 을 호출할 때마다 StarSuite 가 새로운 창을 만들도록 합니다. 기존 창의 내용을 바꾸는 것이 유용한 경우가 있습니다. 이 경우 창의 프레임 개체가 명확한 이름을 가져야 합니다. 이름은 밑줄로 시작해서는 안 됩니다. 또한 해당 프레임워크가 존재하지 않을 때 이를 만들려면 SearchFlags 매개 변수를 반드시 설정해야 합니다. SearchFlags 의 상수는 다음과 같습니다.

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

다음의 예는 프레임 매개 변수와 SearchFlags 를 사용하여 열린 창의 내용을 바꾸는 방법을 보여 줍니다.

```
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)
```

이 예는 먼저 test.sxw 파일을 MyFrame 이라는 프레임 이름을 갖는 새 창에서 엽니다. 메시지 상자를 확인하면 창의 내용을 test2.sxw 파일로 바꿉니다.

loadComponentFromURL 메소드 옵션

loadComponentFromURL 함수의 네 번째 매개 변수인 PropertyValue 데이터 필드는 StarSuite 의 다양한 문서 열기 및 만들기 옵션을 제공합니다. 데이터 필드는 각 옵션마다 옵션 이름이 해당 값 및 문자열로 저장된 PropertyValue 구조를 제공해야 합니다.

loadComponentFromURL 은 다음 옵션을 지원합니다.

- **AsTemplate (Boolean)** – True 일 경우 주어진 URL 에서 이름이 지정되지 않은 새 문서를 로드합니다. False 일 경우 편집 가능한 서식 파일을 로드합니다.
- **CharacterSet (String)** – 문서의 기초가 되는 문자 집합을 지정합니다.
- **FilterName (String)** – loadComponentFromURL 함수용 특수 필터를 지정합니다. 사용 가능한 필터 이름은 \share\config\registry\instance\org\openoffice\office\TypeDetection.xml 파일에서 지정합니다.
- **FilterOptions (String)** – 필터에 대한 추가 옵션을 지정합니다.
- **JumpMark (String)** – 문서를 열면 JumpMark 에서 지정한 위치로 이동합니다.
- **Password (String)** – 보호된 파일의 암호를 전송합니다.
- **ReadOnly (Boolean)** – 읽기 전용 문서를 로드합니다.

다음의 예는 StarSuite Calc 에서 쉼표를 사용하여 구분한 텍스트 파일을 FilterName 옵션으로 여는 방법을 설명합니다.

```
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` 데이터 필드는 하나의 옵션만 기록하므로 정확히 하나의 값만 처리합니다. `Filtername` 은 StarSuite 가 StarSuite Calc 텍스트 필터를 사용하여 파일을 열 것인지 여부를 정확하게 지정합니다.

새 문서 만들기

URL 에서 지정한 문서가 서식 파일이면 StarSuite 는 자동으로 새 문서를 만듭니다.

또는 변경 사항 없는 빈 문서가 필요한 경우 다음과 같이 `private:factory-URL` 을 지정할 수 있습니다.

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

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

위와 같이 호출하면 빈 StarSuite Writer 문서를 만들 수 있습니다.

문서 개체

이전 섹션에서 소개한 `loadComponentFromURL` 함수는 문서 개체를 구합니다. 이 개체는 다음 두 가지 주요 인터페이스를 제공하는 `com.sun.star.document.OfficeDocument` 서비스를 지원합니다.

- `com.sun.star.frame.XStorable` 인터페이스 - 문서 저장을 담당합니다.
- `com.sun.star.view.XPrintable` 인터페이스 - 문서 인쇄 메소드를 포함합니다.

StarSuite 7 에서는 문서 개체의 기능 범위가 대부분 동일하게 유지됩니다. 예를 들어 문서 개체는 문서 저장 및 인쇄용 메소드를 계속 제공합니다. 그러나 메소드의 이름 및 매개 변수는 바뀌었습니다.

문서 저장 및 내보내기

StarSuite 문서는 문서 개체를 통해 직접 저장됩니다. 이를 위해 `store` 메소드 (`com.sun.star.frame.XStorable` 인터페이스)가 제공됩니다.

```
Doc.store()
```

문서에 메모리 공간이 이미 할당된 경우 이 호출이 수행됩니다. 새 문서에는 이 사항이 적용되지 않습니다. 새 문서에는 `storeAsURL` 메소드가 사용됩니다. 또한

com.sun.star.frame.XStorable 에서 이 메소드를 지정하며, 문서의 위치를 정하는 데 이 메소드를 사용할 수 있습니다.

```
Dim URL As String
Dim Dummy()

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

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

com.sun.star.frame.XStorable 은 위 메소드 외에도 문서 저장 시 도움이 되는 몇 가지 메소드를 추가로 제공합니다. 그 메소드는 다음과 같습니다.

- **hasLocation()** – 문서에 이미 URL 이 할당되었는지 여부를 나타냅니다.
- **isReadonly()** - 읽기 전용 보호 문서인지 여부를 나타냅니다.
- **isModified()** - 문서를 마지막으로 저장한 후 변경 여부를 나타냅니다.

이 옵션을 통해 개체가 실제로 수정된 경우에만 문서를 저장하고 실제로 필요한 파일 이름만 쿼리하도록 문서 저장 코드를 확장할 수 있습니다.

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

이 예는 먼저 해당 문서가 마지막으로 저장된 후 수정된 적이 있는지 확인합니다. 수정된 적이 있는 경우에만 저장 프로세스를 진행합니다. 문서에 이미 URL 이 할당되었고 읽기 전용 문서가 아닐 경우 기존 URL 로 저장합니다. 문서가 URL 을 갖지 않거나 읽기 전용으로 열었다면 새 URL 로 저장합니다.

storeAsURL 메소드 옵션

loadComponentFromURL 메소드처럼 storeAsURL 메소드를 사용하여 PropertyValue 데이터 필드 형태로 일부 옵션을 지정할 수 있습니다. 이 옵션은 StarSuite 가 문서를 저장하는 순서를 지정합니다. storeAsURL 은 다음과 같은 옵션을 제공합니다.

- **CharacterSet (String)** – 문서의 기초가 되는 문자 집합을 지정합니다.
- **FilterName (String)** – loadComponentFromURL 함수에 대한 특수 필터를 지정합니다. 사용 가능한 필터 이름은 \share\config\registry\instance\org\openoffice\office\TypeDetection.xml 파일에서 지정합니다.
- **FilterOptions (String)** – 필터에 대한 추가 옵션을 지정합니다.
- **Overwrite (Boolean)** – 쿼리 없이 기존의 파일을 덮어씁니다.
- **Password (String)** – 보호된 파일의 암호를 전송합니다.
- **Unpacked (Boolean)** – (압축하지 않은) 문서를 하위 디렉토리에 저장합니다.

다음의 예는 Overwrite 옵션을 storeAsURL 과 함께 사용하는 방법을 보여 줍니다.

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

' ... Initialize Doc

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

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

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

이 예는 같은 이름의 파일이 이미 존재할 때 지정한 파일 이름으로 Doc 를 저장합니다.

문서 인쇄

저장과 마찬가지로 문서 개체를 통해 문서를 직접 인쇄할 수 있습니다. 이를 위해 com.sun.star.view.Xprintable 인터페이스의 Print 메소드가 제공됩니다. 가장 간단한 인쇄 호출은 다음과 같습니다.

```
Dim Dummy()

Doc.print(Dummy())
```

he loadComponentFromURL 메소드와 마찬가지로 Dummy 매개 변수는 StarSuite 가 다양한 인쇄 옵션을 지정할 수 있는 PropertyValue 데이터 필드입니다.

인쇄 메소드의 옵션

인쇄 메소드는 PropertyValue 데이터 필드를 매개 변수로 사용하며, 이 필드는 StarSuite 의 인쇄 대화 상자 설정을 반영합니다.

- **CopyCount (Integer)** – 인쇄할 복사본의 수를 지정합니다.
- **FileName (String)** – 지정한 파일의 문서를 인쇄합니다.
- **Collate (Boolean)** – 복사본을 한 부씩 인쇄하도록 프린터에 지시합니다.
- **Sort (Boolean)** – 여러 복사본을 인쇄할 때 페이지를 분류합니다. (CopyCount > 1).
- **Pages (String)** – 인쇄할 페이지의 목록을 포함합니다(인쇄 대화 상자에서 지정한 구문).

다음의 예는 Pages 옵션을 사용하여 문서의 여러 페이지를 인쇄하는 방법을 설명합니다.

```
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())
```

프린터 선택 및 설정

com.sun.star.view.XPrintable 인터페이스는 프린터를 선택할 수 있는 Printer 등록 정보를 제공합니다. 이 등록 정보는 다음과 같은 설정의 PropertyValue 데이터 필드를 받습니다.

- **Name (String)** –프린터의 이름을 지정합니다.
- **PaperOrientation (Enum)** –용지 방향을 지정합니다 (com.sun.star.view.PaperOrientation.PORTRAIT 값은 세로 방향, com.sun.star.view.PaperOrientation.LANDSCAPE 는 가로 방향).
- **PaperFormat (Enum)** –용지 형식을 지정합니다(예: com.sun.star.view.PaperFormat.A4 는 DIN A4, com.sun.star.view.PaperFormat.Letter 는 US letter).
- **PaperSize (Size)** –1/100 밀리미터 단위로 용지 크기를 지정합니다.

다음의 예는 Printer 등록 정보를 사용하여 프린터를 바꾸고 용지 크기를 설정하는 방법을 보여 줍니다.

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

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

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

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

Doc.Printer = PrinterProperties()
```

이 예는 com.sun.star.awt.Size 유형의 개체를 PaperSize 라는 이름으로 지정합니다. 용지 크기를 지정하기 위해서는 이 개체가 필요합니다. 또한 이 개체는 2 개의 PropertyValue 항목에 대해 PrinterProperties 라는 이름의 데이터 필드를 만듭니다. 그런 다음 설정할 값으로 이 데이터 필드를 초기화하고 Printer 등록 정보를 할당합니다. UNO 관점에서 프린터는 실제 등록 정보가 아닌 모방 등록 정보입니다.

서식 파일

서식 파일은 서식 설정 속성을 포함하는 명명된 목록입니다. StarSuite 의 모든 응용 프로그램에 적용되며 서식 설정을 크게 간소화하는 데 기여합니다. 사용자가 서식 파일의 속성 중 하나를 바꾸면 StarSuite 는 그 속성을 사용하는 모든 문서 구역을 자동 조정합니다. 예를 들어 사용자는 문서의 중앙 수정 방식을 통해 수준 1 머리글의 글꼴 유형을 한꺼번에 바꿀 수 있습니다. StarSuite 는 해당 문서 유형에 따라 각기 다른 다양한 서식 파일 유형을 인식합니다.

StarSuite Writer 는 다음을 지원합니다.

- 문자 서식 파일,
- 단락 서식 파일,
- 프레임 서식 파일,
- 페이지 서식 파일
- 번호 매기기 서식 파일

StarSuite Calc 는 다음을 지원합니다.

- 셀 서식 파일
- 페이지 서식 파일

StarSuite Impress 는 다음을 지원합니다.

- 문자 요소 서식 파일
- 프레젠테이션 서식 파일

StarSuite 의 용어에서 각기 다른 서식 파일 유형은 그 기초가 되는 `com.sun.star.style.StyleFamily` 서비스를 따라 `StyleFamilies` 라 부릅니다.
문서 개체를 통해 `StyleFamilies` 에 액세스합니다.

```
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")
```

이 예는 스프레드시트 문서의 `StyleFamilies` 등록 정보를 사용하여 사용 가능한 모든 셀 서식 파일의 목록을 만듭니다.

각 서식 파일은 색인을 통해 직접 액세스할 수 있습니다.

```
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
```

이전 예에서 모든 셀 서식 파일의 이름을 차례로 메시지 상자에 표시하기 때문에 루프가 추가되었습니다.

다양한 서식 설정 옵션에 대한 세부 정보

각 서식 파일 유형은 개별 서식 설정 등록 정보를 광범위하게 제공합니다. 다음은 가장 중요한 서식 설정 속성 및 그에 대한 설명이 수록된 곳을 간략하게 소개합니다.

- 문자 등록 정보 6 장, 텍스트 문서, `com.sun.star.style.CharacterProperties` 서비스
- 단락 등록 정보 6 장, 텍스트 문서, `com.sun.star.text.Paragraph` 서비스
- 셀 등록 정보, 7장, 스프레드시트 문서, `com.sun.star.table.CellProperties` 서비스
- 페이지 등록 정보, 7장, 스프레드시트 문서, `com.sun.star.style.PageStyle` 서비스
- 문자 요소 등록 정보 7 장, 스프레드시트 문서, 다양한 서비스

서식 등록 정보는 여기에서 설명한 응용 프로그램에 한정되지 않고 보편적으로 사용합니다. 예를 들어 7 장에서 설명한 페이지 등록 정보 대부분은 StarSuite Calc 뿐 아니라 StarSuite Writer 에서도 사용할 수 있습니다.

서식을 사용하는 작업에 대한 자세한 내용은 6 장, 텍스트 문서의 문자 및 단락 등록 정보의 기본값 섹션에서 확인할 수 있습니다.

6 장

텍스트 문서

텍스트 문서는 순수 문자열 외에 서식 설정 정보를 포함합니다. 이는 텍스트 상의 어떤 위치에도 표시할 수 있습니다. 이 구조는 표를 통해 더 복잡성을 띠게 됩니다. 여기에는 1 차원 문자열뿐 아니라 2 차원 필드도 포함됩니다. 현재 대부분의 워드 프로세싱 프로그램은 그림 개체, 텍스트 프레임 및 기타 개체를 텍스트 내부에 포함시키는 옵션을 제공하고 있습니다. 이러한 개체는 텍스트 흐름 외부를 비롯하여 페이지의 어느 위치에도 놓일 수 있습니다.

이 장에서는 텍스트 문서의 주요 인터페이스와 서비스를 소개합니다. 첫 번째 섹션에서는 텍스트 문서의 구조를 소개하고 StarSuite Basic 프로그램을 사용하여 StarSuite 문서의 반복적 작업을 어떻게 해결할 수 있는지 집중적으로 살펴 봅니다. 단락, 단락의 부분 및 이에 대한 서식 설정에 초점을 두고 있습니다.

두 번째 섹션은 효율적인 텍스트 문서 작업에 중점을 둡니다. 이를 위해 StarSuite 는 첫 번째 섹션에서 지정된 범위를 확장하는 `TextCursor` 개체와 같은 몇 가지 도움 개체를 제공합니다.

세 번째 섹션은 텍스트 작업 이외의 내용을 다룹니다. 표, 텍스트 프레임, 텍스트 필드, 책갈피, 컨텐트 디렉토리 등에 중점을 둡니다.

문서 만들기, 열기, 저장 및 인쇄 방법에 대한 정보는 텍스트 문서뿐 아니라 다른 유형의 문서에도 적용되는 사항이므로 5 에서 다루고 있습니다.

텍스트 문서의 구조

텍스트 문서는 기본적으로 4 가지 유형의 정보를 담을 수 있습니다.

- 실제 텍스트
- 문자, 단락 및 페이지용 서식 파일
- 표, 그림, 그림 개체 등 텍스트가 아닌 요소
- 텍스트 문서의 전역 설정

이 섹션에서는 텍스트 및 관련 서식 설정 옵션을 집중적으로 살펴 봅니다.

StarSuite Writer 용 StarSuite 7.x API 디자인은 이전 버전과 다릅니다. 기존 API 버전은 `Selection` 개체 사용에 중점을 두었기 때문에 최종 사용자를 위한 사용자 인터페이스 개념에 큰 비중을 두고 마우스로 제어하는 강조 표시 작업에 초점을 맞추었습니다.

StarSuite 7.x API 는 사용자 인터페이스와 프로그래머 인터페이스 간의 기존 연결을 대체했습니다. 따라서 프로그래머는 응용 프로그램의 모든 요소를 병렬 액세스하고 문서의 서로 다른 하위 구역의 개체를 동시에 사용하면서 작업할 수 있습니다. 기존의 `Selection` 개체는 더 이상 사용하지 않습니다.

단락 및 단락 부분

텍스트 문서의 핵심은 일련의 단락으로 구성됩니다. 단락은 이름이나 색인이 지정되지 않으므로 각 단락을 직접 액세스할 방법이 없습니다. 그러나 4 장에서 다룬 Enumeration 개체를 사용하여 단락에 연속적으로 액세스할 수 있습니다. 따라서 단락 편집이 가능합니다.

그러나 Enumeration 개체 사용 시 유의해야 할 사항이 한 가지 있습니다.
이 개체는 단락뿐 아니라 표까지 결과로 제공합니다. 엄밀히 말해서 StarSuite Writer 에서 표는 특수한 유형의 단락입니다. 따라서 결과로 제공된 개체를 액세스하기 전에 해당 개체가 단락을 위한 com.sun.star.text.Paragraph 서비스 또는 표를 위한 com.sun.star.text.TextTable 서비스를 지원하는지 확인해야 합니다.

다음의 예는 텍스트 문서의 내용을 루프 방식으로 액세스하고 각 인스턴스마다 메시지를 사용하여 해당 개체가 단락인지 표인지 여부를 사용자에게 알립니다.

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

' 문서 개체 생성
Doc = StarDesktop.CurrentComponent

' enumeration 개체 생성
Enum = Doc.Text.createEnumeration

' 모든 텍스트 요소에 대해 루프 실행
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
```

위의 예에서는 현재 StarSuite 문서를 참조하는 Doc 문서 개체를 만듭니다. 그런 다음 Doc 을 이용하여 텍스트의 각 부분, 즉 단락 및 표에 액세스할 수 있는 Enumeration 개체를 만들고 현재의 요소를 TextElement 개체에 할당합니다. 이 예에서는 supportsService 메소드를 사용하여 TextElement 가 단락인지 표인지 확인하고 해당 결과를 메시지로 표시합니다.

단락

com.sun.star.text.Paragraph 서비스는 단락 내용에 대한 액세스를 가능하게 합니다. 단락의 텍스트는 String 등록 정보를 사용하여 가져오거나 수정할 수 있습니다.

```
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
```

위의 예에서는 현재 텍스트 문서를 열고 `Enumeration` 개체를 사용하여 문서를 검토합니다. 모든 단락에서 `TextElement.String` 등록 정보를 사용하여 관련 단락을 액세스하고 `you`, `too`, `for` 문자열을 `U`, `2`, `4` 문자로 바꿉니다.
바꾸기에 사용한 `Replace` 함수는 StarSuite Basic 의 표준 언어 범위에 해당되지 않습니다. 이는 33 찾기 및 바꾸기 부분의 예에서 설명한 바 있는 함수입니다.

여기에서 설명하는 텍스트 단락 액세스 절차의 내용은 VBA 의 `Paragraphs` 목록 작업과 유사합니다. VBA 에서는 `Range` 및 `Document` 개체를 통해 그러한 기능을 제공합니다. VBA 에서는 번호를 통해 단락을 액세스하지만(예: `Paragraph(1)` 호출) StarSuite Basic 에서는 위에서 설명한 `Enumeration` 개체를 사용해야 합니다.

StarSuite Basic 에서는 VBA 가 제공하는 `Characters`, `Sentences`, `Words` 목록과 같은 것이 없습니다. 그러나 문자, 문장, 단어 수준의 탐색을 지원하는 `TextCursor` 로 전환하는 옵션이 있습니다(*TextCursor* 참조).

단락 부분

이전 예에서는 요청한 대로 텍스트를 바꿀 수 있었지만 서식 설정을 없애는 경우도 있습니다.

그 이유는 단락이 개별 하위 개체로 구성되기 때문입니다. 각 하위 개체는 고유한 서식 설정 정보를 갖습니다. 예를 들어 단락의 가운데에 굵게 표시된 단어가 있다면 StarSuite 에서는 3 개의 단락 부분, 즉 굵게 표시된 단어 이전 부분, 굵게 표시된 단어, 굵게 표시된 단어 이후 부분으로 나누어 지며, 마지막 부분은 표준으로 표시됩니다.

단락의 `String` 등록 정보를 사용하여 해당 단락의 텍스트를 바꾸면 StarSuite 는 먼저 기존 단락 부분을 제거하고 새로운 단락 부분을 삽입합니다. 그럼 다음 이전 구역의 서식 설정은 사라집니다.

이를 방지하기 위해 사용자는 전체 단락 대신 해당 단락 부분만 액세스할 수 있습니다. 이를 위해 단락은 자체적으로 `Enumeration` 개체를 제공합니다. 다음의 예는 이중 루프를 통해 텍스트 문서의 모든 단락 및 단락 부분을 검토하고 이전 예의 바꾸기 프로세스를 적용합니다.

```
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

' 모든 단락에 대해 루프 실행
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration

              ' 모든 하위 단락에 대해 루프 실행
              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
```

위의 예는 이중 루프를 통해 텍스트 문서를 검토합니다. 외부 루프는 텍스트의 단락을 참조합니다. 내부 루프는 각 단락의 단락 부분을 처리합니다. 이 코드의 예는 이전 단락의 예와 마찬가지로 `String` 등록 정보를 사용하여 각 단락 부분의 내용을 수정합니다. 그러나 단락 부분을 직접 편집하므로 문자열을 바꾸더라도 서식 설정 정보는 그대로 유지됩니다.

서식 설정

다양한 방법으로 텍스트 서식을 설정할 수 있습니다. 가장 간단한 방법은 서식 등록 정보를 텍스트 시퀀스에 직접 할당하는 것입니다. 이를 직접 서식 설정이라고 부릅니다. 직접 서식 설정은 사용자가 마우스를 사용하여 서식을 지정할 수 있으므로 특히 짧은 문서에서 사용합니다. 예를 들어 텍스트 내부의 특정 단어를 굵게 표시하여 강조 표시하거나 한 줄을 가운데 맞춤할 수 있습니다.

직접 서식 설정 외에도 서식 파일을 사용하여 텍스트 서식을 설정할 수 있습니다. 이를 간접 서식 설정이라고 부릅니다. 간접 서식 설정에서는 사용자가 미리 지정된 서식 파일을 해당 텍스트 부분에 적용합니다. 나중에 텍스트의 레이아웃을 바꾸려면 사용자는 서식 파일만 바꾸면 됩니다. 그러면 StarSuite 는 해당 서식 파일을 사용하는 모든 텍스트 부분에 해당 변경 사항을 적용합니다.

VBA 에서는 개체의 서식 설정 등록 정보가 일반적으로 광범위한 하위 개체에 걸쳐 분산되어 있습니다(예: `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). 이 등록 정보는 캐스케이딩 식을 통해 액세스합니다(예: `Range.Font.AllCaps`). 한편 StarSuite Basic 에서는 관련 개체를 사용하여 서식 설정 등록 정보를 직접 사용할 수 있습니다(예: `TextCursor`, `Paragraph` 등). StarSuite 에서 사용할 수 있는 문자 및 단락 등록 정보에 대해서는 다음 두 섹션에서 간략하게 소개합니다.

기존 StarSuite API 에서는 기본적으로 `Selection` 개체 및 해당 하위 개체(예: `Selection.Font`,

Selection.Paragraph, Selection.Border)를 사용하여 텍스트 서식을 설정했습니다. 새 API 에서는 각 개체마다 Paragraph, TextCursor 등과 같은 서식 설정 등록 정보가 있기 때문에 직접 적용할 수 있습니다. 사용 가능한 문자 및 단락 등록 정보 목록은 다음 단락에 있습니다.

문자 등록 정보

각 문자를 참조하는 서식 등록 정보를 문자 등록 정보라고 부릅니다. 여기에는 굵게 표시 유형 및 글꼴 유형이 포함됩니다. 문자 등록 정보 설정이 가능한 개체는 com.sun.star.style.CharacterProperties 서비스를 지원해야 합니다. StarSuite 는 이 서비스를 지원하는 전체 서비스 범위를 인식합니다. 여기에는 위에서 다룬 단락용 com.sun.star.text.Paragraph 서비스와 단락 부분용 com.sun.star.text.TextPortion 서비스가 포함됩니다.

com.sun.star.style.CharacterProperties 서비스는 인터페이스를 제공하지 않지만, 그 대신 문자 등록 정보를 지정하고 호출할 수 있는 다양한 등록 정보를 제공합니다. 모든 문자 등록 정보를 수록한 목록은 StarSuite API 참조 설명서에서 확인할 수 있습니다. 다음 목록은 가장 중요한 등록 정보를 소개합니다.

- **CharFontName (String)** – 선택한 글꼴 유형 이름
- **CharColor (Long)** – 텍스트 색상
- **CharHeight (Float)** – 포인트(pt) 단위 문자 높이
- **CharUnderline (Constant group)** – 밑줄 유형(com.sun.star.awt.FontUnderline 을 따르는 상수)
- **CharWeight (Constant group)** – 글꼴 굵기(com.sun.star.awt.FontWeight 를 따르는 상수)
- **CharBackColor (Long)** – 배경 색상
- **CharKeepTogether (Boolean)** – 자동 줄바꿈 안함
- **CharStyleName (String)** – 문자 서식 파일 이름

단락 등록 정보

개별 문자가 아니라 전체 단락을 참조하는 서식 설정 정보를 단락 등록 정보로 간주합니다. 여기에는 페이지 가장자리와 단락 사이의 간격, 줄 간격이 포함됩니다. 단락 등록 정보는 com.sun.star.style.ParagraphProperties 서비스를 통해 사용할 수 있습니다.

단락 등록 정보도 다양한 개체에서 제공됩니다. com.sun.star.text.Paragraph 서비스를 지원하는 모든 개체는 com.sun.star.style.ParagraphProperties 의 단락 등록 정보도 지원합니다.

단락 등록 정보의 전체 목록은 StarSuite API 참조 설명서에서 확인할 수 있습니다. 가장 많이 사용하는 단락 등록 정보는 다음과 같습니다.

- **ParaAdjust (enum)** – 세로 텍스트 방향(com.sun.star.style.ParagraphAdjust 를 따르는 상수)

- **ParaLineSpacing (struct)** – 줄 간격(com.sun.star.style.LineSpacing 을 따르는 상수)
- **ParaBackColor (Long)** – 배경 색상
- **ParaLeftMargin (Long)** – 1/100 밀리미터 단위의 왼쪽 여백
- **ParaRightMargin (Long)** – 1/100 밀리미터 단위의 오른쪽 여백
- **ParaTopMargin (Long)** – 1/100 밀리미터 단위의 위 여백
- **ParaBottomMargin (Long)** – 1/100 밀리미터 단위의 아래 여백
- **ParaTabStops (Array of struct)** – 탭 유형 및 위치(Typs com.sun.star.style.TabStop 구조의 행렬)
- **ParaStyleName (String)** – 단락 서식 파일의 이름

예: 간단한 HTML 내보내기

다음의 예는 서식 설정 정보의 사용 방법을 소개합니다. 텍스트 문서를 반복하여 검토하면서 간단한 HTML 파일을 만듭니다. 각 단락은 단락을 나타내는 HTML 요소 <P>를 사용하여 기록합니다. 굵게 표시된 단락 부분을 내보낼 때에는 <B>라는 HTML 요소를 사용하여 표시합니다.

```
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

' 모든 단락에 대해 루프 실행
While Enum1.hasMoreElements
	TextElement = Enum1.nextElement
	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = TextElement.createEnumeration
		CurLine = "<P>"

		' 모든 단락 부분에 대해 루프 실행
		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

		' 줄 출력
		CurLine = CurLine & "</P>"
		Print #FileNo, CurLine

	End If
Wend

' HTML 바닥글 작성
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

위 예의 기본 구조는 이미 다룬 바 있는 텍스트 단락 부분의 검토 예와 유사합니다. HTML 파일을 쓰는 함수, 해당 텍스트 부분의 글꼴 굵기를 확인하고 굵게 쓰인 단락 부분에 해당 HTML 태그를 제공하는 테스트 코드가 추가되었습니다.

문자 및 단락 등록 정보의 기본값

직접 서식 설정은 항상 간접 서식 설정보다 우선합니다. 즉 서식 파일을 사용한 서식 설정은 텍스트에서의 직접 서식 설정보다 우선 순위가 낮습니다.

문서 일부의 직접 또는 간접 서식 설정 여부는 쉽게 확인할 수 없습니다. StarSuite 가 제공하는 기호 모음은 글꼴 유형, 굵기, 크기 등 자주 사용하는 텍스트 등록 정보를 나타냅니다. 그러나 텍스트에서 해당 설정이 서식 파일을 기반으로 하는지 아니면 직접 서식 설정인지 여부는 불확실합니다.

StarSuite Basic 은 프로그래머가 특성 등록 정보의 설정 방식을 확인할 수 있도록 `getPropertyState` 메소드를 제공합니다. 이 메소드는 등록 정보의 이름을 매개 변수로 받고 해당 서식 설정의 근원에 대한 정보를 나타내는 상수를 구합니다. `com.sun.star.beans.PropertyState` enumeration 에서 지정한 대로 다음과 같은 결과가 나올 수 있습니다.

- **`com.sun.star.beans.PropertyState.DIRECT_VALUE`** – 텍스트에서 직접 지정한 등록 정보(직접 서식 설정)
- **`com.sun.star.beans.PropertyState.DEFAULT_VALUE`** – 서식 파일에서 지정한 등록 정보(간접 서식 설정)
- **`com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE`** – 불확실한 등록 정보
  예를 들어 굵게 표시된 단어와 보통 글꼴 단어가 모두 포함된 어느 단락에 대해 굵게 표시 유형 등록 정보를 쿼리했을 때 이와 같은 결과가 나옵니다.

다음의 예는 StarSuite 에서 서식 등록 정보를 편집하는 방법을 보여 줍니다. 텍스트를 두루 살피면서 직접 서식 설정을 통해 굵게 표시된 단락 부분을 찾습니다. 해당 단락 부분을 찾으면 `setPropertyToDefault` 메소드를 사용하여 직접 서식 설정을 해제하고 해당 단락 부분에 `MyBold` 문자 서식 파일을 적용합니다.

```
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

' 모든 단락에 대해 루프 실행
While Enum1.hasMoreElements
       TextElement = Enum1.nextElement
       If TextElement.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = TextElement.createEnumeration

              '모든 단락 부분에 대해 루프 실행
              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
```

텍스트 문서 편집

이전 섹션에서는 단락 부분 및 단락에 대한 액세스를 가능하게 하는 `com.sun.star.text.TextPortion` 및 `com.sun.star.text.Paragraph` 서비스에 중점을 두고 텍스트 문서 편집의 전체 옵션을 설명했습니다. 이 서비스는 루프를 통해 한 번에 텍스트 내용을 편집하는 응용 프로그램에 적합합니다. 그러나 이는 많은 문제를 해결하기에 충분하지 않습니다. StarSuite 는 보다 복잡한 작업을 위해 `com.sun.star.text.TextCursor` 서비스를 제공합니다. 즉 이 서비스를 이용하여 문서 내부에서 뒤로 탐색하거나 `TextPortions` 대신 문장 및 단어를 기준으로 하여 탐색할 수 있습니다.

TextCursor

StarSuite API 의 `TextCursor` 는 StarSuite 문서에서 사용하는 보이는 커서와 유사합니다. 텍스트 문서 내부의 특정 위치를 표시하며 명령을 사용하여 다양한 방향으로 탐색할 수 있습니다. 그러나 StarSuite Basic 에서 지원하는 `TextCursor` 개체를 보이는 커서와 혼동해서는 안 됩니다. 이 두 가지는 서로 전혀 다릅니다.

경고! 용어는 VBA 에서 사용하는 용어와 다릅니다. 함수 범위 측면에서 VBA 의 `Range` 개체는 StarSuite 의 `TextCursor` 개체와 비교할 수 있지만, 이름이 암시하는 바와 달리 StarSuite 의 `Range` 개체와는 비교할 수 없습니다.

예를 들어 StarSuite 의 `TextCursor` 개체는 VBA 의 `Range` 개체에 포함된 텍스트를 탐색하고 변경할 수 있는 메소드를 제공합니다 (예: `MoveStart`, `MoveEnd`, `InsertBefore`, `InsertAfter`). StarSuite 의 `TextCursor` 개체에 대응하는 요소에 대해서는 다음 섹션에서 설명합니다.

텍스트 내부 탐색

StarSuite Basic 의 `TextCursor` 개체는 텍스트 문서의 보이는 커서와 상관 없이 기능을 수행합니다. `TextCursor` 개체의 프로그램 제어 위치를 변경해도 보이는 커서에는 어떤 영향도 주지 않습니다. 같은 문서에 대해 여러 `TextCursor` 개체를 열고 다양한 위치에서 상호 독립적으로 사용할 수 있습니다.

`TextCursor` 개체는 `createTextCursor` 를 호출하여 만듭니다.

```
Dim Doc As Object
Dim Cursor As Object

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

이렇게 만든 `Cursor` 개체는 `com.sun.star.text.TextCursor` 서비스를 지원하고, 이 서비스는 텍스트 문서 내부를 탐색하는 모든 메소드를 제공합니다. 다음의 예는 먼저 `TextCursor` 를 10 문자 간격만큼 왼쪽으로 옮긴 다음 다시 오른쪽으로 3 문자 간격만큼 옮깁니다.

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

TextCursor 는 전체 영역을 강조 표시할 수 있습니다. 이는 마우스를 사용하여 텍스트 상의 특정 지점을 강조 표시하는 것과 비교할 수 있습니다. 위 함수 호출에서 False 매개 변수는 커서가 이동하며 지나는 영역을 강조 표시할 것인지 여부를 지정합니다. 예를 들어 다음 TextCursor 는

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

먼저 강조 표시하지 않고 10 문자 간격만큼 오른쪽으로 이동하고 다시 3 문자 간격만큼 왼쪽으로 돌아온 다음 강조 표시합니다. 따라서 TextCursor 가 강조 표시하는 영역은 텍스트에서 7 번째 문자부터 10 번째 문자까지입니다.

com.sun.star.text.TextCursor 서비스가 제공하는 주요 탐색용 메소드를 소개합니다.

- **goLeft (Count, Expand)** – Count 문자만큼 왼쪽으로 이동합니다.
- **goRight (Count, Expand)** – Count 문자만큼 오른쪽으로 이동합니다.
- **gotoStart (Expand)** – 텍스트 문서의 시작으로 이동합니다.
- **gotoEnd (Expand)** – 텍스트 문서의 끝으로 이동합니다.
- **gotoRange (TextRange, Expand)** – 지정된 TextRange 개체로 이동합니다.
- **gotoStartOfWord (Expand)** – 현재 단어의 시작으로 이동합니다.
- **gotoEndOfWord (Expand)** – 현재 단어의 끝으로 이동합니다.
- **gotoNextWord (Expand)** – 다음 단어의 시작으로 이동합니다.
- **gotoPreviousWord (Expand)** – 이전 단어의 시작으로 이동합니다.
- **isStartOfWord ()** - TextCursor 가 단어의 시작 위치에 있을 경우 True 를 구합니다.
- **isEndOfWord ()** - TextCursor 가 단어의 끝에 있을 경우 Ture 를 구합니다.
- **gotoStartOfSentence(Expand)** – 현재 문장의 시작 위치로 이동합니다.
- **gotoEndOfSentence(Expand)** – 현재 문장의 끝으로 이동합니다.
- **gotoNextSentence(Expand)** – 다음 문장의 시작 위치로 이동합니다.
- **gotoPreviousSentence(Expand)** – 이전 문장의 시작 위치로 이동합니다.
- **isStartOfSentence ()** - TextCursor 가 문장의 시작 위치에 있을 경우 True 를 구합니다.
- **isEndOfSentence ()** - TextCursor 가 문장의 끝에 있을 경우 True 를 구합니다.
- **gotoStartOfParagraph(Expand)** – 현재 단락의 시작으로 이동합니다.
- **gotoEndOfParagraph(Expand)** – 현재 단락의 끝으로 이동합니다.
- **gotoNextParagraph(Expand)** – 다음 단락의 시작으로 이동합니다.
- **gotoPreviousParagraph(Expand)** – 이전 단락의 시작으로 이동합니다.
- **isStartOfParagraph ()** – TextCursor 가 단락의 시작 위치에 있을 경우 True 를 구합니다.
- **isEndOfParagraph ()** – TextCursor 가 단락의 끝에 있을 경우 True 를 구합니다.

텍스트는 문장 기호를 기준으로 여러 문장으로 나뉩니다. 예를 들어 마침표는 문장의 끝을 알리는 기호로 해석합니다.

Expand 매개 변수는 탐색 중 지나는 영역을 강조 표시할 것인지 여부를 지정하는 Boolean 값입니다. 모든 탐색 메소드는 탐색이 성공했는지 또는 텍스트 부족으로 작업이 중단되었는지를 나타내는 매개 변수를 구합니다.

다음은 TextCursor 를 사용하여 강조 표시된 영역을 편집하며 com.sun.star.text.TextCursor 서비스를 지원하는 몇 가지 메소드입니다.

- **collapseToStart ()** – 강조 표시를 취소하고 TextCursor 를 이전 강조 표시 영역의 시작 위치에 둡니다.
- **collapseToEnd ()** – 강조 표시를 취소하고 TextCursor 를 이전 강조 표시 영역의 끝 위치에 둡니다.
- **isCollapsed ()** – 현재 TextCursor 가 어떤 영역도 강조 표시하지 않으면 True 를 구합니다.

TextCursor 로 텍스트 서식 설정

com.sun.star.text.TextCursor 서비스는 이 장의 시작 부분에서 소개한 모든 문자 및 단락 등록 정보를 지원합니다.

다음의 예는 TextCursor 와 함께 이 등록 정보를 사용하는 방법에 대해 소개합니다. 전체 문서를 검토하면서 각 문장의 첫 번째 단어를 굵게 표시합니다.

```
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
```

위의 예는 방금 열었던 텍스트용 문서 개체를 만듭니다. 그리고 나서 전체 텍스트를 문장별로 반복하면서 첫 번째 단어를 강조 표시하고 굵게 표시합니다.

텍스트 내용 검색 및 수정

`TextCursor` 가 강조 표시 영역을 포함하면 TextCursor 개체의 `String` 등록 정보를 이 텍스트에 적용할 수 있습니다. 다음의 예는 `String` 등록 정보를 사용하여 문장의 첫 단어를 메시지 상자에 표시합니다.

```
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
```

각 문장의 첫 번째 단어는 `String` 등록 정보를 사용하여 똑같이 수정할 수 있습니다.

```
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
```

`TextCursor` 가 강조 표시 영역을 포함하면 `String` 등록 정보 대입은 이 영역을 새로운 텍스트로 바꿉니다. 강조 표시 영역이 없으면 해당 텍스트를 현재 `TextCursor` 위치에 삽입합니다.

제어 코드 삽입

문서의 실제 텍스트 대신 텍스트의 구조를 수정해야 하는 경우가 있습니다. 이를 위해 StarSuite 는 제어 코드를 제공합니다. 제어 코드는 텍스트에 삽입되어 해당 구조에 영향을 줍니다. 제어 코드는 `com.sun.star.text.ControlCharacter` 상수 그룹에서 지정합니다. StarSuite 에서는 다음 제어 코드를 사용할 수 있습니다.

- **`PARAGRAPH_BREAK`** – 단락 나누기.
- **`LINE_BREAK`** – 단락 내 줄바꿈.
- **`SOFT_HYPHEN`** – 음절 구분 가능 포인트.
- **`HARD_HYPHEN`** – 필수 음절 구분 포인트.
- **`HARD_SPACE`** – 양쪽 맞춤 텍스트에서 넓히거나 좁히지 않는 보호 간격.

제어 코드를 삽입하려면 커서뿐 아니라 해당 텍스트 문서 개체가 필요합니다. 다음의 예는 텍스트의 20 번째 문자 다음에 단락을 삽입합니다.

```
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)
```

insertControlCharacter 메소드 호출 시 `False` 매개 변수는 삽입 작업 후에도 `TextCursor` 가 현재 강조 표시하는 영역이 남아 있게 합니다. `True` 매개 변수를 전달하면 `insertControlCharacter` 는 현재 텍스트를 바꿉니다.

텍스트 부분 검색

특정 용어를 찾아 텍스트를 검색하고 해당 포인트를 편집해야 하는 경우가 많습니다. 모든 StarSuite 문서는 이를 위해 특수 인터페이스를 제공하며, 이 인터페이스는 항상 같은 원칙에 따라 기능을 수행합니다. 검색 프로세스를 시작하기 전에 먼저 `SearchDescriptor` 를 만들어야 합니다. 이는 StarSuite 가 문서에서 검색할 대상을 지정합니다. `SearchDescriptor` 는 `com.sun.star.util.SearchDescriptor` 서비스를 지원하는 개체이며, 문서의 `createSearchDescriptor` 메소드를 사용하여 만들 수 있습니다.

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

일단 만들어진 `SearchDescriptor` 는 검색할 텍스트를 받습니다.

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

기능 측면에서 SearchDescriptor 는 StarSuite 의 검색 대화 상자와 가장 유사합니다. 검색 창과 유사한 방식으로 검색에 필요한 설정을 SearchDescriptor 개체에 지정할 수 있습니다.

com.sun.star.util.SearchDescriptor 서비스가 제공하는 등록 정보는 다음과 같습니다.

- **SearchBackwards (Boolean)** - 텍스트를 앞으로 검색하지 않고 뒤로 검색합니다.
- **SearchCaseSensitive (Boolean)** - 대/소문자를 구분하면서 검색합니다.
- **SearchRegularExpression (Boolean)** - 검색 식을 정규식처럼 처리합니다.
- **SearchStyles (Boolean)** - 지정된 단락 서식 파일을 찾아 텍스트를 검색합니다.
- **SearchWords (Boolean)** - 완전한 단어만 검색합니다.

StarSuite Basic 에서는 StarSuite SearchSimilarity(즉 "fuzzy match") 함수도 사용할 수 있습니다. 이 함수를 사용할 경우 StarSuite 는 검색 식과 유사하지만 완전히 일치하지 않는 식을 검색합니다. 이 식에서 추가, 삭제, 수정한 문자의 수는 개별적으로 지정할 수 있습니다.
com.sun.star.util.SearchDescriptor 서비스의 관련 등록 정보를 소개합니다.

- **SearchSimilarity (Boolean)** - 유사 검색을 수행합니다.
- **SearchSimilarityAdd (Short)** - 유사 검색에 추가할 수 있는 문자 수
- **SearchSimilarityExchange (Short)** - 유사 검색의 일부로 바꿀 수 있는 문자 수
- **SearchSimilarityRemove (Short)** - 유사 검색의 일부로 제거할 수 있는 문자 수
- **SearchSimilarityRelax (Boolean)** - 검색 식에 대해 모든 변형 규칙을 동시에 고려합니다.

요청한 대로 SearchDescriptor 를 준비하면 텍스트 문서에 적용할 수 있습니다. 이를 위해 StarSuite 문서는 findFirst 및 findNext 메소드를 제공합니다.

```
Found = Doc.findFirst (SearchDesc)

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

위의 예는 루프를 통해 일치하는 항목을 모두 찾고, 찾은 텍스트 절을 참조하는 TextRange 개체를 구합니다.

예: 유사 검색

이 예는 "turnover"라는 단어를 찾아 텍스트를 검색하고 해당 결과를 굵게 표시하는 방법을 보여 줍니다. “turnover”라는 단어 뿐 아니라 복수형 "turnovers", "turnover’s"처럼 변형된 형태도 찾도록 유사 검색을 사용합니다. 찾은 식은 검색 식과 비교하여 최대 2 문자까지 다를 수 있습니다.

```
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
```

StarSuite 의 찾기 및 바꾸기의 기본 개념을 VBA 의 개념과 비교할 수 있습니다. 두 인터페이스 모두 찾기 및 바꾸기 등록 정보를 지정할 수 있는 개체를 제공합니다. 그리고 이 개체를 필요한 텍스트 영역에 적용하여 작업을 수행할 수 있습니다. 한편 VBA 에서는 해당 보조 개체를 Range 개체의 `Find` 등록 정보를 통해 액세스할 수 있지만, StarSuite Basic 에서는 문서 개체의 `createSearchDescriptor` 또는 `createReplaceDescriptor` 호출을 통해 만든다는 점이 다릅니다. 사용 가능한 검색 등록 정보 및 메소드도 서로 다릅니다.

기존 StarSuite API 처럼 새 API 의 텍스트 찾기 및 바꾸기 역시 문서 개체를 사용하여 수행합니다. 이전에는 이러한 검색 옵션을 지정하기 위해 특별히 `SearchSettings` 라는 개체가 있었지만, 새로운 API 에서는 `SearchDescriptor` 또는 `ReplaceDescriptor` 개체를 사용하여 검색을 수행하고 텍스트를 자동으로 바꿉니다. 이 개체들은 옵션뿐 아니라 현재 검색 텍스트, 그리고 필요하다면 대체 텍스트까지도 대상으로 합니다. 이 설명자 개체는 문서 개체를 사용하여 생성하고, 관련 요청에 따라 완성하며, 검색 메소드의 매개변수 형태로 문서 개체에 다시 전달합니다.

텍스트 부분 바꾸기

StarSuite Basic 에서는 StarSuite 검색 함수와 마찬가지로 바꾸기 함수도 제공합니다. 두 함수는 유사하게 처리됩니다. 바꾸기 프로세스의 경우에도 우선 매개 변수를 기록하는 특별한 개체가 필요합니다. `ReplaceDescriptor` 라고 부르는 이 개체는 `com.sun.star.util.ReplaceDescriptor` 서비스를 지원합니다. 위 단락에서 설명한 `SearchDescriptor` 의 모든 등록 정보는 `ReplaceDescriptor` 에서도 지원합니다. 예를 들어 바꾸기 프로세스에서 대/소문자 구분을 활성화하거나 비활성화하고 유사 검색을 수행할 수 있습니다.

다음은 StarSuite 문서 내부의 검색에 `ReplaceDescriptor` 를 사용하는 예입니다.

```
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
```

찾기 및 바꾸기 식은 ReplaceDescriptor 의 `SearchString` 및 `ReplaceString` 등록 정보를 사용하여 설정합니다. 실제 바꾸기 프로세스는 검색 식을 모두 바꾸는 문서 개체의 `replaceAll` 메소드를 사용하여 최종 구현합니다.

예: 정규식을 통한 텍스트 찾기 및 바꾸기

StarSuite 의 바꾸기 함수는 정규식과 함께 사용하면 특히 효과적입니다. 고정 값 대신 자리 표시자와 특수 문자를 사용하여 변수 검색 식을 지정할 수 있습니다.

StarSuite 가 지원하는 정규식은 StarSuite 의 온라인 도움말 부분에서 자세히 설명합니다. 몇 가지 예를 소개합니다.

- 검색 식 내부의 마침표는 임의의 문자를 의미합니다. 따라서 검색 식 `sh.rt` 는 `shirt` 또는 `short` 를 의미할 수 있습니다.
- 문자 ^는 단락의 시작을 나타냅니다. 따라서 단락 시작 부분에 나오는 `Peter` 라는 이름은 검색 식 `^Peter` 를 사용하여 모두 찾을 수 있습니다.

- 문자 $는 단락의 끝을 의미합니다. 따라서 단락의 끝에 나타나는 Peter 라는 이름은 검색 식 Peter$를 사용하여 모두 찾을 수 있습니다.
- *는 이전 문자를 횟수에 관계 없이 반복할 수 있음을 나타냅니다. 임의의 문자에 대해 자리 표시자인 마침표와 함께 사용할 수 있습니다. 예를 들어 temper.*e 식은 temperance 또는 temperature 를 나타낼 수 있습니다.

다음은 정규식 ^$를 사용하여 텍스트 문서의 모든 빈 줄을 제거하는 방법을 보여 주는 예입니다.

```
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)
```

텍스트 문서: 텍스트 이외의 요소

이 장에서는 지금까지 텍스트 단락과 해당 부분에 대해서만 설명했습니다. 그러나 텍스트 문서는 다른 개체도 포함할 수 있습니다. 여기에는 표, 그림, 텍스트 필드 및 디렉토리가 포함됩니다. 이 개체는 모두 텍스트 내부의 어느 곳에서도 기준 위치를 정할 수 있습니다.

이러한 공통적인 특징 덕분에 StarSuite 의 이 개체들은 모두 com.sun.star.text.TextContent 라는 이름의 공통 기본 서비스를 지원합니다. 이 서비스는 다음과 같은 등록 정보를 제공합니다.

- **AnchorType (Enum)** – TextContent 개체의 기준 위치 유형을 지정합니다. 기본값은 com.sun.star.text.TextContentAnchorType enumeration 에 따라 지정됩니다.
- **AnchorTypes (sequence of Enum)** – 특정 TextContent 개체를 지원하는 모든 AnchorType 의 목록
- **TextWrap (Enum)** – TextContent 개체 주위의 텍스트 배치 유형을 지정합니다. 기본값은 com.sun.star.text.WrapTextMode enumeration 에 따라 지정됩니다.

또한 TextContent 개체는 몇 가지 메소드, 특히 개체 생성, 삽입, 삭제용 메소드를 공유합니다.

- 새로운 TextContent 개체는 문서 개체의 createInstance 메소드를 사용하여 만들 수 있습니다.
- 개체는 텍스트 개체의 insertTextContent 메소드를 사용하여 삽입할 수 있습니다.
- TextContent 개체는 removeTextContent 메소드를 사용하여 삭제할 수 있습니다.

다음 섹션에서는 이 메소드를 사용하는 몇 가지 예를 소개합니다.

표

다음의 예에서는 이미 설명한 바 있는 `createInstance` 메소드를 사용하여 표를 만듭니다.

```
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)
```

일단 작성된 표는 `initialize` 를 호출하여 열과 행 수를 설정하고 `insertTextContent` 를 사용하여 텍스트 문서에 삽입합니다.

이 예에서 볼 수 있듯이 `insertTextContent` 메소드는 삽입할 `Content` 개체뿐 아니라 다른 두 매개 변수를 필요로 합니다.

- 삽입 위치를 지정하는 `Cursor` 개체
- 불리언 변수 - `Content` 개체가 현재 커서 선택 영역을 바꿀 것인지(`True`) 또는 텍스트 중 현재 선택 영역 앞에 삽입될 것인지(`False`) 지정합니다.

텍스트 문서에서 표를 만들거나 삽입할 때 StarSuite Basic 은 VBA 의 개체와 유사한 개체를 사용합니다. StarSuite Basic 의 문서 개체 및 `TextCursor` 개체는 VBA 의 `Range` 개체에 대응합니다. VBA 에서는 `Document.Tables.Add` 메소드가 표를 만들고 설정하는 일을 담당하지만 StarSuite Basic 에서는 위의 예처럼 `createInstance` 를 사용하여 표를 만들고 `insertTextContent` 를 사용하여 표를 초기화하고 문서에 삽입합니다.

텍스트 문서에 삽입한 표는 간단한 루프를 통해 지정할 수 있습니다. 이를 위해 텍스트 문서 개체의 `getTextTables()` 메소드를 사용합니다.

```
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)

      ' 표 편집
Next I
```

텍스트 표는 StarSuite 7 에서 문서 개체의 `TextTables` 목록을 통해 사용할 수 있습니다. 이는 과거 `Selection` 개체가 제공하던 표 목록을 대체합니다. 이전 예에서는 텍스트 표를 만드는 방법을 보여 줍니

다. 텍스트 표 액세스 옵션은 다음 섹션에서 자세히 설명합니다.

표 편집

표는 개별 행으로 구성됩니다. 각 행은 여러 셀을 포함합니다. 엄밀히 말해 StarSuite 에는 표의 열이 없습니다. 행을 상하로 나란히 배열하면서 암시적으로 열이 형성됩니다. 그러나 StarSuite 는 표 액세스를 간소화하기 위해 열을 사용하는 메소드 몇 가지를 제공합니다. 이는 표에서 병합된 셀이 없을 경우 유용합니다.

먼저 표 자체의 등록 정보에 대해 알아 봅니다. 표 등록 정보는 `com.sun.star.text.TextTable` 서비스에서 지정합니다. 표 개체의 가장 중요한 등록 정보를 소개합니다.

- **`BackColor (Long)`** – 표 배경 색상입니다.
- **`BottomMargin (Long)`** – 1/100 밀리미터 단위의 아래쪽 여백입니다.
- **`LeftMargin (Long)`** – 1/100 밀리미터 단위의 왼쪽 여백입니다.
- **`RightMargin (Long)`** – 1/100 밀리미터 단위의 오른쪽 여백입니다.
- **`TopMargin (Long)`** – 1/100 밀리미터 단위의 위쪽 여백입니다.
- **`RepeatHeadline (Boolean)`** – 표 머리말이 모든 페이지에 반복됩니다.
- **`Width (Long)`** – 1/100 밀리미터 단위의 표의 절대 너비입니다.

행

표는 행을 포함하는 목록으로 구성됩니다. 다음의 예는 표의 행을 검색하고 서식 설정하는 방법을 보여 줍니다.

```
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
```

위의 예에서는 먼저 `Table.getRows` 를 호출하여 모든 행을 포함하는 목록을 만듭니다. `getCount` 및 `getByIndex` 메소드는 향후 목록 처리를 지원하며

com.sun.star.table.XtableRows 인터페이스에 속합니다. getByIndex 메소드는 com.sun.star.text.TextTableRow 서비스를 지원하는 행 개체를 구합니다.

com.sun.star.table.XtableRows 인터페이스의 주요 메소드를 소개합니다.

- **getByIndex(Integer)** – 지정된 색인의 행 개체를 구합니다.
- **getCount()** – 행 개체의 수를 구합니다.
- **insertByIndex(Index, Count)** – 표에서 Count 개체의 행을 Index 위치에 삽입합니다.
- **removeByIndex(Index, Count)** – 표에서 Count 개체의 행을 Index 위치에서 삭제합니다.

getByIndex 및 getCount 메소드는 모든 표에서 사용할 수 있지만, insertByIndex 및 removeByIndex 메소드는 병합된 셀이 없는 표에서만 사용할 수 있습니다.

com.sun.star.text.TextTableRow 서비스는 다음과 같은 등록 정보를 제공합니다.

- **BackColor (Long)** – 행 배경 색상
- **Height (Long)** – 1/100 밀리미터 단위의 줄 높이
- **IsAutoHeight (Boolean)** – 표 높이를 내용에 맞게 동적으로 조정합니다.
- **VertOrient (const)** – 텍스트 프레임의 세로 방향 – 표 내부의 텍스트 세로 방향에 대한 세부 정보(com.sun.star.text.VertOrientation 을 따르는 값)

열

열은 행과 똑같이 Column 개체에 대해 getByIndex, getCount, insertByIndex, removeByIndex 메소드를 사용하여 액세스하며, 이 개체는 getColumns 를 통해 액세스합니다. 그러나 셀이 병합되지 않은 표에서만 사용할 수 있습니다. StarSuite Basic 에서는 열을 통해 셀을 서식 설정할 수 없습니다. 이를 위해 각 표 셀의 서식을 설정하는 메소드를 사용해야 합니다.

셀

StarSuite 문서의 각 셀은 고유한 이름을 갖습니다. StarSuite 의 커서가 셀 안에 있으면 해당 셀 이름을 상태 표시줄에서 확인할 수 있습니다. 맨 위 왼쪽 셀은 보통 A1 이라고 부르고, 맨 아래 오른쪽 행은 Xn 이라고 부르며, 여기에서 X 는 맨 위 열의 문자를, n 은 마지막 행 번호를 의미합니다. 셀 개체는 표 개체의 `getCellByName()` 메소드를 통해 사용합니다. 다음의 예는 표의 모든 셀에 대해 루프를 실행하면서 해당 행 및 열 번호를 셀에 입력하는 작업을 보여 줍니다.

```
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
```

표 셀은 표준 텍스트와 유사합니다. 이는 `createTextCursor` 인터페이스를 지원하므로 관련 `TextCursor` 개체를 만들 수 있습니다.

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

따라서 개별 문자 및 단락의 서식 설정 옵션을 모두 사용할 수 있습니다.

다음의 예는 텍스트 문서의 모든 표를 검색하면서 숫자 값을 갖는 모든 셀을 찾아 해당 단락 등록 정보를 사용하여 오른쪽 맞춤 서식을 적용합니다.

```
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
```

이 예에서는 루프를 실행하는 텍스트의 모든 표로 구성된 TextTables 목록을 만듭니다. 그리고 나서 StarSuite 는 각 표에 대해 해당 셀 이름의 목록을 만듭니다. 여기에서도 루프를 실행합니다. 셀이 숫자 값을 포함하면 위의 예는 서식 설정을 바꿉니다. 이를 위해 먼저 표 셀의 내용을 참조하는 TextCursor 개체를 만들고 표 셀의 단락 등록 정보를 조정합니다.

텍스트 프레임

텍스트 프레임은 표와 그래프처럼 TextContent 개체로 간주합니다. 기본적으로 표준 텍스트로 구성될 수 있지만, 페이지 상의 어느 곳에도 위치할 수 있으며 텍스트 흐름에는 포함되지 않습니다.

모든 TextContent 개체와 마찬가지로 텍스트 프레임에서도 실제 만드는 프로세스와 문서에 삽입하는 프로세스가 구분됩니다.

```
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)
```

텍스트 프레임은 문서 개체의 createInstance 메소드를 사용하여 만듭니다. 이렇게 만든 텍스트 프레임은 텍스트 개체의 insertTextContent 메소드를 사용하여 문서에 삽입할 수 있습니다. 이를 위해 적절한 com.sun.star.text.TextFrame 서비스 이름을 지정해야 합니다.

텍스트 프레임의 삽입 위치는 삽입 시 함께 실행되는 Cursor 개체에 의해 결정됩니다.

StarSuite 의 텍스트 프레임은 Word 의 위치 프레임에 대응합니다. VBA 는 이를 위해

`Document.Frames.Add` 메소드를 사용하지만, VBA 에서는 TextCursor 및 문서 개체의 `createInstance` 메소드와 함께 위의 절차를 사용하여 생성합니다.

텍스트 프레임 개체는 프레임의 위치 및 동작에 영향을 주는 다양한 등록 정보를 제공합니다. 이 등록 정보 중 대부분은 `com.sun.star.text.BaseFrameProperties` 서비스에서 지정하며, 각 TextFrame 서비스 역시 이 서비스를 지원합니다. 주요 등록 정보는 다음과 같습니다.

- **BackColor (Long)** – 텍스트 프레임의 배경 색상
- **BottomMargin (Long)** – 1/100 밀리미터 단위의 아래쪽 여백
- **LeftMargin (Long)** – 1/100 밀리미터 단위의 왼쪽 여백
- **RightMargin (Long)** – 1/100 밀리미터 단위의 오른쪽 여백
- **TopMargin (Long)** – 1/100 밀리미터 단위의 위쪽 여백
- **Height (Long)** – 1/100 밀리미터 단위의 텍스트 프레임 높이
- **Width (Long)** – 1/100 밀리미터 단위의 텍스트 프레임 너비
- **HoriOrient (const)** – 텍스트 프레임의 가로 방향 (`com.sun.star.text.HoriOrientation` 을 따름)
- **VertOrient (const)** – 텍스트 프레임의 세로 방향 (`com.sun.star.text.VertOrientation` 을 따름)

다음의 예는 위 등록 정보를 사용하여 텍스트 프레임을 만듭니다.

```
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)
```

이 예에서는 텍스트 프레임의 삽입 표시로 `TextCursor`를 만듭니다. 텍스트의 첫 번째 단어와 두 번째 단어 사이에 위치합니다. 텍스트 프레임은 `Doc.createInstance`를 사용하여 만듭니다. 텍스트 프레임 개체의 등록 정보는 필수 시작 값으로 설정됩니다.

여기에서 `AnchorType`(`TextContent` Service)과 `VertOrient`(`BaseFrameProperties` Service) 등록 정보 사이의 상호 작용을 설명할 필요가 있습니다. `AnchorType`은 `AS_CHARACTER` 값을 받습니다. 따라서 텍스트 프레임은 텍스트 흐름에 직접 삽입되고 문자처럼 동작합니다. 예를 들어 줄바꿈이 수행되면 다음 줄로 이동할 수 있습니다. VertOrient 등록 정보의 `LINE_TOP` 값은 텍스트 프레임의 위쪽 가장자리가 문자의 위쪽 가장자리와 같은 높이가 되게 합니다.

초기화가 완료되면 마지막으로 `insertTextContent` 호출을 통해 텍스트 프레임을 텍스트 문서에 삽입합니다.

텍스트 프레임의 내용을 편집하려면 이미 여러 차례 언급한 바 있고 텍스트 프레임에서도 사용 가능한 TextCursor 를 사용합니다.

```
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!"
```

이 예에서는 텍스트 프레임을 만들어 현재 문서에 삽입하고, 해당 텍스트 프레임의 TextCursor 를 엽니다. 이 커서는 프레임 글꼴을 굵게 표시하고 단락을 가운데 맞춤으로 설정하는 데 사용합니다. 마지막으로 텍스트 프레임에 “This is a small test!”라는 문자열이 할당됩니다.

텍스트 필드

텍스트 필드는 순수 텍스트 이외의 추가 논리를 제공하므로 TextContent 개체에 해당됩니다. 텍스트 필드는 다른 TextContent 개체와 같은 메소드를 사용하여 텍스트 문서에 삽입합니다.

```
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)
```

이 예에서는 현재 텍스트 문서의 시작 위치에 현재 날짜가 입력된 텍스트 필드를 삽입합니다. IsDate 등록 정보의 True 값은 날짜만 제공하며 시간은 표시하지 않습니다. IsFixed 의 False 값은 문서를 열 때 해당 날짜를 자동 업데이트하게 합니다.

VBA 에서는 필드 유형을 Document.Fields.Add 메소드의 매개 변수가 지정하지만, StarSuite Basic 에서는 해당 필드 유형을 담당하는 서비스 이름이 해당 유형을 지정합니다.

예전에는 StarSuite 가 `Selection` 개체에서 제공한 메소드를 통해 텍스트 필드를 액세스했습니다(예: `InsertField`, `DeleteUserField`, `SetCurField`).

StarSuite 7 에서는 개체 지향 개념을 사용하여 필드를 관리합니다. 텍스트 필드를 만들려면 먼저 필요한 유형의 텍스트 필드를 만들고 필요한 등록 정보를 사용하여 초기화해야 합니다. 그리고 나서 `insertTextContent` 메소드를 사용하여 텍스트 필드를 문서에 삽입합니다. 관련 원본 텍스트는 이전 예에서 확인할 수 있습니다. 가장 중요한 필드 유형 및 해당 등록 정보는 다음 섹션에서 설명합니다.

텍스트 필드 삽입 외에도 문서에서 필드를 검색하는 것도 중요한 작업이 될 수 있습니다. 다음의 예는 루프를 통해 텍스트 문서의 모든 텍스트 필드를 검색하고 관련 유형을 확인하는 방법을 보여 줍니다.

```
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
```

텍스트 필드가 존재하도록 설정하는 시작점은 문서 개체의 `TextFields` 목록입니다. 이 예에서는 이 목록을 바탕으로 `Enumeration` 개체를 만들고, 이 개체를 사용하여 모든 텍스트 필드를 쿼리할 수 있는 루프를 실행합니다. 텍스트 필드를 찾으면 `supportsService` 메소드를 사용하여 어떤 서비스를 지원하는지 확인합니다. 해당 필드가 날짜/시간 필드이거나 설명 필드인 경우 해당 필드 유형을 정보 상자에 표시합니다. 한편 위의 예처럼 또 다른 필드를 찾으면 "unknown"이라고 정보를 표시합니다.

아래에는 가장 중요한 텍스트 필드 및 관련 등록 정보의 목록이 나와 있습니다. 모든 텍스트 필드를 수록한 전체 목록은 `com.sun.star.text.TextField` 모듈의 API 참조 설명서에 게재되어 있습니다. StarSuite Basic 에서는 텍스트 필드의 서비스 이름을 표시할 때 이전 예에서처럼 대/소문자를 구분하여 사용해야 합니다.

페이지, 단어, 문자의 수

다음 텍스트 필드는

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

텍스트의 페이지, 단어, 문자의 수를 구합니다. 이 필드들은 다음과 같은 등록 정보를 지원합니다.

- **NumberingType (const)** - 번호 매기기 서식(com.sun.star.style.NumberingType 상수의 지침을 따름)

현재 페이지

com.sun.star.text.TextField.PageNumber 텍스트 필드를 사용하여 현재 페이지 수를 문서에 삽입할 수 있습니다. 다음 등록 정보를 지정할 수 있습니다.

- **NumberingType (const)** - 숫자 표기 형식(com.sun.star.style.NumberingType 상수의 지침을 따름)
- **Offset (short)** – 페이지 번호에 오프셋 추가(음수 값도 지정 가능)

다음의 예는 페이지 번호를 문서 바닥글에 삽입하는 방법을 보여 줍니다.

```
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)
```

이 예는 먼저 com.sun.star.text.TextField.PageNumber 서비스를 지원하는 텍스트 필드를 만듭니다. StarSuite 의 페이지 서식 파일에서 머리글과 바닥글 줄을 지정하므로, 모든 PageStyles 의 목록을 사용하여 초기에 설정합니다.

바닥글 줄을 표시하려면 FooterIsOn 등록 정보를 True 로 설정해야 합니다. 그런 다음 왼쪽 바닥글 줄의 텍스트 개체를 사용하여 텍스트 필드를 문서에 삽입합니다.

설명

설명 필드(com.sun.star.text.TextField.Annotation)는 텍스트의 작은 황색 기호를 사용하여 표시합니다. 이 기호를 누르면 텍스트의 현재 위치에 대한 설명을 기록할 수 있는 텍스트 필드가 열립니다. 설명 필드는 다음과 같은 등록 정보를 갖습니다.

- **Author (String)** - 작성자 이름
- **Content (String)** - 설명 텍스트
- **Date (Date)** - 설명을 작성한 날짜

날짜 / 시간

날짜 / 시간 필드(com.sun.star.text.TextField.DateTime)는 현재 날짜나 시간을 나타냅니다. 다음과 같은 등록 정보를 지원합니다.

- **IsFixed (Boolean)** – True 인 경우 삽입한 시간 세부 정보가 바뀌지 않으며, False 인 경우 문서를 열 때마다 해당 정보를 업데이트합니다.
- **IsDate (Boolean)** – if True 인 경우 현재 날짜를, 그 밖의 경우에는 현재 시간을 표시합니다.
- **DateTimeValue (struct)** – 현재 필드 내용 (com.sun.star.util.DateTime 구조)
- **NumberFormat (const)** – 시간이나 날짜의 표기 형식

장 이름 / 번호

현재 장의 이름을 com.sun.star.text.TextField.Chapter 유형의 텍스트 필드를 통해 사용할 수 있습니다. 이 서식은 두 등록 정보를 사용하여 지정할 수 있습니다.

- **ChapterFormat (const)** – 장 이름이나 장 번호의 표시 여부를 지정합니다(com.sun.star.text.ChapterFormat 을 따름).
- **Level (Integer)** – 이름이나 번호를 표시할 장 수준을 지정합니다. 값 0 은 가장 높은 수준입니다.

책갈피

책갈피(Service `com.sun.star.text.Bookmark`)는 `TextContent` 개체입니다. 책갈피는 이미 설명한 바 있는 개념을 사용하여 만들고 삽입합니다.

```
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)
```

위의 예는 책갈피 삽입 위치를 표시하는 커서 및 실제 책갈피 개체(`Bookmark`)를 만듭니다. 그리고 나서 책갈피에 이름을 할당한 다음 커서 위치에서 `insertTextContent` 를 사용하여 문서에 삽입합니다.

텍스트의 책갈피는 `Bookmarks` 라는 목록을 통해 액세스합니다. 책갈피는 번호나 이름으로 액세스할 수 있습니다.

다음의 예는 텍스트 내부에서 책갈피를 찾고 해당 위치에 텍스트를 삽입하는 방법을 보여 줍니다.

```
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"
```

이 예에서는 `getByName` 메소드를 사용하여 이름을 기준으로 필요한 책갈피를 찾습니다. 그리고 나서 `createTextCursorByRange` 를 호출하여 커서를 만들고, 이를 책갈피의 기준 위치에 둡니다. 그런 다음 커서는 필요한 텍스트를 해당 위치에 삽입합니다.

7 장

스프레드시트 문서

StarSuite Basic 은 프로그램으로 제어되는 스프레드시트의 작성 및 편집을 위하여 광범위한 인터페이스를 제공합니다. 이 장에서는 스프레드시트 문서의 관련 서비스, 메소드 및 등록 정보를 제어하는 방법을 설명합니다.

첫 번째 섹션에서는 스프레드시트 문서의 기본 구조를 살펴보고 개별 셀의 내용을 액세스 및 편집하는 방법을 설명합니다.

두 번째 섹션에서는 셀 영역에 초점을 맞춰 스프레드시트를 효율적으로 편집하는 방법과 셀 내용 검색하여 바꾸는 옵션을 중점적으로 설명합니다.

새 API 에서 확장된 Range 개체를 사용하면 모든 표 영역의 주소를 지정할 수 있습니다.

표 기반 문서(스프레드시트)의 구조

스프레드시트의 문서 개체는 `com.sun.star.sheet.SpreadsheetDocument` 서비스에 기초합니다. 이러한 각 문서는 여러 스프레드시트를 포함할 수 있습니다. 이 설명서에서 *표 기반 문서* 또는 *스프레드시트 문서*는 전체 문서를 말하며, *스프레드시트*(또는 *시트*)는 문서 내의 시트(표)를 가리킵니다.

VBA 와 StarSuite Basic 에서는 스프레드시트와 그 내용에 대한 용어가 다릅니다. VBA 에서 문서 개체를 *통합 문서*라 하고, 개별 페이지를 *워크시트*라고 하는 것과 달리 StarSuite Basic 에서는 각각 *스프레드시트 문서*와 *시트*라고 합니다.

스프레드시트

`Sheets` 목록을 통해 스프레드시트 문서의 개별 시트에 액세스할 수 있습니다.

다음의 예에서는 번호나 이름을 통해 시트에 액세스하는 방법을 보여 줍니다.

예 1: 번호를 통해 액세스(시작 번호: 0)

```
Dim Doc As Object
Dim Sheet As Object

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

예 2: 이름을 통해 액세스

```
Dim Doc As Object
Dim Sheet As Object

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

첫 번째 예에서는 시트를 번호를 통해 액세스합니다(시작 번호: 0). 두 번째 예에서는 이름과 `getByName` 메소드를 통해 시트에 액세스합니다.

`getByName` 메소드를 통해 얻은 `Sheet` 개체는 `com.sun.star.sheet.Spreadsheet` 서비스를 지원합니다. 내용 편집에 사용되는 다양한 인터페이스를 제공하는 것 외에도 이 서비스는 다음 등록 정보를 제공합니다.

- **`IsVisible (Boolean)`** - 스프레드시트를 볼 수 있습니다.
- **`PageStyle (String)`** - 스프레드시트에 사용되는 페이지 서식 파일의 이름입니다.

시트 작성, 삭제 및 이름 바꾸기

또한 스프레드시트 문서의 `Sheets` 목록을 사용하여 개별 시트의 작성, 삭제 및 이름 바꾸기를 수행합니다. 다음 예에서는 `hasByName` 메소드를 사용하여 *MySheet* 라는 시트가 존재하는지 확인합니다. 해당 시트가 존재할 경우 이 메소드는 `getByName` 메소드를 사용하여 해당 개체 참조를 확인한 다음 `Sheet` 의 변수에 참조를 저장합니다. 해당 시트가 존재하지 않을 경우 `createInstance` 호출을 통해 시트가 만들어지고 `insertByName` 메소드에 의해 스프레드시트 문서에 삽입됩니다.

```
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
```

4장에 설명된 것처럼 `getByName` 및 `insertByName` 메소드는 `com.sun.star.container.XnameContainer` 인터페이스에 포함되어 있습니다.

행과 열

각 시트는 행과 열 목록을 포함합니다. 행과 열은 스프레드시트 개체의 `Rows` 및 `Columns` 등록 정보를 통해 사용할 수 있으며 `com.sun.star.table.TableColumns` 및/또는 `com.sun.star.table.TableRows` 서비스를 지원합니다.

다음의 예에서는 시트의 첫 번째 행과 첫 번째 열을 참조하는 두 개의 개체를 만들고 `FirstCol` 및 `FirstRow` 개체 변수에 이러한 참조를 저장합니다.

```
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)
```

열 개체는 다음 등록 정보가 있는 `com.sun.star.table.TableColumn` 서비스를 지원합니다.

- **`Width (long)`** - 열 너비입니다(단위: 0.01mm).
- **`OptimalWidth (Boolean)`** - 열을 최적 너비로 설정합니다.
- **`IsVisible (Boolean)`** - 열을 표시합니다.
- **`IsStartOfNewPage (Boolean)`** - 인쇄 시 열 앞에 페이지 나누기를 만듭니다.

열 너비는 `OptimalWidth` 등록 정보가 `True`로 설정된 경우에만 최적화됩니다. 개별 셀의 너비가 바뀔 경우 해당 셀을 포함하는 열의 너비는 바뀌지 않습니다. 기능과 관련하여 `OptimalWidth`는 등록 정보보다 메소드에 가깝습니다.

행 개체는 다음 등록 정보가 있는 `com.sun.star.table.RowColumn` 서비스에 기초합니다.

- **`Height (long)`** - 행 높이입니다(단위: 0.01mm).
- **`OptimalHeight (Boolean)`** - 행을 최적 높이로 설정합니다.
- **`IsVisible (Boolean)`** - 행을 표시합니다.
- **`IsStartOfNewPage (Boolean)`** - 인쇄 시 행 앞에 페이지 나누기를 만듭니다.

행의 `OptimalHeight` 등록 정보가 `True`로 설정될 경우 행의 셀 높이가 바뀔 때 행 높이가 자동으로 바뀝니다. `Height` 등록 정보를 통해 절대 높이를 행에 할당할 때까지 자동 최적화가 계속됩니다.

다음은 시트의 처음 5 개 행에 대해 자동 높이 최적화를 활성화하고 두 번째 열을 보이지 않게 만드는 예입니다.

```
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
```

StarSuite Basic 에서 Rows 및 Columns 목록을 색인을 통해 액세스할 수 있습니다. VBA 와 달리 첫 번째 열은 색인 1 이 아니라 색인 0 을 가집니다.

행과 열 삽입 및 삭제

시트의 Rows 및 Columns 개체는 기존 행과 열에 액세스할 수 있을 뿐만 아니라 기존 행과 열을 삽입 및 삭제할 수 있습니다.

```
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)
```

이 예는 insertByIndex 메소드를 사용하여 시트의 네 번째 열 위치(색인 3 - 시작 번호: 0)에 새 열을 삽입합니다. 두 번째 매개 변수는 삽입할 열 수(이 예에서는 1 개)를 지정합니다.

removeByIndex 메소드는 여섯 번째 열(색인 5)을 삭제합니다. 이번에도 두 번째 매개 변수는 삭제할 열 수를 지정합니다.

행 삽입 및 삭제를 위한 메소드는 Columns 개체를 사용하여 열을 편집하기 위한 메소드와 같은 방법으로 Rows 개체 함수를 사용합니다.

셀

스프레드시트는 셀을 포함하는 2 차원 목록으로 구성됩니다. 각 셀은 위치 (0,0)을 가지는 왼쪽 위 셀을 기준으로 한 X 및 Y 위치로 지정됩니다.

다음은 왼쪽 위 셀을 참조하는 개체를 만들고 텍스트를 셀에 삽입하는 예입니다.

```
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"
```

숫자 좌표 외에 시트의 각 셀은 이름을 가집니다. 예를 들어, 스프레드시트의 왼쪽 위 셀(0,0)을 `A1` 이라고 부릅니다. 문자 `A` 는 열을 나타내며 숫자 `1` 은 행을 나타냅니다. 이름의 행 번호가 1 부터 시작하는 것과 달리 위치 번호는 0 부터 시작하기 때문에 셀의 *이름*과 *위치*를 혼동하지 않는 것이 중요합니다.

StarSuite 에서 표 셀은 비어 있거나 텍스트, 숫자 또는 수식을 포함할 수 있습니다. 셀 유형은 셀에 저장되는 내용이 아니라 입력에 사용되었던 개체 등록 정보에 의해 결정됩니다. 숫자는 `Value` 등록 정보를, 텍스트는 `String` 등록 정보를, 수식은 `Formula` 등록 정보를 사용하여 삽입 및 호출할 수 있습니다.

```
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"
```

이 예는 A1 에서 A3 필드까지 숫자 하나, 텍스트 하나, 수식 하나를 삽입합니다.

`Value`, `String` 및 `Formula` 등록 정보는 표 셀의 값을 설정하기 위한 `PutCell` 메소드를 대체합니다.

StarSuite 는 `String` 등록 정보를 사용하여 입력한 셀 내용을 텍스트로 간주하며 이는 해당 내용이 숫자인 경우에도 마찬가지입니다. 이러한 방법으로 입력한 숫자는 셀에서 오른쪽 대신 왼쪽으로 맞춰집니다. 또한 수식을 사용할 때 텍스트와 숫자의 차이점에 주의해야 합니다.

```
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
```

셀 A1 이 100 값을 포함하고 셀 A2 가 1,000 값을 포함하지만 A1+A2 수식은 100 값을 구합니다. 이것은 셀 A2 의 내용이 숫자가 아니라 문자열로 입력되었기 때문입니다.

셀 내용에 숫자나 문자열이 포함되었는지 확인하려면 다음과 같이 Type 등록 정보를 사용합니다.

```
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 등록 정보는 셀의 내용 유형을 식별하는 com.sun.star.table.CellContentType 열거에 대한 값을 구합니다. 가능한 값은 다음과 같습니다.

- **EMPTY** - 값 없음
- **VALUE** - 숫자
- **TEXT** - 문자열
- **FORMULA** - 수식

셀 삽입, 삭제, 복사 및 이동

셀 내용을 직접 수정하는 것 외에도 StarSuite Calc 는 셀을 삽입, 삭제, 복사 또는 병합할 수 있는 인터페이스를 제공합니다. 이 인터페이스(`com.sun.star.sheet.XRangeMovement`)는 스프레드시트 개체를 통해 사용할 수 있으며 셀 내용 수정을 위한 네 가지 메소드를 제공합니다.

`insertCell` 메소드는 셀을 시트에 삽입하는 데 사용됩니다.

```
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)
```

이 예는 스프레드시트의 첫 번째 시트(번호 0)에 있는 각각 번호 1 을 가진 두 번째 열과 행에 행 두 개와 열 두 개의 셀 범위를 삽입합니다. 지정된 셀 범위에 있는 모든 기존 값은 범위 아래로 이동합니다.

삽입할 셀 범위를 지정하려면 `com.sun.star.table.CellRangeAddress` 구조를 사용합니다. 이 구조에는 다음 값이 포함됩니다.

- **`Sheet (short)`** - 시트의 번호입니다(시작 번호: 0).
- **`StartColumn (long)`** - 셀 범위의 첫 번째 열입니다(시작 번호: 0).
- **`StartRow (long)`** - 셀 범위의 첫 번째 행입니다(시작 번호: 0).
- **`EndColumn (long)`** - 셀 범위의 마지막 열입니다(시작 번호: 0).
- **`EndRow (long)`** - 셀 범위의 마지막 행입니다(시작 번호: 0).

완성된 `CellRangeAddress` 구조는 첫 번째 매개 변수로 `insertCells` 메소드에 전달되어야 합니다. `insertCells` 의 두 번째 매개 변수는 `com.sun.star.sheet.CellInsertMode` 열거의 값을 포함하며 삽입 위치 앞에 있는 값에 대해 수행할 작업을 지정합니다. `CellInsertMode` 열거는 다음 값을 인식합니다.

- **`NONE`** - 현재 값이 현재 위치를 유지합니다.
- **`DOWN`** - 삽입 위치와 그 아래에 있는 셀이 아래쪽으로 이동합니다.
- **`RIGHT`** - 삽입 위치와 그 오른쪽에 있는 셀이 오른쪽으로 이동합니다.
- **`ROWS`** - 삽입 위치 뒤에 있는 행이 아래쪽으로 이동합니다.
- **`COLUMNS`** - 삽입 위치 뒤에 있는 열이 오른쪽으로 이동합니다.

removeRange 메소드는 insertCells 메소드의 상대 기능을 제공합니다. 이 메소드는 시트에서 CellRangeAddress 구조에 지정된 범위를 삭제합니다.

```
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)
```

이 예는 시트에서 B2:C3 셀 범위를 제거한 다음 기본 셀을 위쪽으로 두 행만큼 이동합니다. 제거 유형은 com.sun.star.sheet.CellDeleteMode 열거에서 다음 값 중 하나에 의해 지정됩니다.

- **NONE** - 현재 값이 현재 위치를 유지합니다.
- **UP** - 삽입 위치와 그 아래에 있는 셀이 위쪽으로 이동합니다.
- **LEFT** - 삽입 위치와 그 오른쪽에 있는 셀이 왼쪽으로 이동합니다.
- **ROWS** - 삽입 위치 뒤에 있는 행이 위쪽으로 이동합니다.
- **COLUMNS** - 삽입 위치 뒤에 있는 열이 왼쪽으로 이동합니다.

XRangeMovement 인터페이스는 셀 범위의 이동(moveRange) 또는 복사(copyRange)를 위한 두 가지 추가 메소드를 제공합니다. 다음은 위치 A6 에서 범위가 시작되도록 B2:C3 범위를 이동하는 예입니다.

```
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)
```

CellRangeAdress 구조 외에 moveRange 메소드를 사용하려면 com.sun.star.table.CellAddress 구조에서 이동의 대상 영역에 대한 원점을 지정해야 합니다. CellAddress 메소드는 다음 값을 제공합니다.

- **Sheet (short)** - 스프레드시트의 번호입니다(시작 번호: 0).
- **Column (long)** - 주소가 지정된 열의 번호입니다(시작 번호: 0).
- **Row (long)** - 주소가 지정된 행의 번호입니다(시작 번호: 0).

moveRange 메소드는 항상 대상 범위의 셀 내용을 덮어씁니다.
InsertCells 메소드와 달리 자동 이동을 수행하기 위한 매개 변수는 removeRange 메소드에서 제공되지 않습니다.

copyRange 메소드는 moveRange 메소드와 같은 방법으로 작동합니다. 단, copyRange 가 셀 범위를 이동하는 대신 셀 범위의 복사본을 삽입한다는 점은 제외입니다.

기능과 관련하여 StarSuite Basic insertCell, removeRange 및 copyRange 메소드는 VBA Range.Insert, Range.Delete 및 Range.Copy 메소드와 비슷합니다. VBA 에서 이러한 메소드가 해당 Range 개체에 적용되는 것과 달리 StarSuite Basic 에서는 관련 Sheet 개체에 적용됩니다.

서식 설정

스프레드시트 문서는 셀과 페이지의 서식 설정을 위한 등록 정보와 메소드를 제공합니다.

셀 등록 정보

텍스트의 글꼴 유형과 크기를 지정하는 것과 같은 셀 서식 설정을 위한 많은 옵션이 존재합니다. 각 셀은 6 장(텍스트 문서)에 설명된 주 등록 정보인 `com.sun.star.style.CharacterProperties` 및 `com.sun.star.style.ParagraphProperties` 의 서비스를 지원합니다. 특수한 셀 서식 설정은 `com.sun.star.table.CellProperties` 서비스에 의해 처리됩니다. 이 서비스의 기본 등록 정보는 다음 섹션에 설명되어 있습니다.

명명된 모든 등록 정보를 개별 셀과 셀 범위에 적용할 수 있습니다.

> StarSuite API 의 `CellProperties` 개체는 셀 특정 등록 정보를 지정하는 VBA 의 `Interior` 개체와 비슷합니다.

배경색 및 그림자

`com.sun.star.table.CellProperties` 서비스는 배경색과 그림자를 지정하기 위한 다음 등록 정보를 제공합니다.

- **`CellBackColor (Long)`** - 표 셀의 배경색입니다.
- **`IsCellBackgroundTransparent (Boolean)`** - 배경색을 투명하게 설정합니다.
- **`ShadowFormat (struct)`** - 셀의 그림자를 지정합니다 (`com.sun.star.table.ShadowFormat` 을 따르는 구조).

`com.sun.star.table.ShadowFormat` 구조와 셀 그림자의 세부 지정은 다음 구조를 가집니다.

- **`Location (enum)`** - 그림자의 위치입니다(`com.sun.star.table.ShadowLocation` 구조의 값 사용).
- **`ShadowWidth (Short)`** - 그림자의 크기입니다(단위: 0.01mm).
- **`IsTransparent (Boolean)`** - 그림자를 투명하게 설정합니다.
- **`Color (Long)`** - 그림자의 색상입니다.

다음은 숫자 1,000 을 B2 셀에 기록하고 `CellBackColor` 등록 정보를 사용하여 배경색을 적색으로 바꾼 다음 왼쪽 아래로 1mm 이동한 밝은 회색의 셀 그림자를 만드는 예입니다.

```
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
```

양쪽 맞춤

StarSuite 는 표 셀에서 텍스트의 양쪽 맞춤을 바꿀 수 있는 다양한 기능을 제공합니다.

다음 등록 정보는 텍스트의 가로 및 세로 양쪽 맞춤을 지정합니다.

- **HoriJustify (enum)** - 텍스트의 가로 양쪽 맞춤입니다 (`com.sun.star.table.CellHoriJustify` 의 값 사용).
- **VertJustify (enum)** - 텍스트의 세로 양쪽 맞춤입니다 (`com.sun.star.table.CellVertJustify` 의 값 사용).
- **Orientation (enum)** - 텍스트의 방향입니다(`com.sun.star.table.CellOrientation` 을 따르는 값).
- **IsTextWrapped (Boolean)** - 셀 안에서 자동 줄바꿈이 가능합니다.
- **RotateAngle (Long)** - 텍스트의 회전 각도입니다(단위: 0.01 도).

다음은 셀의 왼쪽 위 모서리에서 문자가 다른 문자 아래에 인쇄되도록 셀 내용을 "포개는" 방법을 보여 주는 예입니다. 문자는 회전하지 않습니다.

```
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
```

숫자, 날짜 및 텍스트 서식

StarSuite 는 미리 지정된 다양한 날짜 및 시간 서식을 제공합니다. 각 서식은 NumberFormat 등록 정보를 사용하여 서식을 셀에 할당하는 데 사용되는 내부 번호를 가집니다. StarSuite 는 기존 숫자 표기 형식을 액세스할 뿐만 아니라 고유한 숫자 표기 형식을 만들 수 있는 queryKey 및 addNew 메소드를 제공합니다. 이러한 메소드는 다음 개체 호출을 통해 액세스합니다.

```
NumberFormats = Doc.NumberFormats
```

서식은 StarSuite Basic 의 서식 기능과 비슷한 방법으로 구성되는 서식 문자열을 사용하여 지정합니다. 그러나 명령 서식에 영어 약자가 오고 천단위 구분 기호로 소수점이나 문자가 사용되는 것과 달리 NumberFormats 개체의 명령 서식 구조에는 국가별 약자를 사용해야 합니다.

다음은 숫자의 소수점 이하 자릿수를 3 개로 표시하고 천단위 구분 기호로 쉼표를 사용하도록 B2 셀의 서식을 설정하는 예입니다.

```
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 의 셀 서식 설정 대화 상자는 셀 서식 설정을 위한 여러 옵션의 개요를 제공합니다.

페이지 등록 정보

페이지 등록 정보는 특정 페이지의 문서 내용뿐만 아니라 페이지마다 반복되는 다음과 같은 시각적 요소를 배치하는 서식 설정 옵션입니다.

- 용지 서식

- 페이지 여백
- 머리글 및 바닥글

페이지 서식을 지정하기 위한 절차는 다른 서식 설정과 다릅니다. 셀, 단락 및 문자 요소를 직접 적용할 수 있는 것과 달리 페이지 서식은 페이지 스타일을 사용하여 지정하고 간접적으로 적용할 수 있습니다. 예를 들어, 머리글과 바닥글은 페이지 스타일에 추가됩니다.

다음 섹션에서는 스프레드시트 페이지의 기본 서식 설정 옵션을 설명합니다. 설명된 스타일은 대개 텍스트 문서에도 사용할 수 있습니다. 두 유형의 문서에 모두 유효한 페이지 등록 정보는 `com.sun.star.style.PageProperties` 서비스에 지정됩니다. 스프레드시트 문서에만 적용되는 페이지 등록 정보는 `com.sun.star.sheet.TablePageStyle` 서비스에 지정됩니다.

Microsoft Office 문서의 페이지 등록 정보(페이지 여백, 테두리 등)는 `PageSetup` 개체를 통해 `Worksheet` 개체(Excel) 또는 `Document` 개체(Word) 수준에서 지정됩니다.
StarSuite 에서 이러한 등록 정보는 관련 문서에 연결되는 페이지 스타일을 사용하여 지정합니다.

페이지 배경

`com.sun.star.style.PageProperties` 서비스는 페이지 배경의 다음 등록 정보를 지정합니다.

- **`BackColor (long)`** - 배경색입니다.
- **`BackGraphicURL (String)`** - 사용할 배경 그래픽의 URL 입니다.
- **`BackGraphicFilter (String)`** - 배경 그래픽을 해석하기 위한 필터의 이름입니다.
- **`BackGraphicLocation (Enum)`** - 배경 그래픽의 위치입니다 (`com.sun.star.style.GraphicLocation` 열거의 값 사용).
- **`BackTransparent (Boolean)`** - 배경을 투명하게 만듭니다.

페이지 서식

페이지 서식은 `com.sun.star.style.PageProperties` 서비스의 다음 등록 정보를 사용하여 지정합니다.

- **`IsLandscape (Boolean)`** - 가로 서식입니다.
- **`Width (long)`** - 페이지 너비입니다(단위: 0.01mm).
- **`Height (long)`** - 페이지 높이입니다(단위: 0.01mm).
- **`PrinterPaperTray (String)`** - 사용할 프린터 용지함의 이름입니다.

다음은 "Default" 페이지 스타일의 페이지 크기를 DIN A5 가로 서식(높이 14.8cm, 너비 21cm)으로 설정하는 예입니다.

```
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
```

페이지 여백, 테두리 및 그림자

com.sun.star.style.PageProperties 서비스는 페이지 여백, 테두리 및 그림자를 조정하기 위한 다음 등록 정보를 제공합니다.

- **LeftMargin (long)** - 왼쪽 페이지 여백의 너비입니다(단위: 0.01mm).
- **RightMargin (long)** - 오른쪽 페이지 여백의 너비입니다(단위: 0.01mm).
- **TopMargin (long)** - 위쪽 페이지 여백의 너비입니다(단위: 0.01mm).
- **BottomMargin (long)** - 아래쪽 페이지 여백의 너비입니다(단위: 0.01mm).
- **LeftBorder (struct)** - 왼쪽 페이지 테두리 선에 대한 지정입니다 (com.sun.star.table.BorderLine 구조).
- **RightBorder (struct)** - 오른쪽 페이지 테두리 선에 대한 지정입니다 (com.sun.star.table.BorderLine 구조).
- **TopBorder (struct)** - 위쪽 페이지 테두리 선에 대한 지정입니다 (com.sun.star.table.BorderLine 구조).
- **BottomBorder (struct)** - 아래쪽 페이지 테두리 선에 대한 지정입니다 (com.sun.star.table.BorderLine 구조).
- **LeftBorderDistance (long)** - 왼쪽 페이지 테두리와 페이지 내용 간의 간격입니다(단위: 0.01mm).
- **RightBorderDistance (long)** - 오른쪽 페이지 테두리와 페이지 내용 간의 간격입니다(단위: 0.01mm).
- **TopBorderDistance (long)** - 위쪽 페이지 테두리와 페이지 내용 간의 간격입니다(단위: 0.01mm).
- **BottomBorderDistance (long)** - 아래쪽 페이지 테두리와 페이지 내용 간의 간격입니다(단위: 0.01mm).
- **ShadowFormat (struct)** - 페이지 내용 영역의 그림자에 대한 지정입니다 (com.sun.star.table.ShadowFormat 구조).

다음은 "Default" 페이지 스타일의 왼쪽 및 오른쪽 테두리를 1cm 로 설정하는 예입니다.

```
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
```

머리글 및 바닥글

문서의 머리글과 바닥글은 페이지 등록 정보의 일부이며 `com.sun.star.style.PageProperties` 서비스를 사용하여 지정합니다. 머리글의 서식 설정을 위한 등록 정보는 다음과 같습니다.

- **HeaderIsOn (Boolean)** - 머리글을 활성화합니다.
- **HeaderLeftMargin (long)** - 머리글과 왼쪽 페이지 여백 간의 간격입니다(단위: 0.01mm).
- **HeaderRightMargin (long)** - 머리글과 오른쪽 페이지 여백 간의 간격입니다(단위: 0.01mm).
- **HeaderBodyDistance (long)** - 머리글과 문서 본문 간의 간격입니다(단위: 0.01mm).
- **HeaderHeight (long)** - 머리글의 높이입니다(단위: 0.01mm).
- **HeaderIsDynamicHeight (Boolean)** - 머리글의 높이가 내용에 맞게 자동으로 조정됩니다.
- **HeaderLeftBorder (struct)** - 머리글을 둘러싼 프레임의 왼쪽 테두리에 대한 세부 정보입니다(`com.sun.star.table.BorderLine` 구조).
- **HeaderRightBorder (struct)** - 머리글을 둘러싼 프레임의 오른쪽 테두리에 대한 세부 정보입니다(`com.sun.star.table.BorderLine` 구조).
- **HeaderTopBorder (struct)** - 머리글을 둘러싼 위쪽 테두리 선에 대한 세부 정보입니다 (`com.sun.star.table.BorderLine` 구조).
- **HeaderBottomBorder (struct)** - 머리글을 둘러싼 아래쪽 테두리 선에 대한 세부 정보입니다(`com.sun.star.table.BorderLine` 구조).
- **HeaderLeftBorderDistance (long)** - 왼쪽 테두리와 머리글 내용 간의 간격입니다(단위: 0.01mm).
- **HeaderRightBorderDistance (long)** - 오른쪽 테두리와 머리글 내용 간의 간격입니다(단위: 0.01mm).
- **HeaderTopBorderDistance (long)** - 위쪽 테두리와 머리글 내용 간의 간격입니다(단위: 0.01mm).
- **HeaderBottomBorderDistance (long)** - 아래쪽 테두리와 머리글 내용 간의 간격입니다 (단위: 0.01mm).

- **HeaderIsShared (Boolean)** - 짝수 및 홀수 페이지의 머리글이 같은 내용을 가집니다(HeaderText, HeaderTextLeft 및 HeaderTextRight 참조).
- **HeaderBackColor (long)** - 머리글의 배경색입니다.
- **HeaderBackGraphicURL (String)** - 사용할 배경 그래픽의 URL 입니다.
- **HeaderBackGraphicFilter (String)** - 머리글의 배경 그래픽을 해석하기 위한 필터의 이름입니다.
- **HeaderBackGraphicLocation (Enum)** - 머리글의 배경 그래픽의 위치입니다 (com.sun.star.style.GraphicLocation 열거의 값 사용).
- **HeaderBackTransparent (Boolean)** - 머리글의 배경을 투명하게 표시합니다.
- **HeaderShadowFormat (struct)** - 머리글의 그림자에 대한 세부 정보입니다 (com.sun.star.table.ShadowFormat 구조).

바닥글의 서식 설정을 위한 등록 정보는 다음과 같습니다.

- **FooterIsOn (Boolean)** - 바닥글을 활성화합니다.
- **FooterLeftMargin (long)** - 바닥글과 왼쪽 페이지 여백 간의 간격입니다(단위: 0.01mm).
- **FooterRightMargin (long)** - 바닥글과 오른쪽 페이지 여백 간의 간격입니다(단위: 0.01mm).
- **FooterBodyDistance (long)** - 바닥글과 문서 본문 간의 간격입니다(단위: 0.01mm).
- **FooterHeight (long)** - 바닥글의 높이입니다(단위: 0.01mm).
- **FooterIsDynamicHeight (Boolean)** - 바닥글의 높이가 내용에 맞게 자동으로 조정됩니다.
- **FooterLeftBorder (struct)** - 바닥글을 둘러싼 왼쪽 테두리 선에 대한 세부 정보입니다 (com.sun.star.table.BorderLine 구조).
- **FooterRightBorder (struct)** - 바닥글을 둘러싼 오른쪽 테두리 선에 대한 세부 정보입니다(com.sun.star.table.BorderLine 구조).
- **FooterTopBorder (struct)** - 바닥글을 둘러싼 위쪽 테두리 선에 대한 세부 정보입니다 (com.sun.star.table.BorderLine 구조).
- **FooterBottomBorder (struct)** - 바닥글을 둘러싼 아래쪽 테두리 선에 대한 세부 정보입니다(com.sun.star.table.BorderLine 구조).
- **FooterLeftBorderDistance (long)** - 왼쪽 테두리와 바닥글 내용 간의 간격입니다(단위: 0.01mm).
- **FooterRightBorderDistance (long)** - 오른쪽 테두리와 바닥글 내용 간의 간격입니다(단위: 0.01mm).
- **FooterTopBorderDistance (long)** - 위쪽 테두리와 바닥글 내용 간의 간격입니다(단위: 0.01mm).
- **FooterBottomBorderDistance (long)** - 아래쪽 테두리와 바닥글 내용 간의 간격입니다 (단위: 0.01mm).

- **FooterIsShared (Boolean)** - 짝수 및 홀수 페이지의 바닥글이 같은 내용을 가집니다 (FooterText, FooterTextLeft 및 FooterTextRight 참조).
- **FooterBackColor (long)** - 바닥글의 배경색입니다.
- **FooterBackGraphicURL (String)** - 사용할 배경 그래픽의 URL 입니다.
- **FooterBackGraphicFilter (String)** - 바닥글의 배경 그래픽을 해석하기 위한 필터의 이름입니다.
- **FooterBackGraphicLocation (Enum)** - 바닥글의 배경 그래픽의 위치입니다 (com.sun.star.style.GraphicLocation 열거의 값 사용).
- **FooterBackTransparent (Boolean)** - 바닥글의 배경을 투명하게 표시합니다.
- **FooterShadowFormat (struct)** - 바닥글의 그림자에 대한 세부 정보입니다 (com.sun.star.table.ShadowFormat 구조).

머리글 및 바닥글의 텍스트 바꾸기

스프레드시트의 머리글과 바닥글의 내용은 다음 등록 정보를 통해 액세스합니다.

- **LeftPageHeaderContent (Object)** - 짝수 페이지의 머리글 내용입니다 (com.sun.star.sheet.HeaderFooterContent 서비스).
- **RightPageHeaderContent (Object)** - 홀수 페이지의 머리글 내용입니다 (com.sun.star.sheet.HeaderFooterContent 서비스).
- **LeftPageFooterContent (Object)** - 짝수 페이지의 바닥글 내용입니다 (com.sun.star.sheet.HeaderFooterContent 서비스).
- **RightPageFooterContent (Object)** - 홀수 페이지의 바닥글 내용입니다 (com.sun.star.sheet.HeaderFooterContent 서비스).

홀수 및 짝수 페이지에 대해 머리글이나 바닥글을 구별할 필요가 없을 경우(FooterIsShared 등록 정보가 False) 홀수 페이지에서 머리글과 바닥글의 등록 정보를 설정합니다.

명명된 모든 개체는 com.sun.star.sheet.HeaderFooterContent 서비스를 지원하는 개체를 구합니다. LeftText, CenterText 및 RightText 등록 정보(정식이 아님)를 통해 이 서비스는 StarSuite Calc 의 머리글과 바닥글을 위한 세 가지 텍스트 요소를 제공합니다.

다음은 "Default" 서식 파일에서 머리글의 왼쪽 텍스트 필드에 "Just a Test" 텍스트를 기록하는 예입니다.

```
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
```

이 예에서는 마지막 줄에 주의해야 합니다. 텍스트가 바뀌면 해당 변경 사항이 적용되도록 `Text-Content` 개체를 머리글에 다시 할당해야 합니다.

머리글과 바닥글이 단일 텍스트 블록으로 구성되기 때문에 머리글과 바닥글의 텍스트를 바꾸기 위한 또 다른 메커니즘을 텍스트 문서(StarSuite Writer)에 사용할 수 있습니다. 다음 등록 정보는 com.sun.star.style.PageProperties 서비스에 지정됩니다.

- **`HeaderText (Object)`** - 머리글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).
- **`HeaderTextLeft (Object)`** - 왼쪽 페이지의 머리글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).
- **`HeaderTextRight (Object)`** - 오른쪽 페이지의 머리글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).
- **`FooterText (Object)`** - 바닥글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).
- **`FooterTextLeft (Object)`** - 왼쪽 페이지의 바닥글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).
- **`FooterTextRight (Object)`** - 오른쪽 페이지의 바닥글 내용이 포함된 텍스트 개체입니다 (`com.sun.star.text.XText` 서비스).

다음은 텍스트 문서의 "Default" 페이지 스타일에 머리글을 만들고 "Just a Test" 텍스트를 이 머리글에 추가하는 예입니다.

```
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."
```

여기에서 액세스는 `HeaderFooterContent` 개체를 통해서가 아니라 페이지 스타일의 `HeaderText` 등록 정보를 통해 직접 제공됩니다.

가운데 맞춤(스프레드시트만 해당)

`com.sun.star.sheet.TablePageStyle` 서비스는 StarSuite Calc 페이지 스타일에서만 사용되며 인쇄할 셀 범위를 페이지에서 가운데로 맞출 수 있게 합니다. 이 서비스는 다음 등록 정보를 제공합니다.

- **CenterHorizontally (Boolean)** - 표의 내용이 가로로 가운데 맞춤됩니다.
- **CenterVertically (Boolean)** - 표의 내용이 세로로 가운데 맞춤됩니다.

인쇄할 요소 지정(스프레드시트만 해당)

시트의 서식을 설정할 때 페이지 요소를 볼 수 있는지 여부를 지정할 수 있습니다. 이 목적을 위해 `com.sun.star.sheet.TablePageStyle` 서비스는 다음 등록 정보를 제공합니다.

- **PrintAnnotations (Boolean)** - 셀 메모를 인쇄합니다.
- **PrintGrid (Boolean)** - 셀 눈금선을 인쇄합니다.
- **PrintHeaders (Boolean)** - 행과 열 머리글을 인쇄합니다.
- **PrintCharts (Boolean)** - 시트에 포함된 차트를 인쇄합니다.
- **PrintObjects (Boolean)** - 포함된 개체를 인쇄합니다.
- **PrintDrawing (Boolean)** - 그림 개체를 인쇄합니다.
- **PrintDownFirst (Boolean)** - 시트 내용이 여러 페이지에 걸쳐 있는 경우 세로로 위에서 아래로 인쇄된 다음 다시 오른쪽 면의 위에서 아래로 인쇄됩니다.
- **PrintFormulas (Boolean)** - 계산된 값 대신에 수식을 인쇄합니다.
- **PrintZeroValues (Boolean)** - 0 값을 인쇄합니다.

스프레드시트 문서의 효율적 편집

이전 섹션에서는 스프레드시트 문서의 기본 구조에 대해 설명하였다면 이 섹션에서는 개별 셀이나 셀 범위에 쉽게 액세스할 수 있는 서비스에 대해 설명합니다.

셀 범위

개별 셀에 대한 개체(`com.sun.star.table.Cell` 서비스) 외에도 StarSuite 는 셀 범위를 나타내는 `CellRange` 개체를 제공합니다. 이 개체는 다음과 같이 스프레드시트 개체의 `getCellRangeByName` 호출을 사용하여 만듭니다.

```
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")
```

스프레드시트 문서의 셀 범위를 지정하기 위해 콜론(:)이 사용됩니다. 예를 들어, A1:C15 는 행 1 에서 행 15 까지와 열 A, B 및 C 의 모든 셀을 나타냅니다.

셀 범위에 있는 개별 셀의 위치는 `getCellByPosition` 메소드를 사용하여 결정할 수 있으며 셀 범위에서 왼쪽 위 셀의 좌표는 (0, 0)입니다. 다음은 이 메소드를 사용하여 셀 C3 의 개체를 만드는 예입니다.

```
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)
```

셀 범위 서식 설정

개별 셀과 마찬가지로 `com.sun.star.table.CellProperties` 서비스를 사용하여 셀 범위에 서식을 적용할 수 있습니다. 이 서비스에 대한 자세한 내용과 예는 서식 설정 섹션을 참조하십시오.

셀 범위를 사용하여 계산

`computeFunction` 메소드를 사용하여 셀 범위에서 수학 연산을 수행할 수 있습니다. `computeFunction` 에서는 사용할 수학 함수를 설명하는 매개 변수로 상수가 사용됩니다. 관련 상수는 `com.sun.star.sheet.GeneralFunction` 열거에 지정됩니다. 다음 값을 사용할 수 있습니다.

- **`SUM`** - 모든 숫자 값의 합계입니다.
- **`COUNT`** - 모든 값의 총 개수입니다(숫자가 아닌 값 포함).

- **COUNTNUMS** - 모든 숫자 값의 총 개수입니다.
- **AVERAGE** - 모든 숫자 값의 평균입니다.
- **MAX** - 가장 큰 숫자 값입니다.
- **MIN** - 가장 작은 숫자 값입니다.
- **PRODUCT** - 모든 숫자 값의 곱입니다.
- **STDEV** - 표준 편차입니다.
- **VAR** - 분산입니다.
- **STDEVP** - 모집단에 기초한 표준 편차입니다.
- **VARP** - 모집단에 기초한 분산입니다.

다음은 A1:C3 범위의 평균값을 계산하고 결과를 메시지 상자에 인쇄하는 예입니다.

```
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)
```

셀 내용 삭제

`clearContents` 메소드는 셀 범위에서 특정 유형의 내용을 삭제한다는 점에서 셀 내용과 셀 범위를 삭제하는 과정을 단순화합니다.

다음은 B2:C3 범위에서 모든 문자열과 직접 서식 정보를 제거하는 예입니다.

```
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` 에 지정된 플래그는 `com.sun.star.sheet.CellFlags` 상수 목록에서 가져옵니다. 이 목록은 다음 요소를 제공합니다.

- **VALUE** - 날짜나 시간으로 서식 설정되지 않는 숫자 값입니다.

- **DATETIME** - 날짜나 시간으로 서식 설정되는 숫자 값입니다.
- **STRING** - 문자열입니다.
- **ANNOTATION** - 셀에 연결된 메모입니다.
- **FORMULA** - 수식입니다.
- **HARDATTR** - direct 셀의 직접 서식입니다.
- **STYLES** - 간접 서식입니다.
- **OBJECTS** - 셀에 연결되는 그림 개체입니다.
- **EDITATTR** - 셀의 일부에만 적용되는 문자 서식입니다.

또한 상수를 함께 추가함으로써 `clearContents` 에서 호출을 사용하여 다른 정보를 삭제할 수 있습니다.

셀 내용 검색 및 바꾸기

텍스트 문서와 마찬가지로 스프레드시트 문서는 검색 및 바꾸기 기능을 제공합니다.

스프레드시트 문서에서 검색 및 바꾸기를 위한 설명자 개체는 문서 개체를 통해 직접 만들어지지 않고 `Sheets` 목록을 통해 만들어집니다. 다음은 검색 및 바꾸기 과정을 보여 주는 예입니다.

```
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
```

이 예는 문서의 첫 번째 페이지를 사용하여 `ReplaceDescriptor` 를 만든 다음 루프를 통해 모든 페이지에 이 개체를 적용합니다.

8 장

그리기 및 프레젠테이션

이 장에서는 매크로로 제어하는 그리기 작성 및 편집을 소개합니다. 첫 번째 섹션에서는 그리기를 구성하는 기본 요소를 비롯한 그리기의 구조를 설명합니다. 두 번째 섹션에서는 개체 그룹화, 회전 및 배율 조정 같은 복잡한 편집 기능을 설명합니다.

그리기 작성, 열기 및 저장에 대한 자세한 내용은 5 장, *StarSuite* 문서 작업에서 확인할 수 있습니다.

그리기 구조

StarSuite 에서는 그리기 문서의 페이지 수가 제한되지 않으며 각 페이지를 개별적으로 디자인할 수 있습니다. 또한 페이지에 추가할 수 있는 그리기 요소 수에도 제한이 없습니다.

*계층*이 존재하기 때문에 그리기 문서는 약간 복잡합니다. 기본적으로 각 그리기 문서는 *레이아웃*, *컨트롤* 및 *치수선* 계층을 포함하며 모든 그리기 요소는 레이아웃 계층에 추가됩니다. 또한 새 계층을 추가할 수도 있습니다. 그리기 계층에 대한 자세한 내용은 StarSuite Developer's Guide 를 참조하십시오.

페이지

그리기 문서의 페이지는 `DrawPages` 목록을 통해 사용할 수 있습니다. 개별 페이지를 번호나 이름을 통해 액세스할 수 있습니다. 문서에 페이지가 하나이고 이 페이지의 이름이 *Slide 1* 인 경우 다음의 예는 똑같습니다.

예 1:

```
Dim Doc As Object
Dim Page As Object

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

예 2:

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Slide 1")
```

예 1 에서 페이지는 번호(시작 번호: 0)에 의해 지정됩니다. 두 번째 예에서는 이름과 `getByName` 메소드로 페이지에 액세스합니다.

```
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
```

위 호출은 `com.sun.star.drawing.DrawPage` 서비스를 지원하는 페이지 개체를 구합니다. 이 서비스는 다음 등록 정보를 인식합니다.

- **`BorderLeft (Long)`** - 왼쪽 테두리입니다(단위: 0.01mm).
- **`BorderRight (Long)`** - 오른쪽 테두리입니다(단위: 0.01mm).
- **`BorderTop (Long)`** - 위쪽 테두리입니다(단위: 0.01mm).
- **`BorderBottom (Long)`** - 아래쪽 테두리입니다(단위: 0.01mm).
- **`Width (Long)`** - 페이지 너비입니다(단위: 0.01mm).
- **`Height (Long)`** - 페이지 높이입니다(단위: 0.01mm).
- **`Number (Short)`** - 페이지 수(시작 번호: 1)이며 읽기 전용입니다.
- **`Orientation (Enum)`** - `com.sun.star.view.PaperOrientation` 열거에 따른 페이지 방향입니다.

이러한 설정이 바뀌면 문서의 *모든* 페이지에 영향을 줍니다.

다음은 방금 연 그리기 문서의 페이지 크기를 20 × 20cm 로 설정하고 페이지 여백을 0.5cm 로 설정하는 예입니다.

```
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
```

그림 개체의 기본 등록 정보

그림 개체는 직사각형이나 원 등의 도형, 선 및 텍스트 개체를 포함합니다. 이러한 모든 개체는 공통된 여러 기능을 공유하며 `com.sun.star.drawing.Shape` 서비스를 지원합니다. 이 서비스는 그림 개체의 `Size` 및 `Position` 등록 정보를 지정합니다.

또한 StarSuite Basic 은 파일의 서식을 설정하거나 파일을 적용할 때 이러한 등록 정보를 수정할 수 있는 여러 다른 서비스를 제공합니다. 사용할 수 있는 서식 옵션은 그림 개체의 유형에 따라 달라집니다.

다음은 직사각형을 만들어 그림 개체에 삽입하는 예입니다.

```
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)
```

이 예는 `StarDesktop.CurrentComponent` 호출을 사용하여 열려진 문서를 확인합니다. 이 방법으로 확인된 문서 개체는 `drawPages(0)` 호출을 통해 그리기의 첫 번째 페이지를 구합니다.

그런 다음, 원점(왼쪽 모서리)과 그림 개체의 크기를 포함한 `Point` 및 `Size` 구조가 초기화됩니다. 길이는 0.01mm 단위로 지정됩니다.

이어서 프로그램 코드는 `Doc.createInstance` 호출을 사용하여 `com.sun.star.drawing.RectangleShape` 서비스에 지정된 대로 직사각형 그림 개체를 만듭니다. 마지막으로 `Page.add` 호출을 사용하여 페이지에 그림 개체를 할당합니다.

채우기 등록 정보

이 섹션에서는 네 개의 서비스를 설명하며 각 인스턴스에서 샘플 프로그램 코드는 여러 유형의 서식을 결합하는 직사각형 도형 요소를 사용합니다. 채우기 등록 정보는 `com.sun.star.drawing.FillProperties` 서비스에서 결합됩니다.

StarSuite 는 채우기 영역에 대한 네 가지 기본 서식 유형을 인식합니다. 가장 간단한 변형은 단일 색상 채우기입니다. 색상 그라디언트와 격자를 지정하기 위한 옵션을 사용하면 다른 색상을 적용할 수 있습니다. 네 번째 변형은 기존 그래픽을 채우기 영역에 투영하는 옵션입니다.

그림 개체의 채우기 모드는 `FillStyle` 등록 정보를 사용하여 지정합니다. 사용 가능한 값은 `com.sun.star.drawing.FillStyle` 에 지정됩니다.

단일 색상 채우기

단일 색상 채우기의 기본 등록 정보는 다음과 같습니다.

- **`FillColor (Long)`** - 영역의 채우기 색상입니다.

채우기 모드를 사용하려면 `FillStyle` 등록 정보를 `SOLID` 채우기 모드로 설정해야 합니다.

다음은 직사각형 도형을 만들어 적색(RGB 값 255, 0, 0)으로 채우는 예입니다.

```
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)
```

색상 그라디언트

`FillStyle` 등록 정보를 `GRADIENT`로 설정할 경우 StarSuite 문서의 모든 채우기 영역에 색상 그라디언트를 적용할 수 있습니다.

미리 지정된 색상 그라디언트를 적용하려면 단순히 `FillTransparenceGradientName` 등록 정보의 관련 이름을 할당합니다. 고유한 색상 그라디언트를 지정하려면 `com.sun.star.awt.Gradient` 구조를 완성하여 `FillGradient` 등록 정보를 할당해야 합니다. 이 등록 정보는 다음 옵션을 제공합니다.

- **`Style (Enum)`** - 선형 또는 방사형 같은 그라디언트의 유형입니다. 기본값은 `com.sun.star.awt.GradientStyle`에 따라 지정됩니다.
- **StartColor (Long)** - 색상 그라디언트의 시작 색상입니다.
- **EndColor (Long)** - 색상 그라디언트의 끝 색상입니다.
- **Angle (Short)** - 색상 그라디언트의 각도입니다(단위: 0.1 도).
- **XOffset (Short)** - 색상 그라디언트가 시작되는 X 좌표입니다(단위: 0.01mm).
- **YOffset (Short)** - 색상 그라디언트가 시작되는 Y 좌표입니다(단위: 0.01mm).
- **StartIntensity (Short)** - 백분율로 지정되는 `StartColor`의 강도입니다. StarSuite Basic 에서는 100% 이상의 값을 지정할 수 있습니다.
- **EndIntensity (Short)** - 백분율로 지정되는 `EndColor`의 강도입니다. (StarSuite Basic 에서는 100% 이상의 값을 지정할 수 있습니다.)

- **StepCount (Short)** - StarSuite 가 그라디언트에 대해 계산할 색상 단계 수입니다.

다음은 com.sun.star.awt.Gradient 구조를 통해 색상 그라디언트를 사용하는 방법을 보여 주는 예입니다.

```
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)
```

이 예는 선형 색상 그라디언트(Style = LINEAR)를 만듭니다. 이 그라디언트는 왼쪽 위 모서리에서 적색(StartColor)으로 시작하여 45 도 각도(Angle)로 오른쪽 아래 모서리를 향하여 녹색(EndColor)으로 확장됩니다. 시작 및 끝 색상의 색상 강도는 150%(StartIntensity 및 EndIntensity)이기 때문에 StartColor 및 EndColor 등록 정보에 지정된 값보다 밝은 색으로 표시됩니다. 색상 그라디언트는 100 개 단계의 개별 색상을 사용하여 표시합니다(StepCount).

격자

격자 채우기를 만들려면 FillStyle 등록 정보를 HATCH 로 설정해야 합니다. 격자를 지정하기 위한 프로그램 코드는 색상 그라디언트 코드와 매우 비슷합니다. 다시 한 번 보조 구조(이 경우에는 com.sun.star.drawing.Hatch)가 사용되어 격자의 모양을 지정합니다. 격자를 위한 구조는 다음 등록 정보를 가집니다.

- **Style (Enum)** - 격자의 유형으로 단순, 정방형 또는 대각선이 포함된 정방형 중 하나입니다. 기본값은 com.sun.star.awt.HatchStyle 에 따라 지정됩니다.
- **Color (Long)** - 선의 색상입니다.
- **Distance (Long)** - 선 사이의 간격입니다(단위: 0.01mm).
- **Angle (Short)** - 격자의 각도입니다(단위: 0.1 도).

다음은 격자 구조의 사용을 보여 주는 예입니다.

```
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)
```

이 코드는 선이 45 도(Angle) 회전하는 간단한 격자 구조(HatchStyle = SINGLE)을 만듭니다. 선 색상은 진한 회색이고(Color) 선 간격은 0.2mm 입니다(Distance).

비트맵

비트맵 투영상을 채우기로 사용하려면 FillStyle 등록 정보를 BITMAP 으로 설정해야 합니다. 비트맵을 이미 StarSuite 에서 사용할 수 있는 경우 FillBitMapName 등록 정보에 이름을 지정하고 FillBitmapMode 등록 정보에 표시 스타일(단순, 바둑판식 또는 늘이기)을 지정하면 됩니다. 기본값은 com.sun.star.drawing.BitmapMode 에 따라 지정됩니다.

외부 비트맵 파일을 사용하려면 FillBitmapURL 등록 정보에 해당 URL 을 지정합니다.

다음은 직사각형을 만들고 StarSuite 에서 사용할 수 있는 Sky 비트맵을 바둑판식으로 배열하여 직사각형의 영역을 채우는 예입니다.

```
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)
```

투명도

적용할 모든 채우기의 투명도를 조정할 수 있습니다. 그리기 요소의 투명도를 바꾸는 가장 간단한 방법은 `FillTransparence` 등록 정보를 사용하는 것입니다.

다음은 50%의 투명도를 가진 적색 직사각형을 만드는 예입니다.

```
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)
```

채우기를 투명하게 만들려면 `FillTransparence` 등록 정보를 100 으로 설정합니다.

`com.sun.star.drawing.FillProperties` 서비스는 `FillTransparence` 등록 정보 외에도 `FillTransparenceGradient` 등록 정보를 추가로 제공합니다. 이 등록 정보는 채우기 영역의 투명도를 지정하는 그라디언트를 지정하는 데 사용됩니다.

선 등록 정보

테두리 선을 가질 수 있는 모든 그림 개체는 `com.sun.star.drawing.LineStyle` 서비스를 지원합니다. 이 서비스가 제공하는 일부 등록 정보는 다음과 같습니다.

- **`LineStyle (Enum)`** - 선 유형입니다. 기본값은 `com.sun.star.drawing.LineStyle` 에 따라 지정됩니다.
- **`LineColor (Long)`** - 선 색상입니다.
- **`LineTransparence (Short)`** - 선 투명도입니다.
- **`LineWidth (Long)`** - 선 두께입니다(단위: 0.01mm).
- **`LineJoint (Enum)`** - 연결 지점으로 전환됩니다. 기본값은 `com.sun.star.drawing.LineJoint` 에 따라 지정됩니다.

다음은 5mm 두께(LineWidth)와 50% 투명도의 단색 테두리(LineStyle = SOLID)를 가진 직사각형을 만드는 예입니다. 선의 오른쪽 및 왼쪽 가장자리는 서로의 교차점까지 확장(LineJoint = MITER)되어 직각을 이룹니다.

```
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)
```

com.sun.star.drawing.LineStyle 서비스는 나열된 등록 정보 외에도 점선과 파선을 그리기 위한 옵션을 추가로 제공합니다. 자세한 내용은 StarSuite API 참조 설명서를 참조하십시오.

텍스트 등록 정보(그림 개체)

com.sun.star.style.CharacterProperties 및 com.sun.star.style.ParagraphProperties 서비스는 그림 개체의 텍스트 서식을 설정할 수 있습니다. 이러한 서비스는 개별 문자 및 단락과 관련되며 6 장(텍스트 문서)에 자세하게 설명되어 있습니다.

다음은 직사각형에 텍스트를 삽입하고 com.sun.star.style.CharacterProperties 글꼴 서비스의 서식을 설정하는 예입니다.

```
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"
```

이 코드는 직사각형의 String 등록 정보를 사용하여 텍스트를 삽입하고 com.sun.star.style.CharacterProperties 서비스의 CharWeight 및 CharFontName 등록 정보를 사용하여 텍스트 글꼴의 서식을 설정합니다.

텍스트는 그림 개체가 그리기 페이지에 추가된 후에만 삽입할 수 있습니다. 또한 com.sun.star.drawing.Text 서비스를 사용하여 그림 개체에서 텍스트를 배치하고 서식을 설정할 수 있습니다. 이 서비스의 중요한 등록 정보는 다음과 같습니다.

- **TextAutoGrowHeight (Boolean)** - 포함된 텍스트에 맞게 그리기 요소의 높이를 조정합니다.
- **TextAutoGrowWidth (Boolean)** - 포함된 텍스트에 맞게 그리기 요소의 너비를 조정합니다.
- **TextHorizontalAdjust (Enum)** - 그리기 요소에 있는 텍스트의 가로 위치입니다. 기본값은 com.sun.star.drawing.TextHorizontalAdjust 에 따라 지정됩니다.
- **TextVerticalAdjust (Enum)** - 그리기 요소에 있는 텍스트의 세로 위치입니다. 기본값은 com.sun.star.drawing.TextVerticalAdjust 에 따라 지정됩니다.
- **TextLeftDistance (Long)** - 그리기 요소와 텍스트 간의 왼쪽 간격입니다(단위: 0.01mm).
- **TextRightDistance (Long)** - 그리기 요소와 텍스트 간의 오른쪽 간격입니다(단위: 0.01mm).
- **TextUpperDistance (Long)** - 그리기 요소와 텍스트 간의 위쪽 간격입니다(단위: 0.01mm).
- **TextLowerDistance (Long)** - 그리기 요소와 텍스트 간의 아래쪽 간격입니다(단위: 0.01mm).

다음은 명명된 등록 정보의 사용을 보여 주는 예입니다.

```
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"  ' May only take place after Page.add!

RectangleShape.TextVerticalAdjust = com.sun.star.drawing.TextVerticalAdjust.TOP
RectangleShape.TextHorizontalAdjust = com.sun.star.drawing.TextHorizontalAdjust.LEFT

RectangleShape.TextLeftDistance = 300
RectangleShape.TextRightDistance = 300
RectangleShape.TextUpperDistance = 300
RectangleShape.TextLowerDistance = 300
```

이 코드는 페이지에 그림 개체를 삽입한 다음 `TextVerticalAdjust` 및 `TextHorizontalAdjust` 등록 정보를 사용하여 그림 개체의 왼쪽 위 모서리에 텍스트를 추가합니다. 그림 개체의 텍스트 가장자리 간의 최소 거리는 3mm 로 설정됩니다.

그림자 등록 정보

`com.sun.star.drawing.ShadowProperties` 서비스를 사용하여 대부분의 그림 개체에 그림자를 추가할 수 있습니다. 이 서비스의 등록 정보는 다음과 같습니다.

- **`Shadow (Boolean)`** - 그림자를 활성화합니다.
- **`ShadowColor (Long)`** - 그림자 색상입니다.
- **`ShadowTransparence (Short)`** - 그림자의 투명도입니다.
- **`ShadowXDistance (Long)`** - 그림 개체로부터의 그림자의 세로 간격입니다(단위: 0.01mm).
- **`ShadowYDistance (Long)`** - 그림 개체로부터의 그림자의 가로 간격입니다(단위: 0.01mm).

다음의 예는 그림자를 가진 직사각형을 만드는 예입니다. 그림자는 직사각형에서 2mm 만큼 세로 및 가로로 오프셋되며 50% 투명도의 진한 회색으로 렌더링됩니다.

```
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)
```

다양한 그림 개체의 개요

직사각형 도형

직사각형 도형 개체(com.sun.star.drawing.RectangleShape)는 개체 서식 설정을 위한 다음 서비스를 지원합니다.

- **채우기 등록 정보** - com.sun.star.drawing.FillProperties
- **선 등록 정보** - com.sun.star.drawing.LineProperties
- **텍스트 등록 정보** - com.sun.star.drawing.Text (com.sun.star.style.CharacterProperties 및 com.sun.star.style.ParagraphProperties 포함)
- **그림자 등록 정보** - com.sun.star.drawing.ShadowProperties
- **CornerRadius (Long)** - 모서리를 둥글게 하기 위한 반지름입니다(단위: 0.01mm).

원과 타원

com.sun.star.drawing.EllipseShape 서비스는 원과 타원을 담당하며 다음 서비스를 지원합니다.

- **채우기 등록 정보** - com.sun.star.drawing.FillProperties
- **선 등록 정보** - com.sun.star.drawing.LineProperties

- **텍스트 등록 정보** - com.sun.star.drawing.Text (com.sun.star.style.CharacterProperties 및 com.sun.star.style.ParagraphProperties 포함)
- **그림자 등록 정보** - com.sun.star.drawing.ShadowProperties

이러한 서비스 외에도 원과 타원은 다음 등록 정보를 제공합니다.

- **CircleKind (Enum)** - 원과 타원의 유형입니다. 기본값은 com.sun.star.drawing.CircleKind 에 따라 지정됩니다.
- **CircleStartAngle (Long)** - 시작 각도이며 원이나 타원 세그먼트에만 해당합니다(단위: 0.1 도).
- **CircleEndAngle (Long)** - 끝 각도이며 원이나 타원 세그먼트에만 해당합니다(단위: 0.1 도).

CircleKind 등록 정보는 개체가 완전한 원인지, 원형 조각인지, 원의 한 섹션인지 결정합니다. 다음과 같은 값을 사용할 수 있습니다.

- **com.sun.star.drawing.CircleKind.FULL** - 완전한 원이나 타원입니다.
- **com.sun.star.drawing.CircleKind.CUT** - 원의 섹션입니다(인터페이스가 서로 간에 직접 연결된 부분 원).
- **com.sun.star.drawing.CircleKind.SECTION** - 원 조각입니다.
- **com.sun.star.drawing.CircleKind.ARC** - 각도입니다(원의 선은 포함하지 않음).

다음은 원주각이 70 도인 원형 조각을 만드는 예입니다. 이는 20 도의 시작 각도와 90 도의 끝 각도 간의 차이로부터 작성됩니다.

```
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)
```

선

StarSuite 는 선 개체를 위한 `com.sun.star.drawing.LineShape` 서비스를 제공합니다. 선 개체는 영역을 제외한 모든 일반 서식 설정 서비스를 지원합니다. `LineShape` 서비스와 관련된 모든 등록 정보는 다음과 같습니다.

- **선 등록 정보** - `com.sun.star.drawing.LineProperties`
- **텍스트 등록 정보** - `com.sun.star.drawing.Text` (`com.sun.star.style.CharacterProperties` 및 `com.sun.star.style.ParagraphProperties` 포함)
- **그림자 등록 정보** - `com.sun.star.drawing.ShadowProperties`

다음은 명명된 등록 정보를 사용하여 선을 만들고 그 서식을 설정하는 예입니다. 선의 원점은 `Location` 등록 정보에 지정되며 선의 끝점은 `Size` 등록 정보에 나열된 좌표에 의해 지정됩니다.

```
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)
```

Polypolygon 도형

StarSuite 는 또한 com.sun.star.drawing.PolyPolygonShape 서비스를 통해 복잡한 다각형 도형을 지원합니다. 엄격히 말해서 *PolyPolygon* 은 단순한 다각형이 아니라 다중 다각형입니다. 따라서 모서리점을 포함하는 여러 독립적 목록을 지정 및 결합하여 완전한 개체를 이룰 수 있습니다.

직사각형 도형과 마찬가지로 다음과 같은 그림 개체의 모든 서식 설정 등록 정보가 polypolygon 에 제공됩니다.

- **채우기 등록 정보** - com.sun.star.drawing.FillProperties
- **선 등록 정보** - com.sun.star.drawing.LineProperties
- **텍스트 등록 정보** - com.sun.star.drawing.Text (com.sun.star.style.CharacterProperties 및 com.sun.star.style.ParagraphProperties 포함)
- **그림자 등록 정보** - com.sun.star.drawing.ShadowProperties

또한 PolyPolygonShape 서비스에는 다각형의 좌표를 지정할 수 있는 다음 등록 정보가 있습니다.

- PolyPolygon (Array) - 다각형의 좌표를 포함하는 필드입니다 (com.sun.star.awt.Point 유형의 점을 가진 이중 행렬).

다음은 PolyPolygonShape 서비스를 사용하여 삼각형을 지정하는 방법을 보여 주는 예입니다.

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Coordinates(2) As New com.sun.star.awt.Point

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

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")
Page.add(PolyPolygonShape)   ' Page.add must take place before the coordinates are set

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())
```

다각형의 점이 절대값으로 지정되기 때문에 다각형의 크기나 시작 위치를 지정할 필요가 없습니다. 대신, 각 점에 대한 행렬을 만들어 Array(Coordinates()) 호출을 통해 두 번째 행렬로 패키지화한 다음 이 행렬을 다각형에 할당해야 합니다. 해당 호출을 실행할 수 있으려면 먼저 다각형을 문서에 삽입해야 합니다.

정의에 포함되는 이중 행렬을 사용하면 여러 다각형을 병합하여 복잡한 도형을 만들 수 있습니다. 예를 들어, 직사각형을 만든 다음 다른 직사각형을 삽입해 원래 직사각형에 구멍을 만들 수 있습니다.

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Square1(3) As New com.sun.star.awt.Point
Dim Square2(3) As New com.sun.star.awt.Point
Dim Square3(3) As New com.sun.star.awt.Point

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

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")

Page.add(PolyPolygonShape)   ' Page.add must take place before the coordinates are set

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())
```

어떤 영역이 채워지고 어떤 영역이 구멍이 되는지와 관련하여 StarSuite 는 간단한 규칙을 적용합니다. 즉, 외부 도형의 가장자리가 항상 polypolygon 의 외부 테두리가 됩니다. 안쪽의 다음 줄은 도형의 내부 테두리가 되며 첫 번째 구멍으로 전환된다는 것을 표시합니다. 안쪽에 또 다른 선이 있을 경우 이 선은 채워진 영역으로 전환된다는 것을 표시합니다.

그래픽

여기에서 제공되는 마지막 그리기 요소는 `com.sun.star.drawing.GraphicObjectShape` 서비스에 기초하는 그래픽 개체입니다. 다양한 등록 정보를 사용하여 모양을 조정할 수 있는 StarSuite 내의 모든 그래픽과 함께 그래픽 개체를 사용할 수 있습니다.

그래픽 개체는 다음 두 가지의 일반 서식 설정 등록 정보를 지원합니다.

- **텍스트 등록 정보** - `com.sun.star.drawing.Text` (`com.sun.star.style.CharacterProperties` 및 `com.sun.star.style.ParagraphProperties` 포함)
- **그림자 등록 정보** - `com.sun.star.drawing.ShadowProperties`

그래픽 개체가 지원하는 추가 등록 정보는 다음과 같습니다.

- **GraphicURL (String)** - 그래픽의 URL 입니다.
- **AdjustLuminance (Short)** - 백분율로 지정되는 색상의 광도입니다(음수 값 가능).
- **AdjustContrast (Short)** - 백분율로 지정되는 대비입니다(음수 값 가능).
- **AdjustRed (Short)** - 백분율로 지정되는 적색 값입니다(음수 값 가능).
- **AdjustGreen (Short)** - 백분율로 지정되는 녹색 값입니다(음수 값 가능).
- **AdjustBlue (Short)** - 백분율로 지정되는 청색 값입니다(음수 값 가능).
- **Gamma (Short)** - 그래픽의 감마 값입니다.
- **Transparency (Short)** - 백분율로 지정되는 그래픽의 투명도입니다.
- **GraphicColorMode (enum)** - `com.sun.star.drawing.ColorMode` 에 따라 기본값이 지정되는 표준, 회색, 흑백 등의 색상 모드입니다.

다음은 그래픽 개체 Dim Doc As Object 에 페이지를 삽입하는 방법을 보여 주는 예입니다.

```
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                      ' specifications, insignificant because latter
                                    coordinates are binding
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)
```

이 코드는 `test.jpg` 그래픽을 삽입하고 `Adjust` 등록 정보를 사용하여 모양을 조정합니다. 이 예에서 그래픽은 다른 색상 변환이 발생하지 않는 40%의 투명도를 가진 것으로 표시됩니다 (`GraphicColorMode` = `STANDARD`).

그림 개체 편집

개체 그룹화

여러 개별 그림 개체를 그룹화하여 큰 단일 개체로 작동하도록 만드는 것이 여러 상황에서 유리할 수 있습니다.

다음은 두 개의 그림 개체를 결합하는 예입니다.

```
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
' create square drawing element
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' create circle drawing element
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)
' combine square and circle drawing elements
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' centre combined drawing elements
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
```

이 코드는 직사각형과 원을 만들어 페이지에 삽입한 다음 `com.sun.star.drawing.ShapeCollection` 서비스를 지원하는 개체를 만들고 `Add` 메소드를 사용하여 직사각형과 원을 이 개체에 추가합니다. `Group` 메소드를 사용하여 페이지에 추가하는 `ShapeCollection` 은 개별 `Shape` 처럼 편집할 수 있는 실제 `Group` 개체를 구합니다.

그룹의 개별 개체를 서식 설정하려면 이러한 개체를 그룹에 추가하기 전에 서식 설정을 적용합니다. 그룹에 추가된 개체는 수정할 수 없습니다.

그림 개체 회전 및 기울이기

이전 섹션에서 설명한 모든 그림 개체를 `com.sun.star.drawing.RotationDescriptor` 서비스를 사용하여 회전 및 기울일 수도 있습니다.

이 서비스는 다음 등록 정보를 제공합니다.

- **`RotateAngle (Long)`** - 회전 각도입니다(단위: 0.01 도).
- **`ShearAngle (Long)`** - 기울기 각도입니다(단위: 0.01 도).

다음은 직사각형을 만들고 `RotateAngle` 등록 정보를 사용하여 30 도 회전하는 예입니다.

```
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)
```

다음은 위 예와 똑같은 직사각형을 만들지만 ShearAngle 등록 정보를 사용하여 30 도 기울이는 예입니다.

```
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)
```

검색 및 바꾸기

텍스트 문서와 마찬가지로 그리기 문서는 검색 및 바꾸기 기능을 제공합니다. 이 기능은 6 장, 텍스트 문서에 설명된 텍스트 문서에서 사용되는 기능과 비슷합니다. 그러나 그리기 문서에서 검색 및 바꾸기를 위한 설명자 개체는 문서 개체를 통해 직접 만들어지는 대신 관련된 문자 수준을 통해 만들어집니다. 다음은 그리기 내에서의 바꾸기 과정 개요를 보여 주는 예입니다.

```
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
```

이 코드는 문서의 첫 번째 DrawPage 를 사용하여 ReplaceDescriptor 를 만든 다음 루프를 통해 이 설명자를 그리기 문서의 모든 페이지에 적용합니다.

프레젠테이션

StarSuite 프레젠테이션은 그리기 문서에 기초합니다. 프레젠테이션의 각 페이지는 슬라이드입니다. 문서 개체의 `DrawPages` 목록을 통해 액세스하는 표준 그리기와 같은 방식으로 슬라이드를 액세스할 수 있습니다. 또한 프레젠테이션 문서를 담당하는 `com.sun.star.presentation.PresentationDocument` 서비스는 완전한 `com.sun.star.drawing.DrawingDocument` 서비스를 제공합니다.

프레젠테이션 사용

`Presentation` 등록 정보가 제공하는 그리기 기능 외에도 프레젠테이션 문서는 프레젠테이션의 기본 등록 정보 및 제어 메커니즘에 대한 액세스를 제공하는 프레젠테이션 개체를 가지고 있습니다. 예를 들어, 이 개체는 프레젠테이션을 시작할 수 있는 `start` 메소드를 제공합니다.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

이 예에 사용된 코드는 현재 프레젠테이션 문서를 참조하고 관련 프레젠테이션 개체를 설정하는 `Doc` 개체를 만듭니다. 이 개체의 `start()` 메소드가 사용되어 예를 시작하고 화면 프레젠테이션을 실행합니다.

프레젠테이션 개체로 제공되는 메소드는 다음과 같습니다.

- **`start`** - 프레젠테이션을 시작합니다.
- **`end`** - 프레젠테이션을 끝냅니다.
- **`rehearseTimings`** - 프레젠테이션을 처음부터 시작하고 런타임을 설정합니다.

또한 다음 등록 정보를 사용할 수 있습니다.

- **`AllowAnimations (Boolean)`** - 프레젠테이션의 애니메이션을 실행합니다.
- **`CustomShow (String)`** - 프레젠테이션에서 이름을 참조할 수 있도록 프레젠테이션의 이름을 지정할 수 있습니다.
- **`FirstPage (String)`** - 프레젠테이션과 함께 시작할 슬라이드의 이름입니다.
- **`IsAlwaysOnTop (Boolean)`** - 프레젠테이션 창을 항상 화면의 첫 번째 창으로 표시합니다.
- **`IsAutomatic (Boolean)`** - 프레젠테이션을 자동으로 실행합니다.
- **`IsEndless (Boolean)`** - 프레젠테이션이 끝나면 처음부터 다시 시작합니다.
- **`IsFullScreen (Boolean)`** - 프레젠테이션을 자동으로 전체 화면 모드에서 시작합니다.
- **`IsMouseVisible (Boolean)`** - 프레젠테이션 도중에 마우스를 표시합니다.
- **`Pause (long)`** - 프레젠테이션의 끝에서 빈 화면이 표시되는 시간입니다.
- **`StartWithNavigator (Boolean)`** - 프레젠테이션이 시작되면 네비게이터 창을 표시합니다.
- **`UsePn (Boolean)`** - 프레젠테이션 도중에 포인터를 표시합니다.

9 장

다이어그램(차트)

StarSuite 는 데이터 간의 그래픽 링크를 막대, 원형 차트, 선 또는 기타 요소의 형태로 만드는 다이어그램으로 데이터를 표시할 수 있습니다. 데이터는 2D 또는 3D 그래픽으로 표시될 수 있으며 그리기 요소에 사용되는 프로세스와 비슷한 방법으로 다이어그램 요소의 모양을 개별적으로 조정할 수 있습니다.

데이터를 스프레드시트의 형태로 사용할 수 있는 경우 다이어그램에 동적으로 연결할 수 있습니다. 이 경우, 기본 데이터의 모든 수정 내용을 할당된 다이어그램에서 즉시 볼 수 있습니다. 이 장에서는 StarSuite 의 다이어그램 모듈에 대한 프로그래밍 인터페이스를 대략적으로 살펴보고 스프레드시트 문서에서 다이어그램을 사용하는 방법을 설명합니다.

스프레드시트에서 다이어그램 사용

다이어그램은 StarSuite 에서 독립적인 문서가 아니라 기존 문서에 포함되는 개체로 간주됩니다.

스프레드시트 문서에서 사용될 경우 텍스트 문서 및 그리기 문서의 다이어그램이 문서 내용과 격리된 상태로 남아있지만 문서 데이터와 포함된 다이어그램 간에 링크를 설정할 수 있는 메커니즘이 제공됩니다. 다음은 스프레드시트 문서와 다이어그램 간의 상호 작용을 설명하는 예입니다.

```
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)
```

이 예에서 사용한 코드가 복잡하게 보일 수 있지만 중심 프로세스는 세 개의 줄로 제한됩니다. 첫 번째 중심 줄은 현재 스프레드시트 문서를 참조하는 Doc 문서 변수를 만듭니다(Doc 줄 = StarDesktop.CurrentComponent). 그런 다음, 이 예에서 사용한 코드는 첫 번째 스프레드시트의 모든 차트를 포함하는 목록을 만듭니다(Charts 줄 = Doc.Sheets(0).Charts). 마지막으로 addNewByName 메소드를 사용하여 이 목록의 마지막 줄에 새 차트를 추가합니다. 추가된 새 차트를 사용자가 볼 수 있습니다.

마지막 줄은 addNewByName 메소드가 또한 매개 변수로 제공하는 Rect 및 RangeAddress 보조 구조를 초기화합니다. Rect 는 스프레드시트 내의 차트 위치를 결정합니다. RangeAddress 는 데이터가 차트에 연결되는 범위를 결정합니다.

앞의 예는 막대 다이어그램을 만듭니다. 다른 유형의 그래픽이 필요할 경우 다음과 같이 막대 다이어그램을 명시적으로 대체해야 합니다.

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

첫 번째 줄은 해당 차트 개체를 지정합니다. 두 번째 줄은 현재 다이어그램을 새 다이어그램(이 예에서는 선 다이어그램)으로 대체합니다.

Excel 에서는 Excel 문서에 별도의 페이지로 삽입된 차트와 테이블 페이지에 포함된 차트가 구별되며, 이에 따라 차트에 대한 두 개의 다른 액세스 방법이 지정됩니다. StarSuite Basic 에서는 StarSuite Calc 의 차트가 항상 테이블 페이지의 포함 개체로 만들어지기 때문에 이러한 구별이 존재하지 않습니다. 차트는 항상 관련된 Sheet 개체의 Charts 목록을 사용하여 액세스합니다.

다이어그램의 구조

다이어그램의 구조와 이에 따라 지원되는 서비스 및 인터페이스 목록은 구조의 유형에 따라 결정됩니다. 예를 들어, Z 축의 메소드와 등록 정보는 2D 다이어그램이 아니라 3D 다이어그램에서만 사용할 수 있습니다. 또한 원형 차트에는 축을 사용하기 위한 인터페이스가 존재하지 않습니다.

다이어그램의 개별 요소

제목, 부제 및 키

제목, 부제 및 키는 모든 다이어그램의 기본 요소 중 일부분입니다. 다이어그램은 이러한 각 요소를 위한 고유한 개체를 제공합니다. Chart 개체는 이러한 요소를 관리할 수 있는 다음의 등록 정보를 제공합니다.

- **HasMainTitle (Boolean)** - 제목을 활성화합니다.
- **Title (Object)** - 다이어그램 제목에 대한 자세한 정보를 포함하는 개체입니다 (com.sun.star.chart.ChartTitle 서비스 지원).
- **HasSubTitle(Boolean)** - 부제를 활성화합니다.
- **Subtitle (Object)** - 다이어그램 부제에 대한 자세한 정보를 포함하는 개체입니다 (com.sun.star.chart.ChartTitle 서비스 지원).
- **HasLegend (Boolean)** - 키를 활성화합니다.

- **Legend (Object)** - 다이어그램의 키에 대한 자세한 정보를 포함하는 개체입니다 (com.sun.star.chart.ChartLegendPosition 서비스 지원).

지정된 요소는 많은 부분에서 그리기 요소와 일치합니다. 이것은 그리기 요소의 기술적 프로그램 기반을 이루는 com.sun.star.drawing.Shape 서비스를 com.sun.star.chart.ChartTitle 과 com.sun.star.chart.ChartLegendPosition 이 모두 지원하기 때문입니다.

따라서 사용자는 Size 및 Position 등록 정보를 사용하여 요소의 위치와 크기를 결정할 수 있습니다.

요소의 서식 설정을 위해 다른 채우기 및 선 등록 정보 (com.sun.star.drawing.FillProperties 및 com.sun.star.drawing.LineStyle 서비스)와 문자 등록 정보(com.sun.star.style.CharacterProperties 서비스)가 제공됩니다.

com.sun.star.chart.ChartTitle 은 명명된 서식 등록 정보뿐만 아니라 다음의 두 가지 등록 정보를 포함합니다.

- **TextRotation (Long)** - 텍스트의 회전 각도입니다(단위: 0.01 도).
- **String (String)** - 제목 또는 부제로 표시할 텍스트입니다.

다이어그램 키(com.sun.star.chart.ChartLegend 서비스)는 다음의 추가 등록 정보를 포함합니다.

- **Alignment (Enum)** - 키가 표시되는 위치입니다. 기본값은 com.sun.star.chart.ChartLegendPosition 에 따라 지정됩니다.

다음은 다이어그램을 만들어 "Test"라는 제목, "Test 2"라는 부제 및 키를 할당하는 예입니다. 키는 회색 배경색과 7 포인트의 문자 크기를 가지며 다이어그램의 아래쪽에 놓입니다.

```
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
```

배경

모든 다이어그램은 배경 영역을 가집니다. 모든 영역에는 다이어그램 개체의 다음 등록 정보를 사용하여 액세스할 수 있는 개체가 있습니다.

- **`Area (Object)`** - 다이어그램의 배경 영역입니다(`com.sun.star.chart.ChartArea` 서비스 지원).

다이어그램의 배경은 제목, 부제 및 다이어그램 키 밑의 영역을 비롯한 전체 영역을 덮습니다. 관련 `com.sun.star.chart.ChartArea` 서비스는 선 및 채우기 등록 정보만 지원하며 다양한 다른 등록 정보를 제공하지 않습니다.

다이어그램 벽 및 밑면

다이어그램 배경이 다이어그램의 전체 영역을 덮지만 다이어그램 벽은 데이터 영역의 바로 뒤쪽 영역으로 제한됩니다.

일반적으로 3D 다이어그램의 경우 두 개의 다이어그램 벽이 존재하는데, 하나는 데이터 영역의 뒤쪽에 있고 다른 하나는 Y 축의 왼쪽 경계로 사용됩니다. 또한 3D 다이어그램은 대개 밑면을 가집니다.

- **`Floor (Object)`** - 다이어그램의 밑면입니다(3D 다이어그램에만 해당, `com.sun.star.chart.ChartArea` 서비스 지원).
- **`Wall (Object)`** - 다이어그램의 벽입니다(3D 다이어그램에만 해당, `com.sun.star.chart.ChartArea` 서비스 지원).

지정된 개체는 `com.sun.star.chart.ChartArea` 서비스를 지원하며 이 서비스는 일반적인 채우기 및 선 등록 정보를 제공합니다. `com.sun.star.drawing.FillProperties` 및 `com.sun.star.drawing.LineStyle` 서비스, 8장을 참조하십시오.

다이어그램 벽과 밑면은 다음과 같이 `Chart` 개체의 일부인 `Chart` 개체를 통해 액세스합니다.

```
Chart.Area.FillBitmapName = "Sky"
```

다음은 StarSuite 에 이미 포함된 Sky 라는 그래픽을 다이어그램의 배경으로 사용하는 방법을 보여주는 예입니다.

```
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
```

축

StarSuite 는 다이어그램에 사용할 수 있는 5 개의 각기 다른 축을 인식합니다. 가장 간단한 시나리오에서는 X 및 Y 축만 존재하며, 3D 다이어그램을 사용할 경우 종종 Z 축이 제공됩니다. 다양한 데이터 행 값의 편차가 심한 다이어그램을 위해 StarSuite 는 추가 배율 작업을 위한 두 번째 X 및 Y 축을 제공합니다.

첫 번째 X, Y 및 Z 축

첫 번째 X, Y 및 Z 축은 실제 축 외에도 제목, 설명, 눈금선 및 보조 눈금선이 될 수 있습니다. 이러한 모든 요소를 표시 및 숨기는 옵션이 존재합니다. 다이어그램 개체는 이러한 기능을 관리할 수 있는 다음의 등록 정보를 제공합니다. 여기에서는 X 축을 예로 들었으나 Y 및 Z 축의 등록 정보도 같은 방법으로 구성됩니다.

- **HasXAxis (Boolean)** - X 축을 활성화합니다.
- **XAxis (Object)** - X 축에 대한 자세한 정보를 포함하는 개체입니다 (`com.sun.star.chart.ChartAxis` 서비스 지원).
- **HasXAxisDescription (Boolean)** - X 축의 설명을 활성화합니다.
- **HasXAxisGrid (Boolean)** - X 축의 주 눈금선을 활성화합니다.
- **XMainGrid (Object)** - X 축의 주 눈금선에 대한 자세한 정보를 포함하는 개체입니다(`com.sun.star.chart.ChartGrid` 서비스 지원).
- **HasXAxisHelpGrid (Boolean)** - X 축의 보조 눈금선을 활성화합니다.
- **XHelpGrid (Object)** - X 축의 보조 눈금선에 대한 자세한 정보를 포함하는 개체입니다 (`com.sun.star.chart.ChartGrid` 서비스 지원).
- **HasXAxisTitle (Boolean)** - X 축의 제목을 활성화합니다.
- **XAxisTitle (Object)** - X 축의 제목에 대한 자세한 정보를 포함하는 개체입니다 (`com.sun.star.chart.ChartTitle` 서비스 지원).

두 번째 X 및 Y 축

다음의 등록 정보를 두 번째 X 및 Y 축에 사용할 수 있습니다. 여기에서는 두 번째 X 축을 예로 들었습니다.

- **HasSecondaryXAxis (Boolean)** - 두 번째 X 축을 활성화합니다.
- **SecondaryXAxis (Object)** - 두 번째 X 축에 대한 자세한 정보를 포함하는 개체입니다(`com.sun.star.chart.ChartAxis` 서비스 지원).
- **HasSecondaryXAxisDescription (Boolean)** - X 축의 설명을 활성화합니다.

축 등록 정보

StarSuite 다이어그램의 축 개체는 `com.sun.star.chart.ChartAxis` 서비스를 지원합니다. 문자(`com.sun.star.style.CharacterProperties` 서비스, 6장 참조) 및 선

(com.sun.star.drawing.LineStyle 서비스, 8장 참조) 등록 정보 외에도 이 개체는 다음의 등록 정보를 제공합니다.

- **Max (Double)** - 축의 최대값입니다.
- **Min (Double)** - 축의 최소값입니다.
- **Origin (Double)** - 교차하는 축의 교차점입니다.
- **StepMain (Double)** - 축의 두 기본 선 사이의 간격입니다.
- **StepHelp (Double)** - 축의 두 보조 선 사이의 간격입니다.
- **AutoMax (Boolean)** - 축의 최대값을 자동으로 결정합니다.
- **AutoMin (Boolean)** - 축의 최소값을 자동으로 결정합니다.
- **AutoOrigin (Boolean)** - 교차하는 축의 교차점을 자동으로 결정합니다.
- **AutoStepMain (Boolean)** - 축의 기본 선 사이의 간격을 자동으로 결정합니다.
- **AutoStepHelp (Boolean)** - 축의 보조 선 사이의 간격을 자동으로 결정합니다.
- **Logarithmic (Boolean)** - 선형 방식이 아니라 로그 방식으로 축의 배율을 조정합니다.
- **DisplayLabels (Boolean)** - 축의 텍스트 레이블을 활성화합니다.
- **TextRotation (Long)** - 축의 텍스트 레이블의 회전 각도입니다(단위: 0.01 도).
- **Marks (Const)** - 축의 기본 선이 다이어그램 영역의 내부 또는 외부에 있어야 하는지 지정하는 상수입니다. 기본값은 com.sun.star.chart.ChartAxisMarks 에 따라 지정됩니다.
- **HelpMarks (Const)** - 축의 보조 선이 다이어그램 영역의 내부 및/또는 외부에 있어야 하는지 지정하는 상수입니다. 기본값은 com.sun.star.chart.ChartAxisMarks 에 따라 지정됩니다.
- **Overlap (Long)** - 다른 데이터 집합의 막대가 겹칠 수 있는 범위를 지정하는 백분율입니다. 100%의 경우 막대가 완전히 겹쳐서 표시되며 -100%의 경우 막대 사이에 막대 하나만큼의 간격이 생깁니다.
- **GapWidth (long)** - 차트의 다른 막대 그룹 간의 간격을 지정하는 백분율입니다. 100%의 경우 막대 하나의 너비에 해당하는 간격이 생깁니다.
- **ArrangeOrder (enum)** - 레이블의 위치에 대한 세부 정보입니다. 단일 선에서의 위치 지정 외에 두 개의 선에서 레이블을 번갈아 분할하는 옵션이 존재합니다. 기본값은 com.sun.star.chart.ChartAxisArrangeOrderType 에 따라 지정됩니다.
- **TextBreak (Boolean)** - 줄바꿈이 가능합니다.
- **TextCanOverlap (Boolean)** - 텍스트 겹침이 가능합니다.
- **NumberFormat (Long)** - 숫자 표기 형식입니다. 7장의 숫자, 날짜 및 텍스트 서식 섹션을 참조하십시오.

축 눈금선의 등록 정보

축 눈금선 개체는 `com.sun.star.chart.ChartGrid` 서비스에 기초하며 이 서비스는 `com.sun.star.drawing.LineStyle` 지원 서비스의 선 등록 정보를 지원합니다(8장 참조).

축 제목의 등록 정보

축 제목을 서식 설정하기 위한 개체는 다이어그램 제목에도 사용되는 `com.sun.star.chart.ChartTitle` 서비스에 기초합니다.

예

다음은 선 다이어그램을 만드는 예입니다. 다이어그램의 뒤쪽 벽 색상은 흰색으로 설정됩니다. X 및 Y 축은 모두 시각적으로 인식할 수 있도록 회색의 보조 눈금선을 가집니다. Y 축의 최소값과 최대값이 각각 0 과 100 으로 고정되므로 값이 바뀌더라도 다이어그램의 해상도는 유지됩니다.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

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

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")

Chart.Diagram.Wall.FillColor = RGB(255, 255, 255)

Chart.Diagram.HasXAxisGrid = True
Chart.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.HasYAxisGrid = True
Chart.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)
Chart.Diagram.YAxis.Min = 0
Chart.Diagram.YAxis.Max = 100
```

3D 다이어그램

StarSuite 에서는 대부분의 다이어그램을 3D 그래픽을 사용하여 표시할 수도 있습니다. 이 옵션을 제공하는 모든 다이어그램 유형은 `com.sun.star.chart.Dim3DDiagram`. 서비스를 지원합니다. 이 서비스는 다음의 등록 정보 하나만 제공합니다.

- **`Dim3D (Boolean)`** - 3D 표시를 활성화합니다.

누적 다이어그램

누적 다이어그램은 합계를 생성하기 위해 서로 겹쳐지면서 여러 개별 값으로 정렬되는 다이어그램입니다. 이 보기는 개별 값뿐만 아니라 모든 값의 개요를 표시합니다.

StarSuite 에서는 다양한 유형의 다이어그램을 누적형으로 표시할 수 있습니다. 이러한 다이어그램은 모두 다음의 등록 정보를 제공하는 `com.sun.star.chart.StackableDiagram` 서비스를 지원합니다.

- **`Stacked (Boolean)`** - 누적 보기 모드를 활성화합니다.
- **`Percent (Boolean)`** - 절대값이 아니라 백분율 분포를 표시합니다.

다이어그램 유형

선 다이어그램

선 다이어그램(`com.sun.star.chart.LineDiagram` 서비스)은 X 축 한 개, Y 축 두 개, Z 축 한 개를 지원합니다. 선 다이어그램은 2D 또는 3D 그래픽으로 표시될 수 있습니다 (`com.sun.star.chart.Dim3Ddiagram` 서비스). 선은 누적될 수 있습니다 (`com.sun.star.chart.StackableDiagram`).

선 다이어그램은 다음의 등록 정보를 제공합니다.

- **`SymbolType (const)`** - 데이터 포인트를 표시하기 위한 기호입니다 (`com.sun.star.chart.ChartSymbolType` 에 따른 상수).
- **`SymbolSize (Long)`** - 데이터 포인트를 표시하기 위한 기호의 크기입니다(단위: 0.01mm).
- **`SymbolBitmapURL (String)`** - 데이터 포인트를 표시하기 위한 그래픽의 파일 이름입니다.
- **`Lines (Boolean)`** - 선을 사용하여 데이터 포인트를 연결합니다.
- **`SplineType (Long)`** - 선을 부드럽게 하기 위한 스플라인 기능입니다(0: 스플라인 기능 없음, 1: 큐빅 스플라인, 2: B 스플라인).
- **`SplineOrder (Long)`** - 스플라인의 다항식 가중치입니다(B 스플라인만 해당).
- **`SplineResolution (Long)`** - 스플라인 계산을 위한 지원 포인트의 수입니다.

영역 다이어그램

영역 다이어그램(`com.sun.star.chart.AreaDiagram` 서비스)은 X 축 한 개, Y 축 두 개, Z 축 한 개를 지원합니다 영역 다이어그램은 2D 또는 3D 그래픽으로 표시될 수 있습니다

(com.sun.star.chart.Dim3Ddiagram 서비스). 영역은 누적될 수 있습니다 (com.sun.star.chart.StackableDiagram).

막대 다이어그램

막대 다이어그램(com.sun.star.chart.BarDiagram 서비스)은 X 축 한 개, Y 축 두 개, Z 축 한 개를 지원합니다. 막대 다이어그램은 2D 또는 3D 그래픽으로 표시될 수 있습니다 (com.sun.star.chart.Dim3Ddiagram 서비스). 막대는 누적될 수 있습니다 (com.sun.star.chart.StackableDiagram).

막대 다이어그램은 다음의 등록 정보를 제공합니다.

- **Vertical (Boolean) -** 막대를 세로로 표시하도록 지정합니다. 그렇지 않으면 막대는 가로로 표시됩니다.
- **Deep (Boolean)** - 3D 보기 모드에서 막대를 옆으로 나란히 표시하지 않고 서로 겹치게 놓습니다.
- **StackedBarsConnected (Boolean)** - 선을 사용하여 누적 다이어그램의 관련 막대를 연결합니다(수평 차트에서만 사용 가능).
- **NumberOfLines (Long)** - 누적 다이어그램에서 막대가 아니라 선으로 표시되는 선 수입니다.

원형 다이어그램

원형 다이어그램(com.sun.star.chart.PieDiagram 서비스)은 축을 포함하지 않으며 누적될 수 없습니다. 원형 다이어그램은 2D 또는 3D 그래픽으로 표시될 수 있습니다 (com.sun.star.chart.Dim3Ddiagram 서비스).

10 장

데이터베이스 액세스

StarSuite 에는 SDBC(Star Database Connectivity)라고 하는 시스템과 무관한 통합 데이터베이스 인터페이스가 있습니다. 이 인터페이스는 가능한 많은 다양한 데이터 원본에 대한 액세스를 제공하기 위해 개발되었습니다.

이것을 가능하게 하기 위해 드라이버로 데이터 원본에 액세스합니다. 드라이버가 데이터를 가져오는 원본은 SDBC 사용자와 무관합니다. 일부 드라이버는 파일 기반 데이터베이스에 액세스하며 이러한 데이터베이스에서 데이터를 직접 가져옵니다. 다른 드라이버는 JDBC 또는 ODBC 같은 표준 인터페이스를 사용합니다. 또한 MAPI 주소록, LDAP 디렉토리 또는 StarSuite 스프레드시트를 데이터 원본으로 액세스하는 특수한 드라이버도 있습니다.

이러한 드라이버가 UNO 구성 요소에 기초하기 때문에 다른 드라이버를 개발하고 이에 따라 새 데이터 원본을 열 수 있습니다. 이에 대한 자세한 내용은 StarSuite Developer's Guide 에 나와 있습니다.

개념적 측면에서 SDBC 는 VBA 에서 사용할 수 있는 ADO 및 DAO 라이브러리와 호환됩니다. 따라서 기본 데이터베이스 백엔드에 상관 없이 데이터베이스에 대한 높은 수준의 액세스가 가능합니다.

StarSuite 7 이 출시되면서 StarSuite 의 데이터베이스 인터페이스가 향상되었습니다. 이전에는 주로 `Application` 개체의 광범위한 메소드를 사용하여 데이터베이스에 액세스했지만 StarSuite 7 의 인터페이스는 여러 개체로 분리됩니다. 데이터베이스 기능을 위한 루트 개체로 `DatabaseContext` 가 사용됩니다.

SQL: 쿼리 언어

SDBC 사용자를 위한 쿼리 언어로 SQL 언어가 제공됩니다. 다른 SQL 언어 간의 차이점을 비교하기 위해 StarSuite 의 SDBC 구성 요소는 고유한 SQL 파서를 가지고 있습니다. 이 파서는 쿼리 창을 사용하여 입력된 SQL 명령을 검사하고 대/소문자와 관련된 오류 같은 간단한 구문 오류를 수정합니다.

드라이버가 SQL 을 지원하지 않는 데이터 원본에 대한 액세스가 가능할 경우 전송된 SQL 명령을 필요한 기본 액세스로 독립적으로 변환해야 합니다.

SDBC 의 SQL 구현은 SQL ANSI 표준을 따릅니다. `INNER JOIN` 구문 같은 Microsoft 에서 지정한 확장 기능은 지원되지 않습니다. 이러한 기능은 표준 명령으로 대체해야 합니다(예: `INNER JOIN` 을 해당 `WHERE` 절로 대체).

데이터베이스 액세스의 유형

StarSuite 의 데이터베이스 인터페이스는 StarSuite Writer 및 StarSuite Calc 응용 프로그램뿐만 아니라 데이터베이스 양식에서 사용할 수 있습니다.

StarSuite Writer 에서는 SDBC 데이터 원본의 도움을 받아 표준 문자를 만든 다음 이러한 문자를 인쇄할 수 있습니다. 또한 끌어서 놓기 기능을 사용하여 데이터베이스 창에서 텍스트 문서로 데이터를 이동할 수 있습니다.

사용자가 데이터베이스 테이블을 스프레드시트로 이동하면 StarSuite 는 원본 데이터가 수정된 경우 업데이트할 수 있는 테이블 영역을 간단한 마우스 조작으로 만듭니다. 즉, 스프레드시트 데이터를 데이터베이스 테이블로 이동하고 데이터베이스 가져오기를 수행할 수 있습니다.

마지막으로 StarSuite 는 데이터베이스에 기초한 양식을 위한 메커니즘을 제공합니다. 이를 위해 사용자는 우선 표준 StarSuite Writer 또는 StarSuite Calc 양식을 만든 다음 필드를 데이터베이스에 연결합니다.

여기에서 지정하는 모든 옵션은 StarSuite 의 사용자 인터페이스에 기초합니다. 해당 기능을 사용하는 데에는 프로그래밍 지식이 필요하지 않습니다.

그러나 이 장에서는 지정된 기능에 대한 정보를 제공하는 대신 SDBC 의 프로그래밍 인터페이스에 초점을 맞춥니다. 이 인터페이스를 사용하여 자동화된 데이터베이스 쿼리를 작성할 수 있으므로 더 광범위한 응용 프로그램을 사용할 수 있습니다.

단, 다음 섹션을 충분히 이해하려면 데이터베이스 작동 방법과 SQL 쿼리 언어에 대한 기본 지식이 있어야 합니다.

데이터 원본

데이터베이스를 StarSuite 에 통합하기 위해 일반적으로 데이터 *원본*이라고 부르는 것을 만듭니다. 사용자 인터페이스의 **기타** 메뉴에는 데이터 원본을 만들기 위한 해당 옵션이 제공됩니다. 그러나 StarSuite Basic 을 사용하여 데이터 원본을 만들어 이에 대한 작업을 수행할 수도 있습니다.

`createUnoService` 함수를 사용하여 만든 데이터베이스 컨텍스트 개체는 데이터 원본에 액세스하기 위한 시작 위치가 됩니다. 이 개체는 `com.sun.star.sdb.DatabaseContext` 서비스에 기초하며 모든 데이터베이스 작업을 위한 루트 개체입니다.

다음은 데이터베이스 컨텍스트를 만들어 사용 가능한 모든 데이터 원본의 이름을 확인하는 데 사용하는 방법을 보여 주는 예입니다. 데이터 원본의 이름은 메시지 상자에 표시됩니다.

```
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
```

개별 데이터 원본은 com.sun.star.sdb.DataSource 서비스에 기초하며 getByName 메소드를 사용하여 데이터베이스 컨텍스트에서 확인할 수 있습니다.

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
```

이 예는 *Customers* 라는 데이터 원본을 위한 DataSource 개체를 만듭니다.

데이터 원본은 데이터 원본에 대한 일반 정보와 액세스 방법에 대한 정보를 제공하는 광범위한 등록 정보를 제공합니다. 이러한 등록 정보는 다음과 같습니다.

- **Name (String)** - 데이터 원본의 이름입니다.
- **URL (String)** - *jdbc: subprotocol : subname* 또는 *sdbc: subprotocol : subname* 형태의 데이터 원본의 URL 입니다.
- **Info (Array)** - 연결 매개 변수와 함께 PropertyValue 쌍을 포함하는 행렬입니다. 일반적으로 사용자 이름과 암호 이상을 포함합니다.
- **User (String)** - 사용자 이름입니다.
- **Password (String)** - 사용자 암호입니다(저장되지 않음).
- **IsPasswordRequired (Boolean)** - 필요한 암호이며 사용자로부터 대화형으로 요청됩니다.
- **IsReadOnly (Boolean)** - 데이터베이스에 대한 읽기 전용 액세스가 가능합니다.
- **NumberFormatsSupplier (Object)** - 데이터베이스에 사용할 수 있는 숫자 표기 형식을 포함하는 개체입니다. com.sun.star.util.XNumberFormatsSupplier 인터페이스 지원, 7 장의 숫자, 날짜 및 텍스트 서식 섹션을 참조하십시오.
- **TableFilter (Array)** - 표시할 테이블 이름의 목록입니다.
- **TableTypeFilter (Array)** - 표시할 테이블 유형의 목록입니다. 사용할 수 있는 값은 TABLE, VIEW 및 SYSTEM TABLE 입니다.
- **SuppressVersionColumns (Boolean)** - 버전 관리에 사용되는 열을 표시하지 않습니다.

StarSuite 의 데이터 원본은 ODBC 의 데이터 원본과 1:1 로 비교할 수 없습니다. ODBC 데이터 원본이 데이터 원본에 대한 정보만 포함하는 것과 달리 StarSuite 의 데이터 원본은 StarSuite 의 데이터베이스 창 내에서 데이터가 표시되는 방법에 대한 광범위한 정보도 포함합니다.

쿼리

미리 지정된 쿼리를 데이터 원본에 할당할 수 있습니다. StarSuite 는 쿼리의 SQL 명령을 인식하기 때문에 언제든지 이러한 명령을 사용할 수 있습니다. 쿼리는 간단한 마우스 조작만으로도 열 수 있으며 SQL 에 대한 지식이 없는 사용자에게 SQL 명령 실행을 위한 옵션을 제공하기 때문에 데이터베이스 작업을 단순화하는 데 사용됩니다.

com.sun.star.sdb.QueryDefinition 서비스를 지원하는 개체가 쿼리의 배후에 숨겨져 있습니다. 쿼리는 데이터 원본의 QueryDefinitions 메소드를 통해 액세스합니다.

다음은 메시지 박스에 설정할 수 있는 데이터 원본 쿼리의 이름을 나열하는 예입니다.

```
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
```

이 예에서 사용된 `Name` 등록 정보 외에 `com.sun.star.sdb.QueryDefinition`은 다음과 같은 광범위한 다른 등록 정보를 제공합니다.

- **`Name (String)`** - 쿼리 이름입니다.
- **`Command (String)`** - SQL 명령입니다(일반적으로 `SELECT` 명령).
- **`UpdateTableName (String)`** - 여러 테이블에 기초하는 쿼리의 경우 값 수정이 가능한 테이블의 이름입니다.
- **`UpdateCatalogName (String)`** - 업데이트 테이블 카탈로그의 이름입니다.
- **`UpdateSchemaName (String)`** - 업데이트 테이블 다이어그램의 이름입니다.

다음은 프로그램 제어 방식으로 쿼리 개체를 만들어 데이터 원본에 할당하는 방법을 보여 주는 예입니다.

```
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)
```

우선 `createUnoService` 호출을 사용하여 쿼리 개체를 만들어 초기화한 다음 `insertByName` 을 통해 `QueryDefinitions` 개체에 삽입합니다.

데이터베이스 양식과의 링크

데이터 원본에 대한 작업을 단순화하기 위해 StarSuite 는 데이터 원본을 데이터베이스 양식과 연결하기 위한 옵션을 제공합니다. 이러한 링크는 `getBookmarks()` 메소드를 통해 사용할 수 있습니다. 이 메소드는 데이터 원본의 모든 링크를 포함하는 명명된 컨테이너 (`com.sun.star.sdb.DefinitionContainer`)를 구합니다. `Name` 또는 `Index` 를 통해 책갈피에 액세스할 수 있습니다.

다음은 *MyBookmark* 책갈피의 URL 을 확인하는 예입니다.

```
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
```

데이터베이스 액세스

데이터베이스 액세스를 위해 데이터베이스 연결이 필요합니다. 데이터베이스 연결은 데이터베이스와의 직접 통신을 가능하게 하는 전송 채널입니다. 따라서 이전 섹션에서 설명한 데이터 원본과 달리 데이터베이스 연결은 프로그램을 재시작할 때마다 다시 설정해야 합니다.

StarSuite 는 데이터베이스 연결 설정을 위한 다양한 방법을 제공합니다. 다음은 기존 데이터 원본에 기초한 방법을 보여 주는 예입니다.

```
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
```

이 예에서 사용된 코드는 데이터베이스가 암호로 보호되었는지 우선적으로 검사합니다. 그런 다음, 암호로 보호되지 않은 경우 `GetConnection` 호출을 사용하여 필요한 데이터베이스 연결을 만듭니다. 명령줄에 있는 두 개의 빈 문자열은 사용자 이름과 암호를 나타냅니다.

데이터베이스가 암호로 보호된 경우 이 예는 `InteractionHandler` 를 만들고 `ConnectWithCompletion` 메소드를 사용하여 데이터베이스 연결을 엽니다. InteractionHandler 는 필요한 로그인 데이터가 사용자에게 요청되도록 합니다.

테이블 반복

StarSuite 에서는 일반적으로 `ResultSet` 개체를 통해 테이블에 액세스합니다. `ResultSet` 은 `SELECT` 명령을 사용하여 얻은 대량의 결과 내에 있는 현재의 데이터 집합을 나타내는 특정 표시자입니다.

다음은 ResultSet 을 사용하여 데이터베이스 테이블에서 값을 쿼리하는 방법을 보여 주는 예입니다.

```
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
```

데이터베이스 연결이 설정되면 이 예에서 사용된 코드는 우선 Connection.createObject 호출을 사용하여 Statement 개체를 만듭니다. 그런 다음, 이 Statement 개체는 executeQuery 호출을 사용하여 실제 ResultSet 을 구합니다. 이제 프로그램은 ResultSet 이 실제로 존재하는지 검사하고 루프를 사용하여 데이터 레코드를 통과합니다. 필요한 값(이 예에서는 CustomerNumber 필드의 값)은 getString 메소드를 사용하여 ResultSet 을 구하며 여기에서 매개 변수 1 은 호출이 첫 번째 열의 값과 관련된다는 것을 결정합니다.

SDBC 의 ResultSet 개체는 데이터베이스에 대한 반복 액세스를 제공한다는 점에서 DAO 및 ADO 의 Recordset 개체와 유사합니다.

StarSuite 7 에서는 실제로 ResultSet 개체를 통해 데이터베이스에 액세스합니다. 이 개체는 테이블의 내용이나 SQL-SELECT 명령의 결과를 반영합니다. 이전에 ResultSet 개체는 데이터 탐색을 위한 Application 개체에 상주하는 메소드를 제공합니다(예: DataNextRecord).

값 검색을 위한 유형별 메소드

이전 섹션의 예에서 볼 수 있듯이 StarSuite 는 테이블 내용에 액세스하기 위한 getString 메소드를 제공합니다. 이 메소드는 문자열 형태의 결과를 제공합니다. 다음 get 메소드를 사용할 수 있습니다.

- `getByte()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getShort()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getInt()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getLong()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getFloat()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getDouble()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getBoolean()` - 숫자, 문자 및 문자열에 대한 SQL 데이터 유형을 지원합니다.
- `getString()` - 모든 SQL 데이터 유형을 지원합니다.
- `getBytes()` - 이진값에 대한 SQL 데이터 유형을 지원합니다.
- `getDate()` - 숫자, 문자열, 날짜 및 타임스탬프에 대한 SQL 데이터 유형을 지원합니다.
- `getTime()` - 숫자, 문자열, 날짜 및 타임스탬프에 대한 SQL 데이터 유형을 지원합니다.
- `getTimestamp()` - 숫자, 문자열, 날짜 및 타임스탬프에 대한 SQL 데이터 유형을 지원합니다.
- `getCharacterStream()` - 숫자, 문자열 및 이진값에 대한 SQL 데이터 유형을 지원합니다.
- `getUnicodeStream()` - 숫자, 문자열 및 이진값에 대한 SQL 데이터 유형을 지원합니다.
- `getBinaryStream()` - 이진값입니다.
- `getObject()` - 모든 SQL 데이터 유형을 지원합니다.

모든 경우에 해당 값을 쿼리해야 하는 매개 변수로 열 수를 나열해야 합니다.

ResultSet 변형

데이터베이스 액세스는 흔히 속도가 중요한 문제가 됩니다. 따라서 StarSuite 는 `ResultSets` 를 최적화하여 액세스 속도를 제어하는 여러 방법을 제공합니다. `ResultSet` 가 더 많은 기능을 제공할수록 일반적으로 그 구현이 복잡해지며 이에 따라 속도가 저하됩니다.

"테이블 반복" 섹션에서 제공된 것과 같은 간단한 `ResultSet` 은 사용할 수 있는 최소 범위의 기능을 제공합니다. 즉, 반복을 적용하고 값을 조사하는 작업만 가능하며 값을 수정하는 것과 같이 더 광범위한 탐색 옵션은 포함되어 있지 않습니다.

ResultSet 을 만드는 데 사용되는 Statement 개체는 ResultSet 기능에 영향을 줄 수 있는 다음과 같은 등록 정보를 제공합니다.

- **ResultSetConcurrency (const)** - com.sun.star.sdbc.ResultSetConcurrency 에 따라 데이터를 수정할 수 있는지 지정합니다.
- **ResultSetType (const)** - com.sun.star.sdbc.ResultSetType 에 따라 ResultSets 의 유형을 지정합니다.

com.sun.star.sdbc.ResultSetConcurrency 에 지정되는 값은 다음과 같습니다.

- **UPDATABLE** - ResultSet 이 값 수정을 가능하도록 합니다.
- **READ_ONLY** - ResultSet 이 수정을 할 수 없도록 합니다.

com.sun.star.sdbc.ResultSetConcurrency 상수 그룹은 다음 지정을 제공합니다.

- **FORWARD_ONLY** - ResultSet 이 앞으로 탐색만 가능하도록 합니다.
- **SCROLL_INSENSITIVE** - ResultSet 이 모든 유형의 탐색을 할 수 있도록 하지만 원본 데이터에 대한 변경 사항은 인식되지 않습니다.
- **SCROLL_SENSITIVE** - ResultSet 이 모든 유형의 탐색을 가능하도록 하며 원본 데이터에 대한 변경 사항이 ResultSet 에 영향을 줍니다.

A READ_ONLY 및 SCROLL_INSENSITIVE 등록 정보를 포함하는 ResultSet 은 ADO 및 DAO 에서 Snapshot 유형의 레코드 집합에 해당합니다.

ResultSet 의 UPDATEABLE 및 SCROLL_SENSITIVE 등록 정보를 사용할 경우 ResultSet 의 기능 범위는 ADO 및 DAO 의 Dynaset 유형 Recordset 과 유사합니다.

ResultSet 탐색 메소드

ResultSet 은 SCROLL_INSENSITIVE 또는 SCROLL_SENSITIVE 유형일 경우 데이터 탐색을 위한 광범위한 메소드를 지원합니다. 중심 메소드는 다음과 같습니다.

- **next()** - 다음 데이터 레코드를 탐색합니다.
- **previous()** - 이전 데이터 레코드를 탐색합니다.
- **first()** - 첫 번째 데이터 레코드를 탐색합니다.
- **last()** - 마지막 데이터 레코드를 탐색합니다.
- **beforeFirst()** - 첫 번째 데이터 레코드의 앞부분을 탐색합니다.
- **afterLast()** - 마지막 데이터 레코드의 뒷부분을 탐색합니다.

모든 메소드는 탐색이 성공적이었는지 지정하는 불리언 매개 변수를 구합니다.

현재 커서 위치를 확인하기 위해 다음 테스트 메소드가 제공되며 이러한 메소드는 모두 불리언 값을 구합니다.

- **isBeforeFirst()** - ResultSet 이 첫 번째 데이터 레코드 앞에 있습니다.
- **isAfterLast()** - ResultSet 이 마지막 데이터 레코드 뒤에 있습니다.
- **isFirst()** - ResultSet 이 첫 번째 데이터 레코드입니다.
- **isLast()** - ResultSet 이 마지막 데이터 레코드입니다.

데이터 레코드 수정

ResultSet 을 ResultSetConcurrency = UPDATEABLE 값을 사용하여 만든 경우 해당 내용을 편집할 수 있습니다. 이것은 SQL 명령이 데이터를 데이터베이스에 다시 기록하는 것이 가능한 경우에만 원칙에 따라 적용되며 예를 들어, 연결된 열이나 축적된 값이 있는 복잡한 SQL 명령의 경우에는 적용할 수 없습니다.

ResultSet 개체는 값 수정을 위한 Update 메소드를 제공하며 이러한 메소드는 값 검색을 위한 get 메소드와 같은 방법으로 구성됩니다. 예를 들어, updateString 메소드는 문자열 작성을 가능하도록 합니다.

수정 후에는 updateRow()메소드를 사용하여 값을 데이터베이스로 전송해야 합니다. 다음 탐색 명령 전에 호출을 실행해야 하며 그렇지 않을 경우 값이 손실됩니다.

수정 도중에 오류가 발생할 경우 cancelRowUpdates()메소드를 사용하여 실행을 취소할 수 있습니다. updateRow()를 사용하여 데이터를 데이터베이스에 다시 기록하지 않은 경우에만 이 호출을 사용할 수 있습니다.

11 장

대화 상자

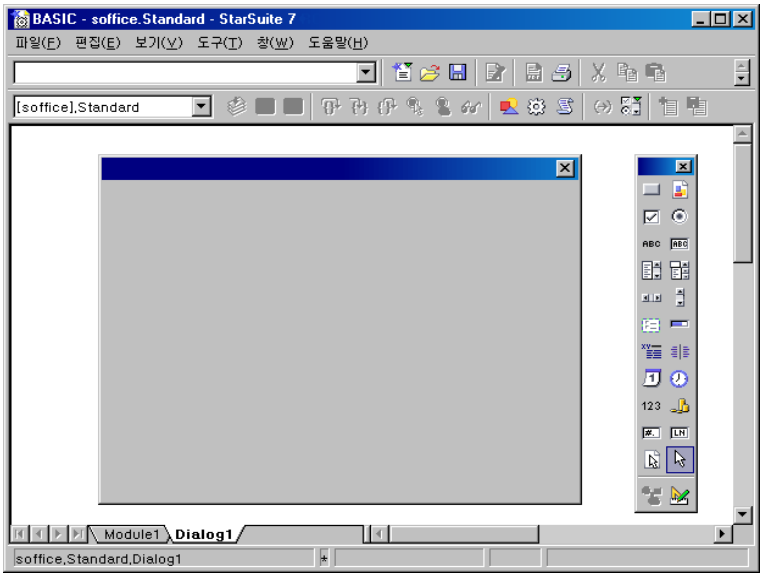
사용자 정의 대화 상자 창과 양식을 StarSuite 문서에 추가할 수 있습니다. 그런 다음, 이러한 창과 양식을 StarSuite Basic 매크로에 연결하여 StarSuite Basic 의 사용 범위를 대폭 확장할 수 있습니다. 예를 들어, 대화 상자는 데이터베이스 정보를 표시하거나 자동 파일럿 양식으로 새 문서를 만드는 단계별 과정을 안내할 수 있습니다.

대화 상자 사용

StarSuite Basic 대화 상자는 텍스트 필드, 목록 상자, 라디오 버튼 및 기타 컨트롤 요소를 포함할 수 있는 대화 상자 창으로 구성됩니다.

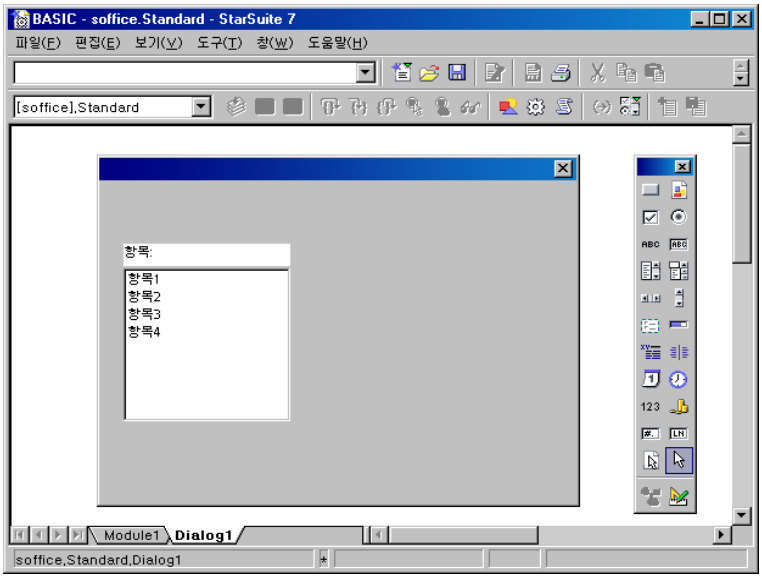
대화 상자 만들기

StarSuite Draw 와 같은 방식으로 사용할 수 있는 StarSuite 대화 상자 편집기를 통해 대화 상자를 만들고 구성할 수 있습니다.



기본적으로 원하는 컨트롤 요소를 오른쪽의 디자인 팔레트에서 대화 상자 영역으로 끕니다. 대화 상자 영역에서 컨트롤 요소의 위치와 크기를 지정할 수 있습니다.

다음은 레이블과 목록 상자를 포함하는 대화 상자를 보여 주는 예입니다.



다음 코드를 사용하여 대화 상자를 열 수 있습니다.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog`는 관련 대화 상자를 참조하는 `Dlg`라는 개체를 만듭니다. 대화 상자를 만들려면 그 전에 대화 상자가 사용하는 라이브러리(이 예에서는 `Standard` 라이브러리)가 로드되었는지 확인해야 합니다. 라이브러리가 로드되지 않은 경우 `LoadLibrary` 메소드가 이 작업을 수행합니다.

`Dlg` 대화 상자 개체가 초기화되면 `Execute` 메소드를 사용하여 대화 상자를 표시할 수 있습니다. 이러한 대화 상자는 닫기 전까지 다른 프로그램 작업을 사용할 수 없기 때문에 모달이라고 부릅니다. 이 대화 상자가 열려 있는 동안 프로그램은 `Execute` 호출 상태를 유지합니다.

코드 맨 끝의 `dispose` 메소드는 프로그램이 끝나고 대화 상자가 사용하는 리소스를 승인합니다.

대화 상자 닫기

확인 또는 취소를 사용하여 닫기

대화 상자에 **확인** 또는 **취소** 버튼이 있을 경우 이러한 버튼 중 하나를 누르면 대화 상자가 자동으로 닫힙니다. 이러한 버튼을 사용하는 것에 대한 자세한 내용은 이 장의 대화 상자 컨트롤 요소 세부 정보 섹션에 나와 있습니다.

대화 상자를 **확인** 버튼을 눌러 닫을 경우 `Execute` 메소드는 값 `1`을 구하며 그렇지 않을 경우 값 `0`을 구합니다.

```
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
```

제목 표시줄의 닫기 버튼을 사용하여 닫기

원할 경우 대화 상자 창의 제목 표시줄에 있는 닫기 버튼을 눌러 대화 상자를 닫을 수 있습니다. 이 경우 취소 버튼을 누를 때와 마찬가지로 대화 상자의 `Execute` 메소드는 값 `0`을 구합니다.

명시적 프로그램 호출을 사용하여 닫기

다음과 같이 `endExecute` 메소드를 사용하여 열린 대화 상자 창을 닫을 수도 있습니다.

```
Dlg.endExecute()
```

개별 컨트롤 요소 액세스

대화 상자는 컨트롤 요소를 얼마든지 포함할 수 있습니다. 컨트롤 요소의 이름을 구하는 `getControl` 메소드를 통해 이러한 요소에 액세스할 수 있습니다.

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

이 코드는 `MyButton` 컨트롤 요소에 대한 개체를 확인한 다음 요소에 대한 참조를 사용하여 `Ctl` 개체 변수를 초기화합니다. 마지막으로 코드는 컨트롤 요소의 `Label` 등록 정보를 `New Label` 값으로 설정합니다.

StarSuite Basic 에서는 컨트롤 요소 이름의 대/소문자를 구분한다는 점에 주의해야 합니다.

Working 대화 상자 및 컨트롤 요소의 *모델* 사용

StarSuite API 의 많은 부분에서 볼 수 있는 프로그램 요소(*보기*)와 이러한 요소 배후의 데이터 또는 문서(*모델*)가 구별됩니다. 컨트롤 요소의 메소드와 등록 정보 외에 대화 상자 및 컨트롤 요소 개체는 모두 하위 `Model` 개체를 가집니다. 이 개체를 사용하면 대화 상자나 컨트롤 요소의 내용을 직접 액세스할 수 있습니다.

대화 상자에서 데이터와 표시 간의 구별이 StarSuite 의 다른 API 영역과 같이 항상 분명한 것은 아닙니다. API 의 요소는 보기와 모델을 통해 사용할 수 있습니다.

`Model` 등록 정보는 대화 상자 및 컨트롤 요소 개체의 모델에 대한 프로그램 제어 액세스를 제공합니다.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

이는 `cmdNtext` 의 모델 개체를 통해 `Dlg` 대화 상자의 `cmdNtext` 버튼을 비활성화하는 예입니다.

등록 정보

이름 및 제목

모든 컨트롤 요소는 다음 모델 등록 정보를 사용하여 쿼리할 수 있는 고유한 이름을 가집니다.

- **`Model.Name (String)`** - 컨트롤 요소 이름입니다.

다음 모델 등록 정보를 사용하여 대화 상자의 제목 표시줄에 표시되는 제목을 지정할 수 있습니다.

- **`Model.Title (String)`** - 대화 상자 제목입니다. 단, 이는 대화 상자에만 적용됩니다.

위치 및 크기

모델 개체의 다음 등록 정보를 사용하여 컨트롤 요소의 크기와 위치를 쿼리할 수 있습니다.

- **`Model.Height (long)`** - 컨트롤 요소의 높이입니다(단위: ma).
- **`Model.Width (long)`** - 컨트롤 요소의 너비입니다(단위: ma).
- **`Model.PositionX (long)`** - 대화 상자의 왼쪽 안 가장자리부터 측정한 컨트롤 요소의 X 위치입니다(단위: ma).
- **`Model.PositionY (long)`** - 대화 상자의 위쪽 안 가장자리부터 측정한 컨트롤 요소의 Y 위치입니다(단위: ma).

대화 상자의 모양이 플랫폼에 영향을 미치지 않도록 하기 위해 StarSuite 는 *Map AppFont(ma)* 내부 단위를 사용하여 대화 상자 내의 위치와 크기를 지정합니다. ma 단위는 운영 체제에 지정되어 있는 시스템 글꼴의 평균 문자 높이 및 너비의 각각 1/8 과 1/4 로 지정됩니다. ma 단위를 사용함으로써 StarSuite 는 대화 상자가 시스템 및 시스템 설정에 상관 없이 동일하게 표시되게 합니다.

런타임에 컨트롤 요소의 크기와 위치를 변경하려는 경우 대화 상자의 총 크기를 확인하고 컨트롤 요소의 값을 해당 부분 비율로 조정합니다.

Map AppFont(ma)는 더 나은 플랫폼 독립성을 실현하기 위해 트윕 단위를 대체합니다.

초점 및 탭 시퀀스

모든 대화 상자에서 <Tab> 키를 눌러 컨트롤 요소를 탐색할 수 있습니다. 이와 관련하여 컨트롤 요소 모델에서 다음 등록 정보를 사용할 수 있습니다.

- **Model.Enabled (Boolean)** - 컨트롤 요소를 활성화합니다.
- **Model.Tabstop (Boolean)** - <Tab> 키를 통해 컨트롤 요소에 도달할 수 있게 합니다.
- **Model.TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.

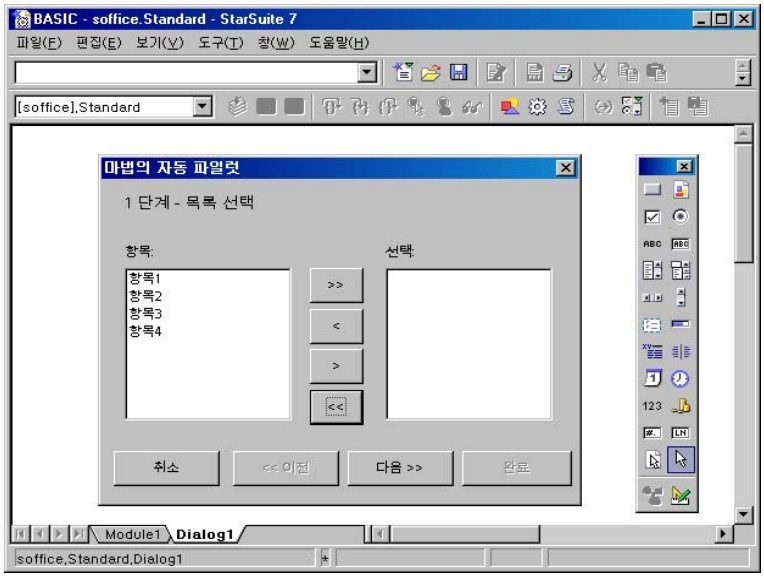
마지막으로 컨트롤 요소는 기본 컨트롤 요소가 초점을 받게 하는 getFocus 메소드를 제공합니다.

- **getFocus** - 컨트롤 요소가 초점을 받습니다. 단, 이는 대화 상자에만 해당됩니다.

여러 페이지 대화 상자

StarSuite 에서 대화 상자는 여러 탭 페이지를 가질 수 있습니다. 컨트롤 요소의 Step 등록 정보가 컨트롤 요소를 표시할 탭 페이지를 지정하는 것과 달리 대화 상자의 Step 등록 정보는 대화 상자의 현재 탭 페이지를 지정합니다.

Step 값 0 은 특수한 경우입니다. 대화 상자에서 이 값을 0 으로 설정할 경우 Step 값에 상관 없이 모든 컨트롤 요소를 볼 수 있습니다. 마찬가지로 컨트롤 요소에 대해 이 값을 0 으로 설정하면 대화 상자의 모든 탭 페이지에 해당 요소가 표시됩니다.



위 예에서 Step 값 0 을 구분선과 Cancel, Prev, Next 및 Done 버튼에 할당하여 이러한 요소를 모든 페이지에 표시할 수 있습니다. 또한 이러한 요소를 개별 탭 페이지(예: 1 페이지)에 할당할 수도 있습니다.

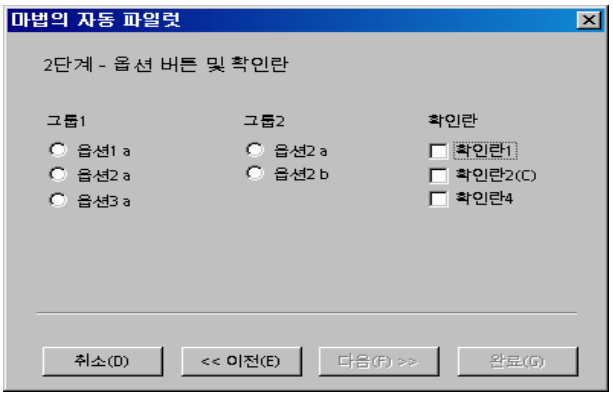
다음 프로그램 코드는 Next 및 Prev 버튼의 이벤트 처리기에 있는 Step 값을 눌리거나 줄이는 방법을 보여 주고 버튼의 상태를 변경합니다.

```
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
```

이 예가 작동하려면 열린 대화 상자를 참조하는 전역 `Dlg` 변수를 포함해야 합니다. 이제 대화 상자의 모양이 다음과 같이 바뀝니다.

1 페이지:

2 페이지:



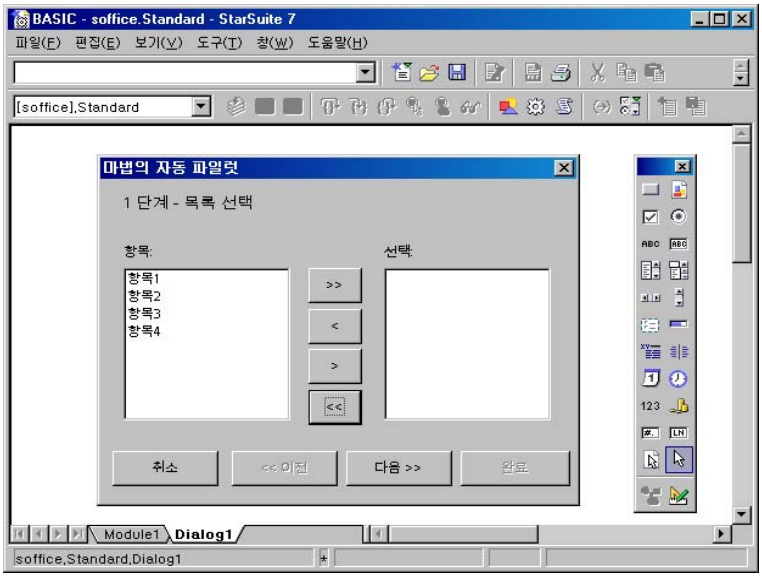
이벤트

StarSuite 대화 상자와 양식은 이벤트 처리기를 컨트롤 요소에 할당할 수 있는 이벤트 지향의 프로그래밍 모델에 기초합니다. 이벤트 처리기는 다른 이벤트인 작업을 비롯한 특정 작업이 발생했을 때 미리 지정된 절차를 실행합니다. 또한 이벤트 처리를 사용하여 문서를 편집하거나 데이터베이스를 열 수 있고 다른 컨트롤 요소에 액세스할 수 있습니다.

StarSuite 컨트롤 요소는 여러 다른 상황에서 트리거될 수 있는 다양한 유형의 이벤트를 인식합니다. 이러한 이벤트 유형은 다음 네 개의 그룹으로 구분할 수 있습니다.

- **마우스 컨트롤**: 마우스 작업(예를 들어, 간단한 마우스 이동이나 특정 화면 위치 누르기)에 해당하는 이벤트입니다.
- **키보드 컨트롤**: 키보드 입력에 의해 트리거되는 이벤트입니다.
- **초점 수정**: 컨트롤 요소가 활성화 또는 비활성화될 때 StarSuite 가 수행하는 이벤트입니다.
- **컨트롤 요소별 이벤트**: 특정 컨트롤 요소와 관련해서만 발생하는 이벤트입니다.

이벤트에 대한 작업을 수행할 때 StarSuite 개발 환경에서 관련 대화 상자를 만들었으며 이 대화 상자에 필수 컨트롤 요소나 문서가 포함되는지(이벤트를 양식에 적용할 경우) 확인합니다.



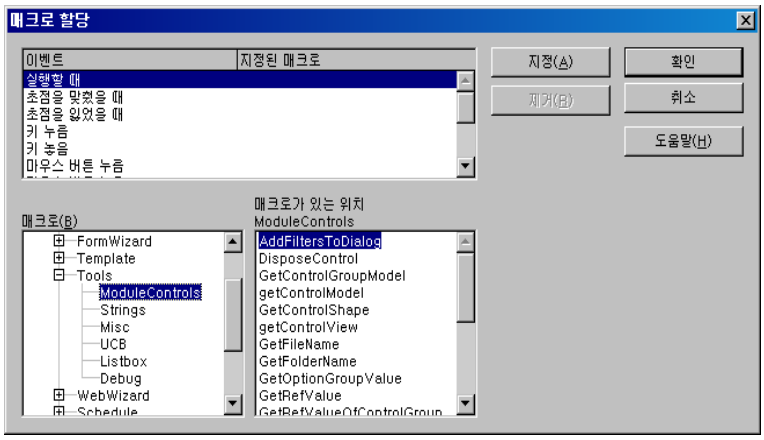
위 그림은 두 개의 목록 상자를 포함하는 대화 상자 창이 있는 StarSuite Basic 개발 환경을 보여 줍니다. 두 목록 상자 사이의 버튼을 사용하여 목록 사이에 데이터를 이동할 수 있습니다.

화면에 레이아웃을 표시하려는 경우 관련 StarSuite Basic 프로시저를 만들어 이벤트 처리기에 의해 호출될 수 있게 해야 합니다. 이러한 프로시저를 모든 모듈에서 사용할 수 있지만 두 개의 모듈로 사용을 제한하는 것이 가장 좋습니다. 코드의 가독성을 높이기 위해 이러한 프로시저에 의미 있는 이름을 할당해야 합니다. 매크로에서 일반 프로그램 프로시저로 직접 이동할 경우 코드가 불분명해질 수 있습니다. 따라서 코드 유지 관리와 문제 해결을 단순화하기 위해 이벤트 처리를 위한 진입점이 되는 다른 프로시저를 만들어야 합니다. 대상 프로시저에 대해 단일 호출만 실행될 경우에도 마찬가지입니다.

다음 예의 코드는 대화 상자의 왼쪽 목록 상자에서 오른쪽 목록 상자로 항목을 이동합니다.

```
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
```

이 프로시저가 StarSuite Basic 에서 만들어진 경우 대화 상자 편집기의 등록 정보 창을 사용하여 필요한 이벤트에 할당할 수 있습니다.



할당 대화 상자는 모든 StarSuite Basic 프로시저를 나열합니다. 프로시저를 이벤트에 할당하려면 프로시저를 선택한 다음 **지정**을 누릅니다.

매개 변수

특정 이벤트의 발생이 항상 적절한 응답을 위해 충분한 것은 아닙니다. 경우에 따라 추가 정보가 필요할 수 있습니다. 예를 들어, 마우스 누르기를 처리하기 위해 마우스 버튼이 눌러진 화면 위치가 필요할 수 있습니다.

StarSuite Basic 에서는 개체 매개 변수를 사용하여 이벤트에 대한 추가 정보를 프로시저에 제공할 수 있습니다. 예를 들면 다음과 같습니다.

```
Sub ProcessEvent(Event As Object)

End Sub
```

Event 개체가 얼마나 정확하게 구성되는지와 그 등록 정보는 프로시저 호출이 트리거하는 이벤트 유형에 따라 달라집니다. 다음 섹션에서는 이벤트 유형에 대해 자세하게 설명합니다.

이벤트 유형에 상관 없이 모든 개체는 관련 컨트롤 요소와 해당 모델에 대한 액세스를 제공합니다. 컨트롤 요소는 다음을 등록 정보를 사용하여 액세스할 수 있습니다.

```
Event.Source
```

컨트롤 요소의 모델은 다음 등록 정보를 사용하여 액세스할 수 있습니다.

```
Event.Source.Model
```

이러한 등록 정보를 사용하여 이벤트 처리기 내에서 이벤트를 트리거할 수 있습니다.

마우스 이벤트

StarSuite Basic 은 다음 마우스 이벤트를 인식합니다.

- **Mouse moved** - 사용자가 마우스를 이동합니다.
- **Mouse moved while key pressed** - 사용자가 키를 누른 상태에서 마우스를 끕니다.
- **Mouse button pressed** - 사용자가 마우스 버튼을 누릅니다.
- **Mouse button released** - 사용자가 마우스 버튼을 놓습니다.
- **Mouse outside** - 사용자가 현재 창의 바깥쪽으로 마우스를 이동합니다.

관련 이벤트 개체의 구조는 다음 정보를 제공하는 `com.sun.star.awt.MouseEvent` 구조에 지정됩니다.

- **Buttons (short)** - 버튼을 눌렀습니다(`com.sun.star.awt.MouseButton` 을 따르는 하나 이상의 상수).
- **X (long)** - 컨트롤 요소의 왼쪽 위 모서리부터 픽셀 단위로 측정된 마우스의 X 좌표입니다.
- **Y (long)** - 컨트롤 요소의 왼쪽 위 모서리부터 픽셀 단위로 측정된 마우스의 Y 좌표입니다.
- **ClickCount (long)** - 마우스 이벤트와 관련된 버튼을 누르는 수입니다. StarSuite 가 신속하게 응답할 수 있는 경우 개별 이벤트만 시작되므로 두 번 누르기에 대한 ClickCount 도 1 입니다.)

마우스 버튼의 `com.sun.star.awt.MouseButton` 에 지정되는 상수는 다음과 같습니다.

- **LEFT** - 왼쪽 마우스 버튼입니다.
- **RIGHT** - 오른쪽 마우스 버튼입니다.
- **MIDDLE** - 가운데 마우스 버튼입니다.

다음은 눌러진 마우스 버튼과 마우스 위치를 출력하는 예입니다.

```
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
```

VBA `Click` 및 `Doubleclick` 이벤트는 StarSuite Basic 에서 사용할 수 없습니다. 따라서 StarSuite Basic `MouseUp` 이벤트를 `click` 이벤트 대신에 사용하고 응용 프로그램 논리를 변경하여 `Doubleclick` 이벤트를 모방합니다.

키보드 이벤트

다음 키보드 이벤트를 StarSuite Basic 에서 사용할 수 있습니다.

- **`Key pressed`** - 사용자가 키를 누릅니다.
- **`Key released`** - 사용자가 키를 놓습니다.

두 이벤트는 모두 물리적 작업이 아니라 논리 키 작업과 관련됩니다. 사용자가 여러 키를 눌러 단일 문자를 출력할 경우, 예를 들어 액센트를 문자에 추가하기 위해서는 StarSuite Basic 는 하나의 이벤트만 만듭니다.

<Shift> 키 또는 <Alt> 키 같은 수정 키에서의 단일 키 작업은 독립적인 이벤트를 만들지 않습니다.

누른 키에 대한 정보는 이벤트 처리를 위해 StarSuite Basic 이 프로시저에 제공하는 이벤트 개체에 의해 제공됩니다. 이 개체는 다음 등록 정보를 포함합니다.

- **`KeyCode (short)`** - `com.sun.star.awt.Key` 에 따라 기본값이 지정되는 누른 키의 코드입니다.
- **`KeyChar (String)`** - 입력되는 문자입니다(수정 키 고려).

다음은 KeyCode 등록 정보를 사용하여 <Enter> 키, <Tab> 키 또는 다른 컨트롤 키 중 하나가 눌러졌는지 확인하는 예입니다. 이러한 키 중 하나가 눌러진 경우 키 이름이 구해지고, 그렇지 않은 경우 입력된 문자가 구해집니다.

```
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
```

다른 키보드 상수에 대한 정보는 com.sun.star.awt.Key 상수 그룹의 API 참조 설명서에서 확인할 수 있습니다.

초점 이벤트

초점 이벤트는 컨트롤 요소가 초점을 받는지 아니면 잃는지 나타냅니다. 이러한 이벤트를 사용하면 예를 들어, 사용자가 컨트롤 요소 처리를 완료했는지 확인하여 대화 상자의 다른 요소를 업데이트할 수 있습니다. 다음 초점 이벤트를 사용할 수 있습니다.

- **When receiving focus** - 요소가 초점을 받습니다.
- **When losing focus** - 요소가 초점을 잃습니다.

초점 이벤트의 Event 개체는 다음과 같이 구성됩니다.

- **FocusFlags (short)** - 초점이 변경되도록 하며 com.sun.star.awt.FocusChangeReason 에 따라 기본값이 지정됩니다.
- **NextFocus (Object)** - 초점을 받는 개체입니다. 이는 When losing focus 이벤트에만 해당됩니다.
- **Temporary (Boolean)** - 일시적으로 초점을 잃습니다.

컨트롤 요소별 이벤트

모든 컨트롤 요소가 지원하는 위 이벤트 외에도 특정 컨트롤 요소에 대해서만 지정되는 몇 가지 컨트롤 요소별 이벤트가 존재합니다. 이러한 이벤트 중 가장 중요한 이벤트는 다음과 같습니다.

- **When Item Changed** - 컨트롤 요소의 값이 변경됩니다.
- **Item Status Changed** - 컨트롤 요소의 상태가 변경됩니다.
- **Text modified** - 컨트롤 요소의 텍스트가 변경됩니다.
- **When initiating** - 컨트롤 요소가 트리거될 때(예: 버튼을 누른 경우) 수행할 수 있는 작업입니다.

이벤트에 대한 작업을 수행할 경우에는 일부 컨트롤 요소(예: 라디오 버튼)를 누를 때마다 `When initiating` 같은 일부 이벤트가 시작될 수 있다는 것에 주의해야 합니다. 이 경우, 컨트롤 요소의 상태가 실제로 변경되었는지 확인하기 위한 작업이 수행되지 않습니다. 이러한 "블라인드 이벤트"를 방지하려면 이전 컨트롤 요소 값을 전역 변수에 저장한 다음 이벤트가 실행될 때 값이 변경되었는지 확인합니다.

`Item Status Changed` 이벤트의 등록 정보는 다음과 같습니다.

- **Selected (long)** - 현재 선택된 항목입니다.
- **Highlighted (long)** - 현재 강조 표시된 항목입니다.
- **ItemId (long)** - 항목의 ID 입니다.

대화 상자 컨트롤 요소 세부 정보

StarSuite Basic 은 다음 그룹으로 구분할 수 있는 광범위한 컨트롤 요소를 인식합니다.

입력 필드:

- 텍스트 필드
- 날짜 필드
- 시간 필드
- 숫자 필드
- 통화 필드
- 임의의 서식을 사용하는 필드

버튼:

- 표준 버튼
- 확인란
- 옵션 버튼

선택 목록:

- 목록 상자
- 콤보 상자

기타 컨트롤 요소:

- 스크롤 막대(가로 및 세로)
- 그룹 필드
- 진행 표시줄
- 구분선(가로 및 세로)
- 그래픽
- 파일 선택 필드

이러한 컨트롤 요소 중 가장 중요한 컨트롤 요소가 아래 설명되어 있습니다.

버튼

버튼은 사용자가 눌렀을 때 작업을 수행합니다.

가장 간단한 시나리오는 사용자가 버튼을 눌렀을 때 `When Initiating` 이벤트를 트리거하는 것입니다. 또한 `PushButtonType` 등록 정보를 사용하여 대화 상자를 열기 위한 다른 작업을 버튼에 연결할 수 있습니다. 이 등록 정보가 0 으로 설정된 버튼을 누르면 대화 상자는 영향을 받지 않습니다. 이 등록 정보가 1 로 설정된 버튼을 누르면 대화 상자가 닫히고 대화 상자의 `Execute` 메소드가 값 1 을 구합니다(대화 상자 시퀀스가 올바르게 종료). `PushButtonType` 의 값이 2 일 경우에는 대화 상자가 닫히고 대화 상자의 `Execute` 메소드가 값 0 을 구합니다(대화 상자 닫힘).

버튼 모델을 통해 사용할 수 있는 전체 등록 정보는 다음과 같습니다.

- **`Model.BackgroundColor (long)`** - 배경색입니다.
- **`Model.DefaultButton (Boolean)`** - 기본값으로 사용되는 버튼이며 초점이 없을 경우 <Enter> 키에 반응합니다.
- **`Model.FontDescriptor (struct)`** - `com.sun.star.awt.FontDescriptor` 구조에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **`Model.Label (String)`** - 버튼에 표시되는 레이블입니다.
- **`Model.Printable (Boolean)`** - 컨트롤 요소를 인쇄할 수 있습니다.
- **`Model.TextColor (Long)`** - 컨트롤 요소의 텍스트 색상입니다.
- **`Model.HelpText (String)`** - 마우스 커서를 컨트롤 요소 위로 이동할 때 표시되는 도움말 텍스트입니다.
- **`Model.HelpURL (String)`** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.
- **`PushButtonType (short)`** - 버튼에 연결되는 작업입니다(0: 작업 없음, 1: 확인, 2: 취소).

옵션 버튼

옵션 버튼은 일반적으로 그룹으로 사용되며 여러 옵션 중 하나를 선택할 수 있게 합니다. 옵션 하나를 선택하면 그룹의 다른 모든 옵션은 비활성화됩니다. 따라서 한 번에 하나의 옵션 버튼만 설정됩니다.

옵션 버튼 컨트롤 요소는 다음 두 가지 등록 정보를 제공합니다.

- **State (Boolean)** - 버튼을 활성화합니다.
- **Label (String)** - 버튼에 표시되는 레이블입니다.

또한 옵션 버튼의 모델에서 다음 등록 정보를 사용할 수 있습니다.

- **Model.FontDescriptor (struct)** - `com.sun.star.awt.FontDescriptor` 에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **Model.Label (String)** - 컨트롤 요소에 표시되는 레이블입니다.
- **Model.Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **Model.State (Short)** - 이 등록 정보가 1 일 경우 옵션이 활성화되고 그렇지 않을 경우 옵션이 비활성화됩니다.
- **Model.TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **Model.HelpText (String)** - 마우스 커서를 컨트롤 요소 위에 놓을 때 표시되는 도움말 텍스트입니다.
- **Model.HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

여러 옵션 버튼을 그룹으로 결합하려면 옵션 버튼을 간격을 두지 않고 활성화 시퀀스대로 하나씩 배치해야 합니다(대화 상자 편집기에서 Order 로 표시되는 `Model.TabIndex` 등록 정보**,** 사용). 활성화 시퀀스가 다른 컨트롤 요소에 의해 중단될 경우 컨트롤 요소의 첫 번째 그룹에 상관 없이 StarSuite 는 활성화할 수 있는 새 컨트롤 요소 그룹부터 자동으로 시작합니다.

VBA 와 달리 StarSuite Basic 에서는 컨트롤 요소의 그룹에 옵션 버튼을 삽입할 수 없습니다. StarSuite Basic 에서 컨트롤 요소의 그룹화는 컨트롤 요소 주위에 프레임을 그려 시각적으로 구분하기 위한 목적으로만 사용됩니다.

확인란

확인란은 예 또는 아니오 값을 입력하는 데 사용되며 모드에 따라 둘 또는 세 개의 상태를 선택할 수 있습니다. 해당하는 예 또는 아니오 상태가 여러 의미를 가지거나 분명하지 않을 경우 확인란은 예 및 아니오 상태 외에 중간 상태를 가질 수 있습니다.

확인란은 다음 등록 정보를 제공합니다.

- **State (Short)** - 확인란의 상태입니다(0: 아니오, 1: 예, 2: 중간 상태).
- **Label (String)** - 컨트롤 요소의 레이블입니다.
- **enableTriState (Boolean)** - 활성화 및 비활성화된 상태 외에 중간 상태를 사용할 수 있습니다.

확인란의 모델 개체는 다음 등록 정보를 제공합니다.

- **Model.FontDescriptor (struct)** - com.sun.star.awt.FontDescriptor 구조에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **Model.Label (String)** - 컨트롤 요소의 레이블입니다.
- **Model.Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **Model.State (Short)** - 확인란의 상태입니다(0: 아니오, 1: 예, 2: 중간 상태).
- **Model.Tabstop (Boolean)** - <Tab> 키를 사용하여 컨트롤 요소에 도달할 수 있습니다.
- **Model.TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **Model.HelpText (String)** - 마우스 커서를 컨트롤 요소 위에 놓을 때 표시되는 도움말 텍스트입니다.
- **Model.HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

텍스트 필드

텍스트 필드를 사용하면 숫자와 텍스트를 입력할 수 있습니다. 텍스트 필드는 com.sun.star.awt.UnoControlEdit. 서비스에 기초합니다.

텍스트 필드는 하나 이상의 줄을 포함할 수 있으며 편집하거나 사용자 입력을 차단할 수 있습니다. 또한 텍스트 필드를 특정 통화 및 숫자 필드와 특정 작업을 위한 화면 필드로 사용할 수 있습니다. 이러한 컨트롤 요소가 UnoControlEdit Uno 서비스에 기초하기 때문에 프로그램에서 제어하는 처리 방식이 비슷합니다.

텍스트 필드는 다음 등록 정보를 제공합니다.

- **Text (String)** - 현재 텍스트입니다.
- **SelectedText (String)** - 현재 강조 표시된 텍스트입니다.
- **Selection (Struct)** - 세부 정보의 읽기 전용 강조 표시입니다. com.sun.star.awt.Selection, 을 따르는 구조로서 현재 강조 표시의 시작과 끝을 지정하기 위한 Min 및 Max 등록 정보가 포함되어 있습니다.
- **MaxTextLen (short)** - 필드에 입력할 수 있는 최대 문자 수입니다.
- **Editable (Boolean)** - True 는 텍스트 입력을 위한 옵션을 활성화하고 False 는 입력 옵션을 차단합니다. 이 등록 정보는 직접 호출할 수 없으며 IsEditable 을 통해서만 호출할 수 있습니다.
- **IsEditable (Boolean)** - 컨트롤 요소의 내용을 변경할 수 있으며 읽기 전용입니다.

또한 관련된 모델 개체를 통해 다음 등록 정보가 제공됩니다.

- **Model.Align (short)** - 텍스트의 방향입니다(0: 왼쪽 맞춤, 1: 가운데 맞춤, 2: 오른쪽 맞춤).
- **Model.BackgroundColor (long)** - 컨트롤 요소의 배경색입니다.
- **Model.Border (short)** - 테두리 유형입니다(0: 테두리 없음, 1: 3D 테두리, 2: 단순 테두리).
- **Model.EchoChar (String)** - 암호 필드의 에코 문자입니다.

- **Model.FontDescriptor (struct)** - com.sun.star.awt.FontDescriptor 구조에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **Model.HardLineBreaks (Boolean)** - 컨트롤 요소 텍스트에 자동 줄바꿈이 영구적으로 삽입됩니다.
- **Model.HScroll (Boolean)** -
- **Model.MaxTextLen (Short)** - 최대 텍스트 길이이며 0 일 경우 길이에 제한이 없습니다.
- **Model.MultiLine (Boolean)** - 여러 줄에 걸친 입력이 가능합니다.
- **Model.Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **Model.ReadOnly (Boolean)** - 컨트롤 요소의 내용이 읽기 전용입니다.
- **Model.Tabstop (Boolean)** - <Tab> 키를 사용하여 컨트롤 요소에 도달할 수 있습니다.
- **Model.Text (String)** - 컨트롤 요소와 연결된 텍스트입니다.
- **Model.TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **Model.VScroll (Boolean)** - 텍스트가 세로 스크롤 막대를 가집니다.
- **Model.HelpText (String)** - 마우스 커서를 컨트롤 요소 위에 놓을 때 표시되는 도움말 텍스트입니다.
- **Model.HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

목록 상자

목록 상자(com.sun.star.awt.UnoControlListBox 서비스)는 다음 등록 정보를 지원합니다.

- **ItemCount (Short)** - 요소 수이며 읽기 전용입니다.
- **SelectedItem (String)** - 강조 표시된 항목의 텍스트이며 읽기 전용입니다.
- **SelectedItems (Array Of Strings)** - 강조 표시된 항목이 있는 데이터 필드이며 읽기 전용입니다.
- **SelectedItemPos (Short)** - 현재 강조 표시된 항목의 번호이며 읽기 전용입니다.
- **SelectedItemsPos (Array of Short)** - 강조 표시된 항목 수가 포함된 데이터 필드이며 (다중 선택을 지원하는 목록의 경우) 읽기 전용입니다.
- **MultipleMode (Boolean)** - True 는 항목의 다중 선택을 위한 옵션을 활성화하며 False 는 다중 선택을 차단합니다. (이 등록 정보는 직접 호출할 수 없으며 IsMultipleMode 를 통해서만 호출할 수 있습니다.)
- **IsMultipleMode (Boolean)** - 목록 내에서 다중 선택이 가능하며 읽기 전용입니다.

목록 상자는 다음 메소드를 제공합니다.

- **addItem (Item, Pos)** - Item 에 지정된 문자열을 목록의 Pos 위치에 입력합니다.
- **addItems (ItemArray, Pos)** - 문자열의 ItemArray 데이터 필드에 나열된 항목을 Pos 위치의 목록에 입력합니다.
- **removeItems (Pos, Count)** - Pos 위치의 Count 항목을 제거합니다.
- **selectItem (Item, SelectMode)** - Item 문자열에 지정된 요소의 강조 표시를 SelectMode 불리언 변수에 따라 활성화 또는 비활성화합니다.
- **makeVisible (Pos)** - Pos 로 지정된 항목을 볼 수 있도록 목록 상자를 스크롤합니다.

목록 상자의 모델 개체는 다음 등록 정보를 제공합니다.

- **Model.BackgroundColor (long)** - 컨트롤 요소의 배경색입니다.
- **Model.Border (short)** - 테두리 유형입니다(0: 테두리 없음, 1: 3D 테두리, 2: 단순 테두리).
- **Model.FontDescriptor (struct)** - com.sun.star.awt.FontDescriptor 구조에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **Model.LineCount (Short)** - 컨트롤 요소의 줄 수입니다.
- **Model.MultiSelection (Boolean)** - 항목의 다중 선택이 가능합니다.
- **Model.SelectedItems (Array of Strings)** - 강조 표시된 항목의 목록입니다.
- **Model.StringItemList (Array of Strings)** - 모든 항목의 목록입니다.
- **Model.Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **Model.ReadOnly (Boolean)** - 컨트롤 요소의 내용이 읽기 전용입니다.
- **Model.Tabstop (Boolean)** - <Tab> 키를 사용하여 컨트롤 요소에 도달할 수 있습니다.
- **Model.TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **Model.HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.
- **Model.HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

숫자 추가 값과 함께 목록 항목을 실행하기 위한 VBA 옵션(ItemData)은 StarSuite Basic 에 존재하지 않습니다. 자연어 텍스트와 함께 숫자 값(예: 데이터베이스 ID)을 관리하려는 경우 목록 상자와 동시에 관리되는 보조 데이터 필드를 만들어야 합니다.

12 장

양식

StarSuite 양식의 구조는 대부분 이전 장에서 설명한 대화 상자와 동일하지만 다음과 같은 몇 가지 주요 차이점이 있습니다.

- 대화 상자는 문서 위에서 단일 대화 상자 창으로 표시되며 대화 상자가 끝날 때까지 대화 상자 처리 이외에 어떠한 작업도 수행할 수 없습니다. 이와 달리 양식은 그리기 요소처럼 문서에 직접 표시됩니다.
- 대화 상자 작성을 위해 대화 상자 편집기가 제공되며 StarSuite Basic 개발 환경에서 이 편집기를 사용할 수 있습니다. 양식은 문서 내에서 **양식 기능 도구 모음**을 사용하여 직접 만듭니다.
- 대화 상자 기능을 모든 StarSuite 문서에서 사용할 수 있는 것과 달리 전체 양식 기능은 텍스트와 스프레드시트에서만 사용할 수 있습니다.
- 양식의 컨트롤 요소를 외부 데이터베이스 테이블에 연결할 수 있습니다. 대화 상자에서는 이 기능을 사용할 수 없습니다.
- 대화 상자와 양식의 컨트롤 요소는 여러 차이점이 있습니다.

이벤트 처리를 위한 고유한 방법을 양식에 제공하려는 경우 11 장(대화 상자)을 참조하십시오. 이 장에서는 양식에 사용되는 것과 동일한 메커니즘을 설명합니다.

양식 사용

StarSuite 양식은 텍스트나 스프레드시트에 직접 삽입되는 텍스트 필드, 목록 상자, 라디오 버튼 및 광범위한 기타 컨트롤 요소를 포함할 수 있습니다. 양식 편집을 위해 **양식 기능 도구 모음**이 사용됩니다.

StarSuite 양식은 초안 모드와 디스플레이 모드의 두 가지 모드 중 하나를 선택할 수 있습니다. 초안 모드에서는 컨트롤 요소의 위치를 바꿀 수 있으며 등록 정보 창을 사용하여 컨트롤 요소의 등록 정보를 편집할 수 있습니다.

또한 **양식 기능 도구 모음**을 사용하여 모드 사이를 전환할 수 있습니다.

개체 양식 확인

StarSuite 는 그리기 개체 수준에서 양식의 컨트롤 요소에 대한 위치를 지정합니다. 실제 개체 양식은 그리기 수준에서 `Forms` 목록을 통해 액세스할 수 있습니다. 텍스트 문서에서 개체는 다음과 같이 액세스됩니다.

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

`GetByIndex` 메소드는 색인 번호가 0 인 양식을 반환합니다.

그리기 수준이 문서에 직접 위치하지 않고 개별 시트에 위치하기 때문에 스프레드시트를 사용하여 작업을 수행할 때 `Sheets` 목록을 통한 중간 단계가 필요합니다.

```
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)
```

`GetByIndex` 메소드 이름에서도 알 수 있듯이 문서는 여러 양식을 포함할 수 있습니다. 이것은 예를 들어, 다른 데이터베이스의 내용이 문서 하나에 표시되거나 1:n 데이터베이스 관계가 양식에 표시될 경우에 유용합니다. 또한 이러한 목적을 위해 하위 양식을 만드는 옵션이 제공됩니다.

컨트롤 요소 양식의 세 가지 측면

양식의 컨트롤 요소는 다음 세 가지 측면을 가집니다.

- 우선, 컨트롤 요소의 모델이 존재합니다. 모델은 StarSuite Basic 프로그래머가 컨트롤 요소 양식을 사용할 때의 주요 개체입니다.
- 모델에 대응하는 것으로 표시 정보를 관리하는 컨트롤 요소의 보기가 있습니다.
- 문서 내의 컨트롤 요소 양식이 특수한 그리기 요소처럼 관리되기 때문에 특히 위치와 크기에서 그리기 요소 고유의 컨트롤 요소 등록 정보를 반영하는 도형 개체도 존재합니다.

컨트롤 요소 양식의 모델 액세스

다음과 같이 `Object form`의 `GetByName` 메소드를 통해 컨트롤 요소 양식의 모델을 사용할 수 있습니다.

```
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")
```

이 예는 열려져 있는 텍스트 문서의 첫 번째 양식에 위치한 `MyListBox` 컨트롤 요소의 모델을 확인합니다.

컨트롤 요소의 양식이 확실하지 않을 경우 다음과 같이 모든 양식에서 필요한 컨트롤 요소를 검색하는 옵션을 사용할 수 있습니다.

```
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
```

이 예는 `HasByName` 메소드를 사용하여 텍스트 문서의 모든 양식을 검사함으로써 `MyListBox`라는 컨트롤 요소 모델이 포함되어 있는지 확인합니다. 해당 모델이 발견되면 모델에 대한 참조가 `Ctl` 변수에 저장되고 검색이 종료됩니다.

컨트롤 요소 양식의 보기 액세스

컨트롤 요소 양식의 보기에 액세스하려면 우선 관련 모델이 필요합니다. 그런 다음, 관련 모델과 문서 컨트롤러를 도움을 받아 컨트롤 요소의 보기를 확인할 수 있습니다.

```
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
```

이 예에 나열된 코드는 컨트롤 요소 모델을 확인하기 위한 이전 예에 나열된 코드와 매우 비슷합니다. 그러나 이 코드는 `Doc` 문서 개체뿐만 아니라 현재 문서 창에 대한 참조를 만드는 `DocCrl` 문서 컨트롤러 개체를 사용합니다. 그런 다음, 이 컨트롤러 개체 및 컨트롤 요소 모델과 함께 `GetControl` 메소드를 사용하여 컨트롤 요소 양식의 보기(`CtlView` 변수)를 확인합니다.

컨트롤 요소 양식의 도형 개체 액세스

컨트롤 요소의 도형 개체에 액세스하기 위한 방법의 경우에도 문서의 해당 그리기 수준이 사용됩니다. 특정 컨트롤 요소를 확인하려면 그리기 수준의 모든 그리기 요소를 검색해야 합니다.

```
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
```

이 예는 제어 요소 양식에 필요한 `com.sun.star.drawing.XControlShape` 인터페이스를 지원하는지 여부를 확인하기 위해 모든 그리기 요소를 검사합니다. 이 인터페이스가 지원될 경우 `Control.Name` 등록 정보는 컨트롤 요소의 이름을 검사하며 이름이 `MyListBox` 일 경우 검색이 종료됩니다.

컨트롤 요소의 크기와 위치 결정

이미 언급한 것처럼 컨트롤 요소의 크기와 위치를 관련 `shape` 개체를 사용하여 결정할 수 있습니다. 다른 모든 `shape` 개체와 같이 컨트롤 요소 도형은 이 목적을 위한 `Size` 및 `Position` 등록 정보를 제공합니다.

- **`Size (struct)`** - 컨트롤 요소의 크기입니다(`com.sun.star.awt.Size` 데이터 구조).
- **`Position (struct)`** - 컨트롤 요소의 위치입니다(`com.sun.star.awt.Point` 데이터 구조).

다음은 관련 `shape` 개체를 사용하여 컨트롤 요소의 위치와 크기를 설정하는 방법을 보여 주는 예입니다.

```
Dim Shape As Object

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

이 코드가 제대로 작동하려면 컨트롤 요소의 `shape` 개체가 이미 알려져 있어야 합니다. 그렇지 않은 경우 앞에 나온 해당 코드를 사용하여 이 개체를 확인해야 합니다.

컨트롤 요소 양식 세부 정보

양식에서 사용할 수 있는 컨트롤 요소는 대화 상자에서 사용할 수 있는 컨트롤 요소와 비슷합니다. 간단한 텍스트 필드에서 목록 및 콤보 상자와 다양한 버튼에 이르기까지 광범위하게 선택할 수 있습니다.

아래에는 컨트롤 요소 양식의 가장 중요한 등록 정보가 나열되어 있습니다. 모든 등록 정보는 관련 모델 개체의 일부입니다.

표준 컨트롤 요소 외에 완전한 데이터베이스 테이블을 통합할 수 있게 하는 테이블 컨트롤 요소를 양식에 사용할 수 있습니다. 이 컨트롤 요소는 11 장의 *데이터베이스 양식* 섹션에 설명되어 있습니다.

버튼

양식 버튼의 모델 개체는 다음 등록 정보를 제공합니다.

- **BackgroundColor (long)** - 배경색입니다.
- **DefaultButton (Boolean)** - 기본값으로 사용되는 버튼입니다. 또한 이 경우에는 초점이 없을 때 입력 버튼에도 반응합니다.
- **Enabled (Boolean)** - 컨트롤 요소를 활성화할 수 있습니다.
- **Tabstop (Boolean)** - 탭 버튼을 통해 컨트롤 요소에 도달할 수 있습니다.
- **TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **Tag (String)** - 프로그램 제어 액세스를 위해 버튼에 저장할 수 있는 추가 정보가 포함된 문자열입니다.
- **TargetURL (String)** - URL 유형의 버튼에 대한 대상 URL 입니다.
- **TargetFrame (String)** - 버튼을 활성화할 때 `TargetURL` 이 열리는 창 또는 프레임의 이름입니다(URL 유형의 버튼인 경우).
- **Label (String)** - 버튼 레이블입니다.
- **TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.
- **HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.
- **ButtonType (Enum)** - 버튼과 연결된 작업입니다(`com.sun.star.form.FormButtonType` 의 기본값 사용).

버튼을 눌렀을 때 자동으로 수행되는 작업을 ButtonType 등록 정보를 통해 지정할 수 있습니다. 관련 com.sun.star.form.FormButtonType 상수 그룹은 다음 값을 제공합니다.

- **PUSH** - 표준 버튼입니다.
- **SUBMIT** - 양식 입력을 끝냅니다(특히 HTML 양식과 관련).
- **RESET** - 영식의 모든 값을 원래 값으로 다시 설정합니다.
- **URL** - TargetURL 에 지정된 URL 을 호출합니다(TargetFrame 을 통해 지정된 창에서 열림).

대화 상자에서 제공되는 **확인** 및 **취소** 버튼 유형은 양식에서 지원되지 않습니다.

옵션 버튼

옵션 버튼의 모델 개체를 통해 다음 등록 정보를 사용할 수 있습니다.

- **Enabled (Boolean)** - 컨트롤 요소를 활성화할 수 있습니다.
- **Tabstop (Boolean)** - <Tab> 키를 통해 컨트롤 요소에 도달할 수 있습니다.
- **TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **Tag (String)** - 프로그램 제어 액세스를 위해 버튼에 저장할 수 있는 추가 정보가 포함된 문자열입니다.
- **Label (String)** - 버튼 레이블입니다.
- **Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **State (Short)** - 1 인 경우 옵션이 활성화되고 그렇지 않은 경우 옵션이 비활성화됩니다.
- **RefValue (String)** - 추가 정보(예: 데이터 레코드 ID 관리를 위한 정보)를 저장하기 위한 문자열입니다.
- **TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.
- **HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

옵션 버튼 그룹화를 위한 메커니즘은 대화 상자의 컨트롤 요소와 양식 간에 구별됩니다. 대화 상자에서 차례대로 표시되는 컨트롤 요소가 자동으로 결합되어 그룹을 구성하는 것과 달리 양식의 그룹화는 이름에 기초하여 수행됩니다. 이를 위해서 그룹의 모든 옵션 버튼은 동일한 이름을 포함해야 합니다. StarSuite 는 그룹화된 컨트롤 요소를 행렬로 결합하여 StarSuite Basic 프로그램의 개별 버튼을 이전과 동일한 방법으로 액세스할 수 있게 합니다.

다음은 컨트롤 요소 그룹의 모델을 확인하는 방법을 보여 주는 예입니다.

```
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
```

이 코드는 간단한 컨트롤 요소 모델을 확인하기 위한 이전 예와 동일합니다. 이 코드는 현재 텍스트 문서의 모든 양식을 루프로 검색하고 `HasByName` 메소드를 사용하여 검색 중인 `MyOptions` 이름을 가진 요소가 해당 양식에 포함되어 있는지 검사합니다. 이 요소가 포함된 경우 간단한 모델을 확인하기 위한 `GetByName` 메소드가 아니라 `GetGroupByName` 메소드를 사용하여 모델 행렬에 액세스합니다.

확인란

확인란 양식의 모델 개체는 다음 등록 정보를 제공합니다.

- **Enabled (Boolean)** - 컨트롤 요소를 활성화할 수 있습니다.
- **Tabstop (Boolean)** - <Tab> 키를 통해 컨트롤 요소에 도달할 수 있습니다.
- **TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **Tag (String)** - 프로그램 제어 액세스를 위해 버튼에 저장할 수 있는 추가 정보가 포함된 문자열입니다.
- **Label (String)** - 버튼 레이블입니다.
- **Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **State (Short)** - 1인 경우 옵션이 활성화되고 그렇지 않은 경우 옵션이 비활성화됩니다.
- **RefValue (String)** - 추가 정보(예: 데이터 레코드 ID 관리를 위한 정보)를 저장하기 위한 문자열입니다.
- **TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.

- **HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

텍스트 필드

텍스트 필드 양식의 모델 개체는 다음 등록 정보를 제공합니다.

- **Align (short)** - 텍스트의 방향입니다(0: 왼쪽 맞춤, 1: 가운데 맞춤, 2: 오른쪽 맞춤).
- **BackgroundColor (long)** - 컨트롤 요소의 배경색입니다.
- **Border (short)** - 테두리 유형입니다(0: 테두리 없음, 1: 3D 테두리, 2: 단순 테두리).
- **EchoChar (String)** - 암호 필드의 에코 문자입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **HardLineBreaks (Boolean)** - 컨트롤 요소 텍스트에 자동 줄바꿈이 영구적으로 삽입됩니다.
- **HScroll (Boolean)** - 텍스트가 가로 스크롤 막대를 가집니다.
- **MaxTextLen (Short)** - 최대 텍스트 길이이며 0 일 경우 길이에 제한이 없습니다.
- **MultiLine (Boolean)** - 여러 줄 입력이 가능합니다.
- **Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **ReadOnly (Boolean)** - 컨트롤 요소의 내용이 읽기 전용입니다.
- **Enabled (Boolean)** - 컨트롤 요소를 활성화할 수 있습니다.
- **Tabstop (Boolean)** - <Tab> 키를 통해 컨트롤 요소에 도달할 수 있습니다.
- **TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **Text (String)** - 컨트롤 요소의 텍스트입니다.
- **TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **VScroll (Boolean)** - 텍스트가 세로 스크롤 막대를 가집니다.
- **HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.
- **HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

목록 상자

목록 상자 양식의 모델 개체는 다음 등록 정보를 제공합니다.

- **BackgroundColor (long)** - 컨트롤 요소의 배경색입니다.
- **Border (short)** - 테두리 유형입니다(0: 테두리 없음, 1: 3D 프레임, 2: 단순 프레임).
- **FontDescriptor (struct)** - `com.sun.star.awt.FontDescriptor` 구조에 따라 사용할 글꼴의 세부 정보를 지정하는 구조입니다.
- **LineCount (Short)** - 컨트롤 요소의 줄 수입니다.
- **MultiSelection (Boolean)** - 항목의 다중 선택이 가능합니다.
- **SelectedItems (Array of Strings)** - 강조 표시된 항목의 목록입니다.
- **StringItemList (Array of Strings)** - 모든 항목의 목록입니다.
- **ValueItemList (Array of Variant)** - 각 항목에 대한 추가 정보(예: 데이터 레코드 ID 관리를 위한 정보)를 포함하는 목록입니다.
- **Printable (Boolean)** - 컨트롤 요소를 인쇄할 수 있습니다.
- **ReadOnly (Boolean)** - 컨트롤 요소의 내용이 읽기 전용입니다.
- **Enabled (Boolean)** - 컨트롤 요소를 활성화할 수 있습니다.
- **Tabstop (Boolean)** - <Tab> 키를 통해 컨트롤 요소에 도달할 수 있습니다.
- **TabIndex (Long)** - 활성화 순서에 따른 컨트롤 요소의 위치입니다.
- **FontName (String)** - 글꼴 유형의 이름입니다.
- **FontHeight (Single)** - 포인트(pt) 단위의 문자 높이입니다.
- **Tag (String)** - 프로그램 제어 액세스를 위해 버튼에 저장할 수 있는 추가 정보가 포함된 문자열입니다.
- **TextColor (Long)** - 컨트롤 요소의 텍스트 색상입니다.
- **HelpText (String)** - 마우스 커서가 컨트롤 요소 위에 있을 때 자동으로 표시되는 도움말 텍스트입니다.
- **HelpURL (String)** - 해당 컨트롤 요소에 대한 온라인 도움말의 URL 입니다.

> 목록 상자 양식은 `ValueItemList` 등록 정보를 통해 VBA 등록 정보 `ItemData` 에 대응하는 기능을 제공합니다. 이 등록 정보를 사용하여 개별 목록 항목에 대한 추가 정보를 관리할 수 있습니다.

또한 목록 상자의 보기 개체를 통해 다음 메소드가 제공됩니다.

- **addItem (Item, Pos)** - `Item` 에 지정된 문자열을 목록의 `Pos` 위치에 삽입합니다.
- **addItems (ItemArray, Pos)** - 문자열의 `ItemArray` 데이터 필드에 나열된 항목을 목록의 `Pos` 위치에 삽입합니다.
- **removeItems (Pos, Count)** - `Pos` 위치의 `Count` 항목을 제거합니다.
- **selectItem (Item, SelectMode)** - `Item` 문자열에 지정된 요소의 강조 표시를 `SelectMode` 변수에 따라 활성화 또는 비활성화합니다.
- **makeVisible (Pos)** - `Pos` 로 지정된 항목을 볼 수 있도록 목록 상자를 스크롤합니다.

데이터베이스 양식

StarSuite 양식은 데이터베이스에 직접 연결될 수 있습니다. 이 방법으로 만든 양식은 독립적인 프로그래밍 작업을 요구하지 않고 완전한 데이터베이스 프런트 엔드의 모든 기능을 제공합니다.

사용자는 선택된 테이블과 쿼리를 탐색 및 검색하는 것 외에도 데이터 레코드를 바꾸고 새 데이터 레코드를 삽입할 수 있습니다. StarSuite 에서는 관련 데이터가 데이터베이스에서 검색되고 모든 변경 사항이 데이터베이스에 다시 기록되는 작업이 자동으로 수행됩니다.

데이터베이스 양식은 기본적으로 표준 StarSuite 양식에 해당합니다. 표준 등록 정보 외에 다음 데이터베이스별 등록 정보를 양식에서 설정해야 합니다.

- **DataSourceName (String)** - 데이터 원본의 이름입니다. 10장의 데이터베이스 액세스; 데이터베이스 액세스를 참조하십시오. 데이터 원본은 StarSuite 에서 전역으로 만들어야 합니다.
- **Command (String)** - 링크가 만들어지는 SQL select 명령, 테이블 또는 쿼리의 이름입니다.
- **CommandType (Const)** - Command 가 테이블, 쿼리 또는 SQL 명령인지 지정합니다 (com.sun.star.sdb.CommandType 열거의 값 사용).

com.sun.star.sdb.CommandType 열거는 다음 값을 포함합니다.

- **TABLE** - 테이블
- **QUERY** - 쿼리
- **COMMAND** - SQL 명령

데이터베이스 필드는 다음 등록 정보를 통해 개별 컨트롤 요소에 할당됩니다.

- **DataField (String)** - 연결된 데이터베이스 필드의 이름입니다.

테이블

데이터베이스 작업을 위한 또 다른 컨트롤 요소인 테이블 컨트롤 요소가 제공됩니다. 이 컨트롤 요소는 완전한 데이터베이스 테이블이나 쿼리의 내용을 나타냅니다. 가장 간단한 시나리오의 예로 자동 파일럿 양식을 사용하여 테이블 컨트롤 요소를 데이터베이스에 연결하는 경우를 들 수 있습니다. 자동 파일럿 양식은 사용자 지정에 따라 모든 열을 관련 데이터베이스 필드와 연결합니다. 관련된 API 가 비교적 복잡하기 때문에 API 에 대한 구체적인 내용은 여기에서 다루지 않습니다.

부록

VBA 이전(Migration) 팁

StarOffice 5.x 이전(Migration) 팁

색인

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y