StarOffice™ 7 Office Suite

A Sun™ ONE Software Offering

Manual de programación en Basic

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

Ref. 817-3923-10
2003, Revisión A

Copyrights y marcas comerciales

Tabla de contenido

CAPÍTULO 1

Introducción

Este manual ofrece una introducción a la programación con StarOffice Basic 6.0 e indica las posibles aplicaciones del uso de StarOffice Basic en StarOffice. Para sacar el máximo provecho de este manual, es conveniente estar familiarizado con otros lenguajes de programación.

El libro incluye ejemplos detallados para ayudarle a desarrollar sus propios programas de StarOffice Basic con rapidez.

También se ofrecen diversos consejos de migración para programadores de Microsoft Visual Basic o para quienes hayan trabajado con versiones anteriores de StarOffice Basic. Dichos consejos están señalados mediante un pequeño símbolo en el borde de la página. El anexo de este manual contiene un índice con todos los consejos de migración, lo que permite encontrar rápidamente al consejo que se desee consultar.

Acerca de StarOffice Basic

El lenguaje de programación StarOffice Basic se ha desarrollado especialmente para StarOffice y está plenamente integrado dentro de éste.

Como su propio nombre indica, StarOffice Basic es un lenguaje de programación de la familia Basic. Quienes hayan trabajado anteriormente con otros lenguajes de Basic, especialmente con Visual Basic o Visual Basic for Applications (VBA) de Microsoft, se acostumbrarán con rapidez a StarOffice Basic. Gran parte de las construcciones básicas de StarOffice Basic son compatibles con Visual Basic.

El lenguaje de programación StarOffice Basic se puede dividir en cuatro componentes:

- **El lenguaje de StarOffice Basic**: define las construcciones elementales del lenguaje; por ejemplo, declaraciones de variables, bucles y funciones.
- **La biblioteca de ejecución**: proporciona funciones estándar sin referencia directa a StarOffice; por ejemplo, funciones para la edición de números, cadenas de caracteres, fechas o archivos.
- **La API (Interfaz de programación de aplicaciones) de StarOffice**: permite acceder a los documentos de StarOffice, crearlos, guardarlos, modificarlos e imprimirlos.
- **El Editor de diálogos**: crea cuadros de diálogo personalizados y ofrece un entorno para agregar elementos de control y gestores de eventos.

La compatibilidad entre StarOffice Basic y VBA hace referencia tanto al lenguaje StarOffice Basic como a la biblioteca de ejecución. La API de StarOffice y el Editor de diálogos *no* son compatibles con VBA (la estandarización de dichas interfaces hubiese impedido la incorporación de muchos de los conceptos proporcionados por StarOffice).

Usuarios previstos de StarOffice Basic

El ámbito de aplicación de StarOffice Basic empieza donde terminan las funciones estándar de StarOffice. StarOffice Basic permite automatizar las tareas rutinarias, crear vínculos con otros programas (por ejemplo, un servidor de bases de datos) y utilizar scripts predefinidos para llevar a cabo actividades complejas con una simple pulsación de botón.

StarOffice Basic ofrece acceso a la totalidad de funciones de StarOffice, es compatible con todas ellas, puede modificar los tipos de documentos y ofrece opciones para crear cuadros de diálogo personalizados.

Uso de StarOffice Basic

Cualquier usuario puede utilizar StarOffice Basic sin necesidad de programas ni ayudas adicionales. Incluso con la instalación estándar, StarOffice Basic dispone de todos los componentes necesarios para crear macros de Basic propias; se incluyen:

- **El entorno de desarrollo integrado** (IDE) incluye un editor para la entrada y verificación de macros.
- **El intérprete** necesario para ejecutar macros de StarOffice Basic.
- **Las interfaces** para diversas aplicaciones de StarOffice que permiten acceder directamente a documentos de Office.

Estructura de este manual

Los tres primeros capítulos constituyen una introducción a StarOffice Basic:

- Capítulo 2: El lenguaje de StarOffice Basic
- Capítulo 3: La biblioteca de ejecución de StarOffice Basic
- Capítulo 4: Introducción a la API de StarOffice

Estos capítulos ofrecen una visión general de StarOffice Basic; es conveniente leerlos antes de escribir programas en StarOffice Basic.

El resto de capítulos describe con mayor detalle los componentes individuales de la API de StarOffice; pueden leerse de forma selectiva según sea necesario:

- Capítulo 5: Trabajo con documentos de StarOffice
- Capítulo 6: Documentos de texto:
- Capítulo 7: Documentos de hoja de cálculo
- Capítulo 8: Dibujos y presentaciones
- Capítulo 9: Diagramas
- Capítulo 10: Acceso a bases de datos
- Capítulo 11: Diálogos
- termCapítulo 12: Formularios

Información adicional

Para la selección de los componentes de la API de StarOffice que se comentan en este manual se ha tenido en cuenta la utilidad que tienen para el programador de StarOffice Basic. Por ello, generalmente sólo se tratan aspectos generales de las interfaces. Si se desea obtener más detalles al respecto se ha de consultar la referencia de la API, disponible en la siguiente dirección de Internet:

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

La Developer's Guide (Guía del desarrollador) describe la API de StarOffice con mayor detalle que este manual, pero está principalmente dirigida a programadores de Java y C++. Quienes ya estén familiarizados con el lenguaje de programación StarOffice Basic encontrarán en dicha guía información adicional acerca de StarOffice Basic y la programación en StarOffice; se puede descargar de la siguiente dirección de Internet:

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

Los programadores que deseen trabajar directamente con Java o C++ en vez de con StarOffice Basic deberán consultar la Developer's Guide de StarOffice en lugar de este manual. La programación en StarOffice mediante Java o C++ es considerablemente más compleja que mediante StarOffice Basic.

CAPÍTULO **2**

El lenguaje de StarOffice Basic

StarOffice Basic pertenece a la familia de lenguajes Basic. Muchas de sus partes son idénticas a las contrapartidas de Microsoft Visual Basic for Applications y Microsoft Visual Basic. Quienes hayan utilizado estos lenguajes se acostumbrarán rápidamente a StarOffice Basic.

Los programadores de otros lenguajes, como Java, C++ o Delphi, tampoco tendrán problemas para familiarizarse rápidamente con StarOffice Basic. Éste es un lenguaje de programación por procedimientos totalmente desarrollado y ya no utiliza estructuras de control rudimentarias, como `GoTo` y `GoSub`.

También podrá obtener todas las ventajas de la programación orientada a objetos, puesto que StarOffice Basic dispone de una interfaz que permite utilizar bibliotecas de objetos externas. La API de StarOffice se basa en este tipo de interfaces que se describen más adelante.

Este capítulo ofrece una visión general de los elementos y construcciones principales del lenguaje StarOffice Basic, así como la estructura de las aplicaciones y bibliotecas desde el punto de vista de StarOffice Basic.

Visión general de un programa de StarOffice Basic

StarOffice Basic es un lenguaje interpretado. A diferencia de C++ o Turbo Pascal, el compilador de StarOffice no crea archivos ejecutables o autoextraíbles que puedan ejecutarse de forma automática. Un programa de StarOffice Basic se puede ejecutar pulsando un botón. El código se verifica primero para localizar los errores más obvios y a continuación se ejecuta línea por línea.

Líneas de programa

El funcionamiento línea por línea del intérprete de Basic representa una de las diferencias fundamentales entre Basic y otros lenguajes de programación. Mientras que la posición de los saltos de línea duros en el código fuente de los programas en Java, C++ o Delphi, por ejemplo, es irrelevante, cada una de las líneas de un programa en Basic constituye una unidad independiente. Las llamadas a funciones, las expresiones matemáticas y otros elementos del lenguaje, como las cabeceras de funciones y bucles, deben completarse en la misma línea en la que empiezan.

Si el espacio no es suficiente o si las líneas son muy largas, es posible enlazar varias líneas mediante signos de subrayado (_). En el ejemplo siguiente se muestra cómo se enlazan cuatro líneas que constituyen una única expresión matemática:

```
ExpresionLarga = (Expresion1 * Expresion2) + _
      (Expresion3 * Expresion4) + _
      (Expresion5 * Expresion6) + _
      (Expresion7 * Expresion8)
```

El subrayado debe ser siempre el último carácter de una línea enlazada y no puede ir seguido de un espacio o un tabulador; en caso contrario, el código generará un error.

Aparte de enlazar líneas individuales, StarOffice Basic permite a los programadores utilizar el signo de dos puntos para dividir una línea en varias secciones y aprovechar el espacio para varias expresiones. Las asignaciones

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

se pueden escribir, por ejemplo, de la siguiente forma:

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

Comentarios

Además del código ejecutable, los programas de StarOffice Basic pueden contener comentarios en los que se dan explicaciones acerca de las distintas partes del programa, así como información importante mediante la cual se puede volver a analizar más adelante el código del programa para, por ejemplo, solucionar errores.

StarOffice Basic ofrece dos métodos para introducir comentarios en el código del programa:

- Todos los caracteres situados a continuación de un apóstrofe se tratan como comentarios:

  ```
  Dim A    ' Éste es un comentario sobre la variable A
  ```

- La palabra clave `Rem`, seguida por el comentario.

  ```
  Rem Este comentario está precedido de la palabra clave Rem.
  ```

Un comentario suele incluir todos los caracteres hasta el final de la línea. StarOffice Basic interpreta la línea siguiente de nuevo como una instrucción normal. Si los comentarios abarcan varias líneas, es necesario identificar cada una de ellas como un comentario:

```
Dim B       ' Este comentario acerca de la variable B es relativamente largo
            ' y abarca varias líneas. El
            ' carácter de comentario debe repetirse
            ' en cada línea.
```

Marcador

Un programa de StarOffice Basic puede contener docenas, cientos e incluso miles de ***marcadores*** (nombres de variables, constantes, funciones, etc.). Al seleccionar un nombre de marcador, tenga en cuenta lo siguiente:

- Los marcadores sólo pueden contener caracteres latinos, números y signos de subrayado (_).
- El primer carácter de un marcador debe ser una letra o un subrayado.
- Los marcadores no pueden contener caracteres especiales, como ä, â, î ß.
- La longitud máxima de un marcador es de 255 caracteres.
- No hay distinción entre mayúsculas y minúsculas. Por ejemplo, el marcador `UnaVariableDePrueba` define la misma variable que `unavariabledeprueba` o que `UNAVARIABLEDEPRUEBA.`

  Esta regla tiene una excepción: hay distinción entre mayúsculas y minúsculas en las constantes de la API UNO. Encontrará más información acerca de UNO en el capítulo 4.

Las reglas de construcción de marcadores son distintas en StarOffice Basic y en VBA. Por ejemplo, a diferencia de VBA, StarOffice Basic no permite utilizar caracteres especiales en los marcadores, ya que podrían provocar problemas en los proyectos internacionales.

A continuación se muestran varios ejemplos de marcadores correctos e incorrectos:

```
Apellido             ' Correcto
Apellido5            ' Correcto (el número 5 no es el primer carácter)
Nombre Completo            ' Incorrecto (no se permite utilizar espacios)
DéjàVu                     ' Incorrecto (no se permite utilizar letras como é, à.)
5Apellidos           ' Incorrecto (el primer carácter no puede ser un número)
Nombre,Completo            ' Incorrecto (no se permite utilizar comas ni puntos)
```

Trabajo con variables

Declaración de variables implícita

Los lenguajes Basic están diseñados para ser fáciles de usar. StarOffice Basic permite crear una variable simplemente utilizándola, sin una declaración explícita. En otras palabras, una variable existe desde el mismo momento en que aparece en el código. En función de las variables ya presentes, el siguiente fragmento puede declarar hasta tres variables nuevas:

```
a = b + c
```

La declaración implícita de variables no es una práctica deseable, ya que puede provocar la introducción inesperada de una nueva variable mediante un simple error mecanográfico, por ejemplo. En lugar de generar un mensaje de error, el intérprete simplemente inicializa el error mecanográfico como variable nueva con valor 0. Localizar este tipo de errores en el código puede ser muy difícil

Declaración de variables explícita

Para impedir los errores que provoca la declaración de variables implícita, StarOffice Basic dispone de una opción denominada:

```
Option Explicit
```

Esta opción debe aparecer en la primera línea de programa de cada uno de los módulos. La opción garantiza que se genere un mensaje de error si alguna de las variables utilizadas no se ha declarado. Es conveniente incluir la opción `Option Explicit` en todos los módulos de Basic.

Es su forma más simple, la orden de declaración explícita de una variable tiene el siguiente aspecto:

```
Dim MiVar
```

En este ejemplo se declara una variable con el nombre `MiVar` y el tipo variant. Un variant es una variable universal que puede contener cualquier tipo de valor concebible, incluidas cadenas de caracteres, números enteros, números en coma flotante y valores lógicos o booleanos. He aquí varios ejemplos de variables de tipo Variant:

```
MiVar = "Hola mundo"        ' Asignación de una cadena
MiVar = 1                         ' Asignación de un número entero
MiVar = 1.0                       ' Asignación de un número de coma flotante
MiVar = True                      ' Asignación de un valor booleano
```

Las variables declaradas de esta forma se pueden incluso utilizar para distintos tipos de variable en un mismo programa. Aunque esta característica proporciona mucha flexibilidad, es más conveniente restringir una variable a un único tipo. Cuando StarOffice Basic detecta un tipo de variable definido incorrectamente en un contexto específico, se genera un mensaje de error.

Para hacer una declaración de variable de tipo fijo, utilice esta forma:

```
Dim MiVar As Integer    ' Declaración de una variable de tipo entero
```

La variable se declara con el tipo entero, lo que le permite contener valores numéricos enteros. También puede utilizar el siguiente estilo para declarar una variable de tipo entero:

```
Dim MiVar%              ' Declaración de una variable de tipo entero
```

La instrucción `Dim` puede contener varias declaraciones de variables:

```
Dim MiVar1, MiVar2
```

Si desea asignar a estas variables un tipo permanente, deberá efectuar asignaciones independientes para cada una de ellas:

```
Dim MiVar1 As Integer, MiVar2 As Integer
```

Si no se declara el tipo de una variable, StarOffice Basic le asigna el tipo variant. Por ejemplo, en la siguiente declaración de variable, `MiVar1` se crea como variant y `MiVar2` se crea como entero:

```
Dim MiVar1, MiVar2 As Integer
```

En las secciones siguientes se enumeran los tipos de variable disponibles en StarOffice Basic y se describa la forma de usarlos y declararlos.

Cadenas

Las cadenas, conjuntamente con los números, son los tipos básicos más importantes de StarOffice Basic. Una cadena consta de una secuencia de caracteres individuales consecutivos. El sistema guarda internamente las cadenas como secuencias de números, en las que cada número representa un carácter específico.

De un juego de caracteres ASCII a Unicode

Los juegos de caracteres asignan en una tabla los caracteres de una cadena al código correspondiente (números y caracteres); dicha tabla indica al sistema cómo se representa el carácter en pantalla o en una impresora.

El juego de caracteres ASCII

El juego de caracteres ASCII es un conjunto de códigos que representan números, caracteres y símbolos especiales mediante un byte de datos. Los códigos ASCII 0 a 127 corresponden al alfabeto y a los símbolos más comunes (como puntos, paréntesis y comas), así como algunos códigos especiales de control para la pantalla y la impresora. El juego de caracteres ASCII se suele utilizar como formato estándar para la transferencia de datos de texto entre sistemas.

No obstante, dicho juego de caracteres no incluye, por ejemplo, muchos de los caracteres especiales utilizados en Europa, como â, ä e î, ni otros formatos de caracteres, como el alfabeto cirílico.

El juego de caracteres ANSI

Microsoft basa su producto Windows en el juego de caracteres ANSI (American National Standards Institute) que se ha ampliado gradualmente hasta incluir caracteres que no están en ASCII.

Páginas de códigos

Los juegos de caracteres ISO 8859 han ofrecido un estándar internacional que está obsoleto desde hace tiempo. Los primeros 128 caracteres del juego de caracteres ISO corresponden a los caracteres ASCII. El estándar ISO introduce nuevos juegos de caracteres (***páginas de códigos***) para poder mostrar correctamente más idiomas. Sin embargo, la consecuencia es que el mismo valor representa caracteres distintos en los distintos idiomas.

Unicode

Unicode aumenta la longitud de cada carácter a cuatro bytes y combina distintos juegos de caracteres para crear un estándar de representación para el máximo número de idiomas posible. Muchos programas (incluidos StarOffice y StarOffice Basic) admiten ya la versión 2.0 de Unicode.

Variables de cadena

StarOffice Basic guarda las cadenas de caracteres como variables de cadena en Unicode. Una variable de cadena puede almacenar un máximo de 65535 caracteres. Internamente, StarOffice Basic guarda el valor Unicode asociado a cada carácter. La memoria de trabajo necesaria para una variable de cadena depende de la longitud de ésta.

Ejemplo de declaración de una variable de cadena:

```
Dim Variable as String
```

Esta declaración se puede también escribir como:

```
Dim Variable$
```

Al portar aplicaciones de VBA, compruebe que se respete la máxima longitud de cadena permitida en StarOffice Basic (65535 caracteres).

Especificación de cadenas explícitas

Para asignar una cadena explícita a una variable de cadena, escriba la cadena entre comillas (").

```
Dim MiCadena As String
MiCadena = " Esto es una prueba"
```

Para dividir una cadena en dos líneas, agregue un signo más al final de la primera línea:

```
Dim MiCadena As String
MiCadena =    "Esta cadena es tan larga que" + _
              "la hemos dividido en dos líneas."
```

Para incluir un signo de comillas (") dentro de una cadena, escríbalo dos veces en el lugar correspondiente:

```
Dim MiCadena As String
MiCadena = "un signo de "" comillas."      ' produce un signo de comillas "
```

Números

StarOffice Basic puede procesar cinco tipos básicos de números:

- Entero (Integer)
- Entero largo (Long Integer)
- Coma flotante (Float)
- Doble precisión (Double)
- Moneda (Currency)

Integer

Las variables de tipo Integer pueden almacenar cualquier número entero entre -32768 y 32767. Una variable entera puede ocupar un máximo de dos bytes de memoria. EL símbolo de declaración de tipo para una variable entera es %. Los cálculos con variables enteras son muy rápidos, por lo que son especialmente útiles en contadores de bucles. Si asigna un número de coma flotante a una variable entera, el número se redondeará hacia arriba o hacia abajo al número entero más próximo.

Ejemplos de declaraciones de variables enteras:

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

Long Integer

Las variables de tipo Long integer (entero largo) pueden almacenar cualquier número entero entre 2147483648 y 2147483647. Una variable de tipo entero largo puede ocupar un máximo de cuatro bytes de memoria. El símbolo de declaración de tipo para una variable de tipo entero largo es &. Los cálculos con variables de tipo entero largo son muy rápidos, por lo que son especialmente útiles en contadores de bucles. Si asigna un número de coma flotante a una variable de tipo entero largo, el número se redondeará hacia arriba o hacia abajo al número entero más próximo.

Ejemplos de declaraciones de variables de tipo entero largo:

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

Single

Las variables Single (coma flotante de precisión simple) pueden almacenar números de coma flotante positivos o negativos entre $3{,}402823 \times 10^{38}$ y $1{,}401298 \times 10^{-45}$. Una variable de tipo simple puede ocupar un máximo de cuatro bytes de memoria. El símbolo de declaración de tipo para una variable de tipo simple es !.

Originalmente, las variables de tipo simple se utilizaban para reducir el tiempo de cálculo requerido por las variables de doble precisión. No obstante, las consideraciones de velocidad son ya irrelevantes, lo que reduce la necesidad de utilizarlas.

Ejemplos de declaración de variables de tipo simple:

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

Double

Las variables Double (coma flotante de doble precisión) pueden almacenar números de coma flotante positivos o negativos entre 1,79769313486232 x 10^{308} y 4.94065645841247 x 10^{-324}. Una variable doble puede ocupar un máximo de ocho bytes de memoria. Las variables dobles son adecuadas para cálculos precisos. El símbolo de declaración de tipo es #.

Ejemplos de declaración de variables dobles:

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

Variables Currency

Las variables de tipo Currency (moneda) difieren del resto de tipos de variables en la forma en que manejan los valores. El signo decimal es fijo y va seguido de cuatro decimales. La variable puede contener un máximo de 15 números antes del signo decimal. Una variable de moneda puede almacenar valores entre -922337203685477,5808 y +922337203685477,5807 y ocupa un máximo de ocho bytes de memoria. El símbolo de declaración de tipo para una variable de moneda es @.

Las variables de moneda se utilizan principalmente en cálculos económicos que generan errores de redondeo no previsibles con el uso de variables de coma flotante.

Ejemplos de declaración de variables de moneda:

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

Especificación de números explícitos

Los números se pueden presentar de diversas formas; por ejemplo, en formato decimal o en notación científica, o incluso en una base distinta a la decimal. Los caracteres numéricos cumplen las siguientes limitaciones en StarOffice Basic:

Números enteros

El método más simple es trabajar con números enteros. En el texto fuente aparecen sin la coma de separación de miles:

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

Los números pueden ir precedidos de los signos más (+) o menos (-) (con o sin espacio):

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

Números decimales

Al introducir un número decimal, utilice el punto (.) como separador de decimales. De esta forma se garantiza que el código fuente se puede transferir de un país a otro sin necesidad de conversión.

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

A = 1223.53       ' está redondeado
B = - 23446.46    ' está redondeado
C = + 3532.76323
```

También puede usar los signos más (+) o menos (-) como prefijos de los números decimales (con o sin espacios).

Si se asigna un número decimal a una variable entera, StarOffice Basic redondea la cifra hacia arriba o hacia abajo.

Estilo de escritura exponencial

StarOffice Basic permite especificar números con estilo de escritura exponencial; por ejemplo, se puede escribir 1.5e-10 para el número 1,5 •~ 10^{-10} (0,00000000015). La letra "e" puede estar en mayúscula o en minúscula y puede ir precedida del signo (+) .

Estos son varios ejemplos, unos correctos y otros incorrectos, de números en formato exponencial:

```
Dim A As Double

A = 1.43E2           ' Correcto
A = + 1.43E2         ' Correcto (espacio entre el signo más y el número)
A = -1.43E2          ' Correcto (espacio entre el signo menos y el número)
A = 1.43E-2          ' Correcto (exponente negativo)

A = 1.43E -2         ' Incorrecto (no se permiten los espacios dentro del número)
A = 1,43E-2          ' Incorrecto (no se permite utilizar comas como separadores
decimales)
A = 1.43E2.2         ' Incorrecto (el exponente debe ser un número entero)
```

Tenga en cuenta que los ejemplos incorrectos primero y tercero no generan ningún mensaje de error, a pesar de que las variables devuelvan valores incorrectos. La expresión

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

se interpreta como 1,43 menos 2, que corresponde al valor -0.57. En realidad, el valor que se pretendía escribir era $1,43 * 10^2$ (que corresponde a 0.0143). En el valor

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

StarOffice Basic ignora la parte del exponente situada después del signo decimal e interpreta la expresión como

```
A = 1.43E2
```

Valores hexadecimales

El sistema hexadecimal (sistema de base 16) tiene la ventaja de que un número de dos dígitos corresponde exactamente a un byte. Esto permite manejar los números de una forma más próxima a la arquitectura de la máquina. En el sistema hexadecimal se utilizan como dígitos los números del 0 al 9 y las letras de la A a la F. La A representa el número decimal 10, mientras que la F representa el 15. StarOffice Basic permite utilizar valores hexadecimales como números enteros, mientras vayan precedidos de `&H`.

```
Dim A As Long

A = &HFF      ' Valor hexadecimal FF, corresponde al valor decimal 255
A = &H10      ' Valor hexadecimal 10, corresponde al valor decimal 16
```

Valores octales

StarOffice Basic entiende también el sistema octal (base 8) que utiliza los números del 0 al 7; para ello se deben especificar números enteros precedidos de `&O`.

```
Dim A As Long

A = &O77      ' Valor octal 77, corresponde al valor decimal 63
A = &O10      ' Valor octal 10, corresponde al valor decimal 8
```

Variables booleanas

Las variables lógicas o booleanas sólo pueden contener uno de estos dos valores: `True` (verdadero) y `False` (falso). Son adecuadas para especificaciones binarias que sólo pueden adoptar uno de los dos estados mencionados. Un valor booleano se guarda internamente como valor de dos bytes; 0 corresponde al valor `False` y cualquier otro valor a `True`. Las variables booleanas no tienen un símbolo de declaración de tipo asociado. Sólo se puede efectuar la declaración mediante el uso del fragmento ***As Boolean***. Un ejemplo de declaración de una variable booleana es:

```
Dim Variable As Boolean
```

Detalles de fecha y hora

Variables Date

Las variables Date (fecha) pueden contener valores de fecha y hora. Para guardar valores de fecha, StarOffice Basic utiliza un formato interno que permite efectuar comparaciones y operaciones matemáticas entre valores de fecha y hora. Las variables de fecha no tienen un símbolo de declaración de tipo asociado. Sólo se puede efectuar la declaración mediante el uso del fragmento ***As Date***.

Ejemplo de declaración de una variable de fecha:

```
Dim Variable As Date
```

Campos de datos

Además de las variables sencillas (***escalares***), StarOffice Basic admite también campos de datos (***matrices***). Un campo de datos contiene varias variables a las que se puede acceder a través de un índice.

Matrices simples

Una declaración de matriz es similar a la de una variable simple pero, a diferencia de ésta, el nombre de la matriz va seguido por un paréntesis que contiene la especificación del número de elementos. La expresión

```
Dim MiMatriz(3)
```

declara una matriz con cuatro variables de tipo variant: `MiMatriz(0)`, `MiMatriz(1)`, `MiMatriz(2)` y `MiMatriz(3)`.

También se pueden declarar matrices de variables de tipos específicos; por ejemplo, la línea siguiente representa una declaración de matriz con cuatro variables enteras:

```
Dim MiEntero(3) As Integer
```

En los ejemplos anteriores, el índice de la matriz empieza siempre por el valor inicial estándar de 0. Es posible especificar un rango de validez con valores inicial y final en la declaración del campo de datos. En el ejemplo siguiente se declara un campo de datos con seis valores enteros que tiene asignados los índices 5 al 10:

```
Dim MiEntero(5 To 10)
```

Los índices no tienen por qué ser valores positivos. En el ejemplo siguiente se muestra una declaración correcta con límites negativos.

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

En ella se declara un campo de datos enteros con 6 valores a los que se puede acceder mediante los índices -10 a -5.

Los índices de los campos de datos deben cumplir tres limitaciones:

- El menor índice posible es -32768.
- El mayor índice posible es 32767.
- El número máximo de elementos (en una dimensión del campo de datos) es de 16368.

A veces se aplican otros valores de límite para índices de campos de datos en VBA. El mismo tipo de limitaciones se aplica, asimismo, al número máximo de elementos posibles por dimensión. Los valores válidos aparecen en la documentación de VBA pertinente.

Valor especificado para el índice inicial

El índice inicial de un campo de datos suele ser 0. Es posible cambiar a 1 el índice inicial para todas las declaraciones de campos de datos mediante la llamada:

```
Option Base 1
```

Dicha llamada deberá incluirse en la cabecera de un módulo si se desea aplicar a todas las declaraciones de matrices de éste. Esta llamada, no obstante, no afecta a las secuencias de UNO definidas a través de la API de StarOffice, cuyo índice empieza ***siempre*** por 0. Para mejorar la claridad, es conveniente no utilizar Option Base 1.

El uso de Option Base 1 no afecta al número de elementos de la matriz, únicamente al índice inicial de la ésta. La declaración

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

crea 4 variables enteras que pueden describirse mediante las expresiones `MiEntero(1)`, `MiEntero(2)`, `MiEntero(3)` y `MiEntero(4)`.

En StarOffice Basic, la expresión Option Base 1 no afecta al número de elementos de la matriz, a diferencia de lo que sucede en VBA. En StarOffice Basic cambia únicamente el índice inicial. Mientras que la declaración `MiEntero(3)` crea en VBA tres variables enteras con los índices 1 a 3, la misma declaración en StarOffice Basic crea cuatro variables enteras con los índices 1 a 4.

Campos de datos multidimensionales

Además de los campos de datos de una dimensión, StarOffice Basic puede trabajar también con campos de datos multidimensionales. Las dimensiones correspondientes se separan entre sí mediante comas. El ejemplo

```
Dim MiMatrizEntera(5, 5)
```

define una matriz entera de dos dimensiones, con 6 índices en cada una de ellas (a los que se puede acceder mediante los valores 0 a 5). La matriz puede almacenar un total de 6 × 6 = 36 valores enteros.

Aunque StarOffice Basic permite definir matrices con cientos de dimensiones, la memoria disponible limita el número de éstas posible.

Cambios dinámicos en las dimensiones de los campos de datos

Los ejemplos anteriores se basan en campos de datos con una dimensión específica. Se pueden también definir matrices en las que las dimensiones de los campos de datos cambien de forma dinámica. Por ejemplo, se puede definir una matriz que contenga todas las palabras de un texto que empiecen con la letra A. Puesto que el número de dichas palabras no se conoce inicialmente, es necesario poder cambiar los límites del campo. Para hacerlo, utilice la siguiente llamada en StarOffice Basic:

```
ReDim MiMatriz(10)
```

A diferencia de VBA, que sólo permite especificar mediante Dim MiMatriz() las dimensiones de matrices dinámicas, StarOffice Basic permite modificar tanto las matrices estáticas como las dinámicas mediante ReDim.

En el ejemplo siguiente se modifica la dimensión de la matriz inicial para que pueda almacenar 11 o 21 valores:

```
Dim MiMatriz(4) As Integer          ' Declaración con cinco elementos

' ...

ReDim MiMatriz(10) As Integer              ' Aumentar a 11 elementos

' ...

ReDim MiMatriz(20) As Integer              ' Aumentar a 21 elementos
```

Al restablecer las dimensiones de una matriz se puede utilizar cualquiera de las opciones indicadas en las secciones anteriores. Entre éstas, la declaración de campos de datos multidimensionales o la especificación de valores de inicio y final explícitos. Cuando se modifican las dimensiones de un campo de datos, el contenido de éste ***se pierde***. Si desea conservar los valores originales utilice la orden `Preserve`:

```
Dim MiMatriz(10) As Integer                        ' Definición de las
                                                   ' dimensiones iniciales

' ...

ReDim Preserve MiMatriz(20)  As Integer            ' Ampliar el
                                                   ' campo de datos
                                                   ' conservando el contenido
```

Al utilizar `Preserve` se garantiza que el número de dimensiones y el tipo de variables permanezcan invariables.

A diferencia de VBA, que sólo permite cambiar el límite superior de la última dimensión del campo de datos al utilizar `Preserve`, StarOffice Basic permite también cambiar las otras dimensiones.

Si utiliza `ReDim` junto con `Preserve`, deberá utilizar el mismo tipo de datos especificado en la declaración original delcampo de datos.

Ámbito y vida de las variables

Las variables de StarOffice Basic tienen una vida y un ámbito limitados en los que pueden leerse y pueden utilizarse en otros fragmentos de programa. El tiempo de conservación de una variable y el ámbito desde el que se puede acceder a ésta dependen de la ubicación y el tipo especificados para ella.

Variables locales

Las variables declaradas dentro de una función o procedimiento se denominan variables locales:

```
Sub Prueba
      Dim MiEntero As Integer

      ' ...
End Sub
```

Las variables locales sólo son válidas mientras la función o procedimiento se están ejecutando; a continuación se reinician con el valor cero. Cada vez que se llama a la función, los valores generados anteriormente ya no están disponibles.

Para conservar los valores anteriores se debe definir la variable como `Static`:

```
Sub Prueba
      Static MiEntero As Integer

      ' ...

End Sub
```

A diferencia de VBA, StarOffice Basic comprueba que el nombre de una variable local no se utilice simultáneamente como variable global y como variable privada en la cabecera del módulo. Al portar una aplicación de VBA a StarOffice Basic será necesario cambiar los nombres de variable duplicados.

Variables de dominio público

Las variables de dominio público se definen en la sección de cabecera de un módulo mediante la palabra clave `Dim`. Dichas variables están disponibles para todos los módulos de su biblioteca:

Módulo A:

```
Dim A As Integer

Sub Prueba
      Flip
      Flop
End Sub

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

Módulo B:

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

El valor de la variable `A` no cambia por la función `Prueba`, pero la función `Flip` lo incrementa en uno y la función `Flop` lo disminuye en uno. Ambos cambios en la variable son globales.

En lugar de `Dim` se puede utilizar la palabra clave `Public` para declarar una variable de dominio público:

```
Public A As Integer
```

Una variable de dominio público sólo está disponible mientras se está ejecutando la macro asociada; a continuación, la variable se reinicia.

Variables globales

Desde el punto de vista de su función, las variables globales son similares a las variables de dominio público, con la excepción de que sus valores se conservan incluso después de que la macro asociada se haya ejecutado. Las variables globales se declaran en la sección de cabecera de un módulo mediante la palabra clave `Global`:

```
Global A As Integer
```

Variables privadas

Las variables privadas sólo están disponibles en el módulo en el que se definen. Utilice la palabra clave Private para definir la variable:

```
Private MiEntero As Integer
```

Si hay varios módulos que contienen una variable privada con el mismo nombre, StarOffice Basic crea una variable distinta para cada una de las apariciones del nombre. En el ejemplo siguiente, tanto el módulo A como el módulo B contienen una variable privada denominada C. La función Prueba en primer lugar establece la variable privada del módulo A y a continuación hace lo mismo con la del módulo B.

Módulo A:

```
Private C As Integer

Sub Prueba
      SetModuleA            ' Establece la variable C del módulo A
      SetModuleB            ' Establece la variable C del módulo B

      ShowVarA              ' Muestra la variable C del módulo A (= 10)
      ShowVarB              ' Muestra la variable C del módulo B (= 20)
End Sub

Sub SetmoduleeA
      A = 10
End Sub

Sub ShowVarA
      MsgBox C              ' Muestra la variable C del módulo A.
End Sub
```

Módulo B:

```
Private C As Integer

Sub SetModuleB
     A = 20
End Sub

Sub ShowVarB
      MsgBox C              ' Muestra la variable C del módulo B.
End Sub
```

Constantes

Para declarar una constante en StarOffice Basic utilice la palabra clave `Const`.

```
Const A = 10
```

Si lo desea puede especificar en la declaración el tipo de constante:

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

Operadores

StarOffice Basic comprende los operadores matemáticos, lógicos y de comparación comunes.

Operadores matemáticos

Los operadores matemáticos se pueden utilizar con todo tipo de números; asimismo, el operador `+` puede utilizarse para enlazar cadenas.

`+` Adición de números y fechas, enlace de cadenas

`-` Substracción de números y fechas

`*` Multiplicación de números

`/` División de números

`\` División de números con resultado entero (redondeado)

`^` Elevación de números a potencias

`MOD` Operación módulo (cálculo del resto de una división)

Operadores lógicos

Los operadores lógicos permiten vincular elementos según las reglas del álgebra de Boole. Si los operadores se aplican a variables booleanas, la operación proporciona directamente el resultado requerido. Si se usan con valores enteros o enteros largos, la operación se efectúa a nivel de bit.

`AND` Operación de Y lógico

`OR` Operación de O lógico

`XOR` Operación de O exclusivo

`NOT` Negación lógica

`EQV` Prueba de equivalencia (ambas partes `True` o `False`)

`IMP` Implicación (si la primera expresión es verdadera [True], la segunda también debe ser verdadera)

Operadores de comparación

Los operadores de comparación se pueden aplicar a todos los tipos de variables elementales (números, fechas, cadenas y valores booleanos).

=	Igualdad de números, valores de fecha y cadenas
<>	Desigualdad de números, valores de fecha y cadenas
>	Comprobación "mayor que" para números, fechas y cadenas
>=	Comprobación "mayor o igual que" para números, fechas y cadenas
<	Comprobación "menor que" para números, fechas y cadenas
<=	Comprobación "menor o igual que" para números, fechas y cadenas

StarOffice Basic no admite el operador de comparación de VBA `Like`.

Bifurcación

Utilice expresiones de bifurcación para restringir la ejecución de un bloque de código mientras no se cumpla una condición determinada.

If...Then...Else

La expresión de bifurcación más común es una expresión `If`.Por ejemplo:

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

La asignación `B = 2` sólo tiene lugar cuando el valor de la variable A es mayor que tres. Una variación de la expresión `If` es la cláusula `If/Else`:

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

En este ejemplo se asigna a la variable B el valor 2 si A es mayor que 3; en caso contrario se asigna a B el valor 0.

Para construir expresiones más complejas, éstas se pueden unir en cascada, por ejemplo:

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

Si el valor de la variable A es igual a cero se asigna a B el valor 0. Si A es menor que 3 (pero no igual a cero) se asigna a B el valor 1. En cualquier otro caso (es decir, si A es mayor o igual que 3) se asigna a B el valor 2.

Select...Case

La instrucción `Select...Case` es una alternativa a las expresiones `If` en cascada y se utiliza cuando se debe comprobar el valor de una variable con varias condiciones:

```
Select Case DiaDeLaSemana
Case 1:
      NombreDeDiaDeLaSemana = "Domingo"
Case 2:
      NombreDeDiaDeLaSemana = "Lunes"
Case 3:
      NombreDeDiaDeLaSemana = "Martes"
Case 4:
      NombreDeDiaDeLaSemana = "Miércoles"
Case 5:
      NombreDeDiaDeLaSemana = "Jueves"
Case 6:
      NombreDeDiaDeLaSemana = "Viernes"
Case 7:
      NombreDeDiaDeLaSemana = "Sábado"
End Select
```

En este ejemplo, el nombre de un día de la semana corresponde a un número, de forma que se asigna a la variable `DiaDeLaSemana` el valor de 1 para `Domingo`, 2 para `Lunes`, etc.

La orden `Select` no está restringida a estas asignaciones simples de 1 a 1; en una bifurcación `Case` se pueden también especificar operadores de comparación o listas de expresiones. En el ejemplo siguiente se enumeran las variantes sintácticas más importantes:

```
Select Case Var
Case 1 To 5

      ' ... Var está entre los números 1 y 5

Case 6, 7, 8

      ' ... Var es 6, 7 u 8

Case Var > 8 And Var < 11

      ' ... Var es mayor que 8 y menor que 11

Case Else

      ' ... el resto de casos

End Select
```

Bucles

Un bucle ejecuta un bloque de código el número de veces especificado. También son posibles los bucles en los que no se especifica el número de ejecuciones.

For...Next

El bucle `For...Next` se ejecuta un número fijo de veces. El contador del bucle define el número de veces que éste se ejecuta. En el ejemplo siguiente,

```
Dim I

For I = 1 To 10

      ' ...  Parte interna del bucle

Next I
```

La variable `I` es el contador del bucle; su valor inicial es 1. El contador se incrementa en 1 al final de cada pase. Cuando la variable I es igual a 10, el bucle se detiene.

Si desea incrementar el contador del bucle al final de cada pase en un valor distinto de 1, utilice la función `Step`:

```
Dim I

For I = 1 To 10 Step 0.5

      ' ... Parte interna del bucle

Next I
```

En el ejemplo anterior, el contador se incrementa en 0,5 al final de cada pase y el bucle se ejecuta 19 veces.

También se pueden utilizar valores de Step negativos:

```
Dim I

For I = 10 To 1 Step -1

      ' ... Parte interna del bucle

Next I
```

En este ejemplo, el contador empieza en 10 y se reduce en 1 al final de cada pase hasta que llega a 1.

La instrucción `Exit For` permite salir de un bucle `For` de forma prematura.

En el ejemplo siguiente, el bucle termina durante el quinto pase:

```
Dim I
```

```
For I = 1 To 10

      If I = 5 Then
            Exit For
      End If

      ' ... Parte interna del bucle

Next I
```

La variante de bucle `For Each...Next` de VBA no se admite en StarOffice Basic.

Do...Loop

El bucle `Do...Loop` no está vinculado con un número de pases fijo. En vez de eso, `Do...Loop` se ejecuta hasta que se cumple una condición determinada. Hay cuatro variantes de `Do...Loop` (en los ejemplos siguientes, `A > 10` representa cualquier condición):

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

   ```
   Do While A > 10
       ' ... cuerpo del bucle
   Loop
   ```

   comprueba si la condición se sigue cumpliendo antes de cada pase y, en caso afirmativo, ejecuta el bucle.

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

   ```
   Do Until A > 10
         ' ... cuerpo del bucle
   Loop
   ```

   ejecuta el bucle hasta que la condición *deja de* cumplirse.

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

   ```
   Do
         ' ... cuerpo del bucle
   Loop While A > 10
   ```

   sólo comprueba la condición después del primer pase del bucle y termina si la condición se ***cumple***.

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

   ```
   Do
         ' ... cuerpo del bucle
   Loop Until A > 10
   ```

   también comprueba la condición después del primer pase, pero ejecuta el bucle hasta que la condición ***deja de*** cumplirse.

Como en el caso del bucle `For...Next` , `Do...Loop` ofrece una orden de terminación. La orden `Exit Do` puede salir del bucle en cualquier punto de éste.

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

      ' ... cuerpo del bucle
While A > 10
```

Ejemplo de programación: Ordenación mediante bucles anidados

Los usos de los bucles son diversos; por ejemplo, buscar en listas, devolver valores o ejecutar tareas matemáticas complejas. El ejemplo siguiente es un algoritmo que utiliza bucles para ordenar una lista de nombres.

```
Sub Sort
      Dim Entrada(1 To 10) As String
      Dim Contador As Integer
      Dim Contador2 As Integer
      Dim Temp As String

      Entrada(1) = "Patricia"
      Entrada(2) = "Carlos"
      Entrada(3) = "Tomás"
      Entrada(4) = "Miguel"
      Entrada(5) = "David"
      Entrada(6) = "Catalina"
      Entrada(7) = "Susana"
      Entrada(8) = "Eduardo"
      Entrada(9) = "Cristina"
      Entrada(10) = "Gerardo"

      For Contador = 1 To 10
            For Contador2 = Contador + 1 To 10
                  If Entrada(Contador) > Entrada(Contador2) Then
                        Temp = Entrada(Contador)
                        Entrada(Contador) = Entrada(Contador2)
                        Entrada(Contador2) = Temp
                  End If
            Next Contador2
      Next Contador

      For Contador = 1 To 10
            Print Entrada(Contador)
      Next Contador
End Sub
```

Los valores se intercambian por parejas diversas veces hasta que finalmente quedan ordenados en orden ascendente. Como si fuesen burbujas, las variables se desplazan gradualmente a la posición correcta. Es por ello que este algoritmo se denomina ***Ordenación por el método de la burbuja***.

Procedimientos y funciones

Los procedimientos y las funciones son puntos clave dentro de un programa. Constituyen la estructura que permite dividir un problema complejo en varias subtareas.

Procedimientos

Un ***procedimiento*** ejecuta una acción sin devolver un valor de forma explícita. Su sintaxis es

```
Sub Prueba

      ' ... código del procedimiento

End Sub
```

El ejemplo define un procedimiento denominado `Prueba` que contiene código al que se puede acceder desde cualquier punto del programa. La llamada se efectúa especificando el nombre del procedimiento en el lugar pertinente del programa:

```
Prueba
```

Funciones

Una ***función,*** igual que un procedimiento, combina en una unidad lógica un bloque de programa que debe ejecutarse. Sin embargo, a diferencia de un procedimiento, las funciones devuelven un valor.

```
Function Prueba

      ' ... código de la función

      Prueba = 123
End Function
```

El valor de retorno se indica mediante una simple asignación. Ésta no tiene por qué situarse al final de la función; puede estar en cualquier lugar dentro de la función.

La función superior se puede llamar desde un programa, de la siguiente forma:

```
Dim A

A = Prueba
```

En el código se define una variable `A` a la que se asigna el resultado de la función `Prueba`.

El valor de retorno se puede sobrescribir dentro de la función el número de veces que se desee. Como sucede en la asignación clásica de variables, la función del ejemplo devuelve el último valor que se le asignó.

```
Function Prueba

      Prueba = 12

      ' ...

      Prueba = 123

End Function
```

En este ejemplo, el valor de retorno de la función es `123`.

Si se para una asignación, la función devuelve un valor `cero` (el número 0 en el caso de valores numéricos, un espacio en blanco en el caso de cadenas).

El valor de retorno de la función puede ser de cualquier tipo. El tipo se declara de la misma forma que en las declaraciones de variables:

```
Function Prueba As Integer

      ' ... código de la función

End Function
```

Si se detiene la especificación de un valor explícito, se asigna al valor de retorno el tipo variant.

Terminación prematura de procedimientos y funciones

En StarOffice Basic se pueden utilizar las órdenes `Exit Sub` y `Exit Function` para terminar un procedimiento o función de forma prematura; por ejemplo, para la gestión de errores. Estas órdenes detienen el procedimiento o función y devuelven el programa al punto en el que se efectuó la llamada de dicho procedimiento o función.

En el ejemplo siguiente se muestra un procedimiento que termina su implementación cuando la variable `ErrorOcurrido` toma el valor `True`.

```
Sub Prueba
      Dim ErrorOcurrido As Boolean

      ' ...

      If ErrorOcurrido Then
            Exit Sub
      End If

      ' ...

End Sub
```

Paso de parámetros

Las funciones y procedimientos pueden recibir uno o más parámetros. Los parámetros esenciales deben aparecer entre paréntesis a continuación del nombre del procedimiento o función. El ejemplo

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

define un procedimiento que espera recibir como parámetros un valor entero A y una cadena B.

En StarOffice Basic, los parámetros se suelen pasar por ***referencia***. Los cambios efectuados en las variables se conservan al salir del procedimiento o función:

```
Sub Prueba
      Dim A As Integer
      A = 10
      CambiarValor(A)
      ' El valor del parámetro A es ahora 20
End Sub
Sub CambiarValor(ElValor As Integer)
      ElValor = 20
End Sub
```

En este ejemplo, el valor A definido en la función `Prueba` se pasa como parámetro a la función `CambiarValor`. A continuación, el valor se cambia a 20 y se pasa a `ElValor`, que se conserva al salir de la función.

También se puede pasar un parámetro como ***valor***, en el caso de que no desee que los cambios subsiguientes del parámetro afecten al valor que se pasa originalmente. Para especificar que un parámetro se pasará como valor, compruebe que la palabra clave `ByVal` preceda a la declaración de la variable en la cabecera de la función.

En el ejemplo anterior, si se sustituye la función `CambiarValor` por

```
Sub CambiarValor(ByVal ElValor As Integer)
      ElValor = 20
End Sub
```

dicho cambio no afecta a la variable superior A. Después de llamar a la función `CambiarValor`, la variable A conserva el valor `10`.

El método de paso de parámetros a procedimientos y funciones en StarOffice Basic es prácticamente idéntico al utilizado en VBA. De forma predeterminada, los parámetros se pasan por referencia. Para pasar los parámetros por valor, utilice la palabra clave `ByVal`. En VBA se puede utilizar también la palabra clave `ByRef` para forzar que un parámetro se pase por referencia. StarOffice Basic no admite esta palabra clave puesto que el paso por referencia es ya el comportamiento predeterminado.

Por regla general, las funciones y procedimientos de StarOffice Basic son `públicos`. Las palabras clave `Public` y `Private` que se utilizan en VBA no se admiten en StarOffice Basic.

Parámetros opcionales

Las funciones y procedimientos sólo pueden llamarse si en la llamada se pasan todos los parámetros necesarios.

StarOffice Basic permite definir parámetros ***opcionales***; es decir, si la llamada no incluye los valores correspondientes, StarOffice Basic pasa un parámetro vacío. En el ejemplo

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

End Sub
```

el parámetro A es obligatorio, mientras que B es opcional.

La función `Falta` comprueba si un parámetro se ha pasado o no.

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

      ' Comprobar si el parámetro B está presente
      If Not Falta (B) Then
             B_Local = B           ' el parámetro B está presente
      Else
             B_Local = 0           ' falta el parámetro B -> valor predeterminado 0
      End If

      ' ... Iniciar la función en sí

End Sub
```

El ejemplo comprueba en primer lugar si se ha pasado el parámetro `B` y, si es necesario, pasa el mismo parámetro a la variable interna `B_Local`. Si el parámetro correspondiente no está presente, se pasa un valor predeterminado (en este ejemplo, el valor `0`) a `B_ Local`, en lugar del parámetro.

La opción incorporada en VBA para la definición de valores predeterminados de parámetros opcionales no se admite en StarOffice Basic.

La palabra clave `ParamArray` de VBA no se admite en StarOffice Basic.

Recursión

StarOffice Basic permite ahora la recursión. Un procedimiento o una función recursivos pueden llamarse a sí mismos hasta que detectan el cumplimento de una condición de base. Al llamar a la función con la condición base, se devuelve un resultado.

El ejemplo siguiente utiliza una función recursiva para calcular el factorial de los números `42`, `-42` y `3,14`:

```
Sub Main
 Msgbox CalcularFactorial( 42 ) ' Muestra 1,40500611775288E+51
 Msgbox CalcularFactorial( -42 ) ' Muestra "N但ero no v疝ido para c疝culo del
factorial!"
```

```
 Msgbox CalcularFactorial( 3.14 ) ' Muestra "N但ero no v疝ido para c疝culo del
factorial!"
End Sub

Function CalcularFactorial( Numero )
 If Numero < 0 Or Numero <> Int( Numero ) Then
 CalcularFactorial = "N但ero no v疝ido para c疝culo del factorial!"
 ElseIf Numero = 0 Then
 CalcularFactorial = 1
 Else
 ' Esta es la llamada recursiva:
 CalcularFactorial = Numero * CalcularFactorial( Numero - 1 )
 Endif
End Function
```

El ejemplo devuelve el factorial del número `42` mediante la llamada recursiva a la función `CalcularFactorial` hasta que se alcanza la condición base de `0! = 1`.

Tenga en cuenta que el nivel de recursión de StarOffice Basic está limitado a 500.

Gestión de errores

La gestión correcta de las situaciones de error es una de las tareas más costosas de la programación desde el punto de vista del tiempo dedicado. StarOffice Basic ofrece una gama de herramientas para simplificar la gestión de errores.

La instrucción On Error

La instrucción `On Error` es la clave de cualquier sistema de gestión de errores:

```
Sub Prueba
      On Error Goto ManejadorError

      ' ... efectuar tarea durante la cual puede tener lugar un error

Exit Sub

ManejadorError:

      ' ... código individual para manejo de errores

End Sub
```

La línea `On Error Goto ManejadorError` define el comportamiento de StarOffice Basic en caso de error. `Goto ManejadorError` asegura que StarOffice Basic salga de la línea de programa actual y ejecute el código del `ManejadorError`: .

La orden Resume

La orden `Resume Next` prosigue el programa desde la línea siguiente a la línea en la que ocurrió el error, una vez ejecutado el código del manejador de error:

```
ManejadorError:

      ' ... código individual para manejo de errores

      Resume Next
```

Utilice la orden `Resume Proceed` para especificar un punto de salto para continuar con el programa después de la gestión del error:

```
ManejadorError:

      ' ... código individual para manejo de errores
       Resume Proceed

Proceed:

      ' ... el programa continúa a partir de aquí después del error
```

Para proseguir con un programa sin mostrar un mensaje de error cuando un error tiene lugar, utilice el formato siguiente:

```
Sub Prueba
      On Error Resume Next

      ' ... efectuar tarea durante la cual puede tener lugar un error

End Sub
```

Utilice la orden `On Error Resume Next` con precaución, ya que sus efectos son globales. Para obtener más información, consulte ***Consejos para el manejo estructurado de errores***.

Consultas acerca de la información de errores

En la gestión de errores resulta útil disponer de una descripción del error, así como del lugar y motivo por el que éste ha ocurrido:

- La variable `Err` contiene el número de errores ocurridos.
- La variable `Error$` contiene una descripción del error.
- La variable `Erl` contiene el número de línea en el que ha ocurrido el error.

La llamada

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

muestra la información acerca del error en una ventana de mensaje.

Mientras que VBA resume los mensajes de error en un objeto estadístico denominado `Err`, StarOffice Basic proporciona las variables `Err`, `Error$` y `Erl`.

La información de estado sigue siendo válida hasta que el programa llega a una orden `Resume` u `On Error`; en ese momento la información se reinicia.

En VBA, el método `Err.Clear` del objeto `Err` reinicia el estado de error después de un error. En StarOffice Basic se utilizan para ello las órdenes `On Error` o `Resume`.

Consejos para el manejo estructurado de errores

Tanto la orden de definición, `On Error`, como la orden de retorno, `Resume`, son variantes de la construcción `Goto`.

Si desea estructurar el código de forma clara para impedir que se generen errores al emplear esta construcción, deberá evitar utilizar órdenes de salto sin supervisarlas.

Debe utilizar la orden `On Error Resume Next` con precaución, ya que ésta desecha todos los mensajes de error abiertos.

La mejor solución es utilizar una única estrategia de gestión de errores en un programa, mantener el código de gestión de errores separado del propio código del programa y no volver al código original después de un error.

A continuación se muestra un ejemplo de un procedimiento de gestión de errores:

```
Sub Ejemplo

	' Definir el manejador de error al principio de la función
	On Error Goto ManejadorError

	' ... Código del programa en sí

	' Desactivar gestión de errores
	On Error Goto 0

	' Fin de la implementación del programa normal
Exit Sub

' Inicio de la gestión de errores
ManejadorError:

	' Comprobar si se esperaba un error
	If Err = NumErrorEsperado Then
		' ... Procesar error
	Else
		' ... Advertencia de error inesperado
	End If

	On Error Goto 0                              ' Desactivar gestión de errores
End Sub
```

Este procedimiento empieza por la definición de un manejador de error, seguido por el código del programa en sí. Al final del código del programa, la gestión de errores se desactiva mediante la llamada `On Error Goto 0` y la implementación del procedimiento finaliza mediante la orden `Exit Sub` (que no debe confundirse con `End Sub`).

El ejemplo comprueba en primer lugar si el número de error corresponde al número esperado (almacenado en la constante imaginaria `NumErrorEsperado`) y gestiona el error según corresponda. Si ocurre un error distinto, el sistema emite una advertencia. Es importante comprobar el número de error a fin de que se puedan detectar los errores no previstos.

La llamada `On Error Goto 0` al final del código reinicia la información de estado del error (el código del error en las variables de sistema `Err`) de forma que se pueda reconocer con claridad un error que tenga lugar en un momento posterior.

CAPÍTULO 3

La biblioteca de ejecución de StarOffice Basic

En las secciones siguientes se presentan las funciones principales de la biblioteca de ejecución.

Funciones de conversión

En diversas situaciones concurren circunstancias en las que una variable de un tipo debe convertirse en una variable de un tipo distinto.

Conversiones de tipos implícitas y explícitas

La forma más sencilla de cambiar el tipo de una variable es mediante una asignación.

```
Dim A As String
Dim B As Integer

B = 101
A = B
```

En este ejemplo, la variable A es una cadena y la variable B es un entero. StarOffice Basic convierte la variable B en cadena durante la asignación a la variable A. Esta conversión es mucho más compleja de lo que parece: el entero B permanece en la memoria de trabajo en forma de número de dos bytes de longitud; por otra parte, A es una cadena y el sistema guarda un valor de uno o dos bytes de longitud para cada carácter (cada número), por consiguiente, antes de copiar el contenido de B en A, es necesario convertir B al formato interno de A.

A diferencia de casi todos los demás lenguajes de programación, Basic efectúa la conversión de tipos de forma automática. No obstante, las consecuencias de dicha conversión pueden ser fatales. Al examinar detenidamente la siguiente secuencia de código

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

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

que parece bastante directa a primera vista, resulta ser más bien una trampa. El intérprete de Basic calcula primero el resultado del proceso de suma y a continuación efectúa la conversión de éste en una cadena, lo que produce como resultado la cadena 2. Si, en cambio, el intérprete de Basic

convierte primero los valores iniciales `B` y `C` en una cadena y aplica al resultado el operador de suma, se genera el resultado `11`.

Lo mismo sucede cuando se utilizan variables de tipo variant:

```
Dim A
Dim B
Dim C

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

Puesto que las variables de tipo variant pueden contener tanto números como cadenas, no está claro si se asigna a la variable A el número 2 o la cadena 11.

Las fuentes de error indicadas para las conversiones de tipo implícitas sólo pueden evitarse mediante una disciplina de programación adecuada; evitando, por ejemplo, el uso del tipo de datos variant (una recomendación en la insistimos).

Para evitar otro tipo de errores provocados por la conversión de tipos implícita, StarOffice Basic ofrece una amplia gama de funciones de conversión que se pueden utilizar para definir cuándo se debe convertir el tipo de datos de una operación:

- **`CStr(Var)`** : convierte cualquier tipo de datos en una cadena.
- **`CInt(Var)`** : convierte cualquier tipo de datos en un valor entero.
- **`CLng(Var)`** : convierte cualquier tipo de datos en un valor entero largo.
- **`CSng(Var)`** : convierte cualquier tipo de datos en un valor de coma flotante de precisión simple.
- **`CDbl(Var)`** : convierte cualquier tipo de datos en un valor de coma flotante de precisión doble.
- **`CBool(Var)`** : convierte cualquier tipo de datos en un valor booleano.
- **`CDate(Var)`** : convierte cualquier tipo de datos en un valor de fecha.

Estas funciones de conversión se pueden utilizar para definir cómo StarOffice Basic efectúa esas operaciones de conversión de tipo:

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

B = 1
C = 1

A = CInt(B + C)             ' B y C se suman en primer lugar y a continuación se
convierten
                              (se genera el número 2)
A = CStr(B) + Cstr(C)       ' B y C se convierten al tipo cadena y a continuación
                            ' se combinan (se genera la cadena "11")
```

Durante la primera suma del ejemplo, StarOffice Basic en primer lugar suma las variables enteras y, a continuación, convierte el resultado en una cadena de caracteres. Se asigna a A la cadena 2. En el segundo ejemplo, las variables enteras se convierten primero en dos cadenas y a continuación se combinan entre sí mediante la asignación. Se asigna a A la cadena 11.

Las funciones de conversión numéricas CSng y CDbl aceptan también como argumentos números decimales. Se debe utilizar como signo decimal el símbolo definido en la configuración específica de cada país. De forma similar, el método CStr utiliza la configuración específica de país actualmente seleccionada para dar formato a números, fechas y horas.

La función Val es distinta de los métodos Csng, Cdbl y Cstr. Esta función convierte una cadena en un número; pero, siempre espera un punto como símbolo de separación de decimales.

```
Dim A As String
Dim B As Double

A = "2.22"
B = Val(A)        '  La conversión es correcta independientemente de la configuración
específica del país
```

Comprobación del contenido de variables

En algunos casos, una fecha no puede convertirse:

```
Dim A As String
Dim B As Date

A = "prueba"
B = A                   ' Crea un mensaje de error
```

En este ejemplo, la asignación de la cadena prueba a una variable de fecha no tiene sentido, por lo que el intérprete de Basic genera un error. Es similar al caso de intentar asignar una cadena a una variable booleana:

```
Dim A As String
Dim B As Boolean

A = "prueba"
B = A                   ' Crea un mensaje de error
```

De nuevo, el intérprete de Basic genera un error.

Estos mensajes de error se pueden evitar efectuando una comprobación en el programa antes de la asignación, para de este modo asegurarse de que el contenido de la variable se ajuste al tipo de la variable de destino. StarOffice Basic proporciona para ello las siguientes funciones de comprobación:

- **IsNumeric(Valor)** : comprueba si un valor es un número.
- **IsDate(Valor)** : comprueba si un valor es una fecha.
- **IsArray(Valor)** : comprueba si un valor es una matriz.

Estas funciones son particularmente útiles para solicitar entradas de datos al usuario. Por ejemplo, se puede comprobar si un usuario ha introducido un número o una fecha válidos.

```
If IsNumeric(EntradaUsuario) Then
      EntradaValida = EntradaUsuario
Else
      EntradaValida = 0
      MsgBox "Mensaje de error."
End If
```

En el ejemplo anterior, si la variable `EntradaUsuario` contiene un valor numérico válido, se asignará a la variable `EntradaValida`. Si `EntradaUsuario` no contiene un número válido, se asignará a `EntradaValida` el valor `0` y se devolverá un mensaje de error.

Basic proporciona funciones de comprobación para números, fechas y matrices, pero no para valores booleanos. No obstante, se puede imitar dicha funcionalidad mediante la función `IsBoolean`:

```
Function IsBoolean(Valor As Variant) As Boolean
      On Error Goto ErrorEsBooleano:
      Dim Dummy As Boolean

      Dummy = Valor

      IsBoolean = True
      On Error Goto 0
Exit Sub

ErrorEsBooleano:
      IsBoolean = False
      On Error Goto 0
End Function
```

La función `IsBoolean` define un variable intermedia interna `Dummy` de tipo booleano e intenta asignarle el valor transferido. Si la asignación es posible, la función devuelve `True`. En caso contrario se genera un error de ejecución que la función de prueba intercepta para devolver un error.

En StarOffice Basic, si una cadena contiene un valor no numérico y se asigna a un número, no se genera un mensaje de error, sino que se transfiere a la variable el valor `0`. Este procedimiento es distinto en VBA. En VBA, la ejecución de una asignación de este tipo activa un error y la implementación del programa finaliza.

Cadenas

Trabajo con juegos de caracteres

StarOffice Basic utiliza para la gestión de cadenas el juego de caracteres Unicode. Las funciones `Asc` y `Chr` permiten establecer el valor Unicode de un carácter y, también, encontrar el que corresponda a un valor Unicode. En las expresiones siguientes se asignan diversos valores Unicode a la variable `Codigo`:

```
Codigo = Asc("A")           ' Letra latina A (valor Unicode 65)
Código = Asc("?")           ' Carácter de euro (valor Unicode 8364)
Codigo = Asc(""|")          ' Letra cirílica "|(valor Unicode 1083)
```

A su vez, la expresión

```
MiCadena = Chr(13)
```

garantiza que la cadena `MiCadena` se inicializará con el valor del número `13` que corresponde a un salto de línea forzado.

La orden `Chr` se suele utilizar en los lenguajes Basic para insertar caracteres de control en una cadena. La asignación

```
MiCadena = Chr(9) + "Esto es una prueba" + Chr(13)
```

antepone al texto un carácter de tabulador (valor Unicode `9`) y pospone un salto de línea forzado (valor Unicode `13`).

Acceso a partes de una cadena

StarOffice Basic ofrece cuatro funciones que devuelven cadenas parciales:

- **`Left(MiCadena, Longitud)`** : devuelve los primeros caracteres de `Longitud` de `MiCadena`.
- **`Right(MiCadena, Longitud)`** : devuelve los últimos caracteres de `Longitud` de `MiCadena`.
- **`Mid(MiCadena, Inicio, Longitud)`** : devuelve los primeros caracteres de `Longitud` de `MiCadena` a partir de la posición `Inicio`.
- **`Len(MiCadena)`** : devuelve el número de caracteres de `MiCadena`.

A continuación se incluyen varios ejemplos de llamadas a las funciones mencionadas:

```
Dim MiCadena As String
Dim MiResultado As Strings
Dim MiLongitud As Integer

MiCadena = "Esto es una pequeña prueba"

MiResultado = Left(MiCadena,5)             ' devuelve la cadena "Esto e"
```

```
MiResultado = Right(MiCadena, 5)    ' devuelve la cadena "ueba"
MiResultado = Mid(MiCadena, 8, 5)   ' devuelve la cadena "una p"
MiLongitud = Len(MiCadena)          ' Devuelve el valor 4
```

Buscar y reemplazar

StarOffice Basic ofrece la función `InStr` para buscar una cadena parcial contenida dentro de otra cadena:

```
Cadena resultado = InStr (Cadena búsqueda, MiCadena)
```

El parámetro `Cadena búsqueda` especifica la cadena que se debe buscar dentro de `MiCadena`. La función devuelve un número que contiene la posición en la que aparece por primera vez `Cadena búsqueda` dentro de `MiCadena`. Si desea buscar otras apariciones de la cadena, la función ofrece también la oportunidad de especificar opcionalmente una posición inicial desde la que StarOffice Basic comienza la búsqueda. En este caso, la sintaxis de la función es:

```
Cadena resultado = InStr (PosicionInicial, Cadena búsqueda, MiCadena)
```

En los ejemplos anteriores, `InStr` no distingue entre caracteres en mayúscula y en minúscula. Para cambiar el método de búsqueda para que `InStr` distinga entre mayúsculas y minúsculas, agregue el parámetro `0`, como se muestra en el ejemplo siguiente:

```
Cadena resultado = InStr (Cadena búsqueda, MiCadena, 0)
```

Mediante las funciones de cadena mencionadas, los programadores pueden buscar y reemplazar una cadena dentro de otra:

```
Function Reemplazar(Origen As String, Busqueda As String, ParteNueva As String)
  Dim Resultado As String
  Dim PosInicial As Long
  Dim PosActual As Long

  Resultado = ""
  PosInicial = 1
  PosActual = 1

  If Busqueda = "" Then
        Resultado = Origen
    Else
        Do While PosActual <> 0
            PosActual = InStr(PosInicial, Origen, Busqueda)
            If PosActual <> 0 Then
                Resultado = Resultado + Mid(Origen, PosInicial, _
                        PosActual - PosInicial)
                Resultado = Resultado + ParteNueva
                PosInicial = PosActual + Len(Busqueda)
            Else
                Resultado = Resultado + Mid(Origen, PosInicial, Len(Origen))
            End If ' Posición <> 0
        Loop
```

```
    End If
    Reemplazar = Resultado
End Function
```

La función busca la cadena `Busqueda` indicada en un bucle mediante el uso de `InStr` en la cadena `Origen`. Si encuentra el término buscado, toma la parte de la cadena anterior a la expresión y la escribe en la memoria intermedia de retorno `Resultado`. Agrega la sección `ParteNueva` en el punto del término `Busqueda`. Si no se encuentran más apariciones del término buscado, la función halla la parte restante de la cadena y la agrega a la memoria intermedia de retorno. Finalmente, la función devuelve la cadena así generada como resultado del proceso de sustitución.

La sustitución de partes de secuencias de caracteres es una de las funciones utilizadas con más frecuencia, por lo que en StarOffice Basic se ha ampliado la función `Mid` para que efectúe dicha tarea de forma automática. El ejemplo siguiente

```
Dim MiCadena As String

MiCadena  = "Éste era mi texto"
Mid(MiCadena, 6, 3, "es")
```

reemplaza tres caracteres por la cadena `es` a partir de la sexta posición de la cadena `MiCadena`.

Formato de cadenas

La función `Format` formatea números como cadenas. Para ello, la función toma como parámetro una expresión `Format` y la utiliza como plantilla para dar formato a los números. Cada uno de los comodines de la plantilla garantiza que el elemento correspondiente adquirirá el formato adecuado en el valor de salida. Los cinco comodines más importantes en una plantilla son el ***cero*** (0), el ***signo de almohadilla*** (#), el ***punto*** (.), la ***coma*** (,) y ***el signo de dólar*** ($).

En una plantilla, el carácter ***cero*** garantiza que un número se sitúe siempre en el punto correspondiente. Si no se proporciona un número, en su lugar aparece 0.

El ***punto*** simboliza el signo de decimales definido por el sistema operativo en la configuración regional.

En el ejemplo siguiente se muestra de qué forma los caracteres ***cero*** y ***punto*** pueden definir los dígitos que aparecen después del signo decimal en una expresión:

```
MiFormato = "0.00"

MiCadena = Format(-1579.8, MiFormato)     ' Devuelve "-1579,80"
MiCadena = Format(1579.8, MiFormato)           ' Devuelve "1579,80"
MiCadena = Format(0.4, MiFormato)         ' Devuelve "0,40"
MiCadena = Format(0.434, MiFormato)       ' Devuelve "0,43"
```

De forma similar, se pueden agregar ceros precediendo a un número hasta adquirir la longitud deseada:

```
MiFormato = "0000.00"
```

```
MiCadena = Format(-1579,80, MiFormato)    ' Devuelve "-1.579,80"
MiCadena = Format(1579.8, MiFormato)           ' Devuelve "1579,80"
MiCadena = Format(0.4, MiFormato)         ' Devuelve "0000,40"
MiCadena = Format(0.434, MiFormato)       ' Devuelve "0000,43"
```

La ***coma*** representa el carácter utilizado por el sistema operativo como separador de miles y el ***signo de almohadilla*** corresponde a un dígito o lugar que sólo se muestra si la cadena de entrada lo requiere.

```
MiFormato = "#,##0.00"

MiCadena = Format(-1579.8, MiFormato)     ' Devuelve "-1.579,80"
MiCadena = Format(1579.8, MiFormato)           ' Devuelve "1.579,80"
MiCadena = Format(0.4, MiFormato)         ' Devuelve "0,40"
MiCadena = Format(0.434, MiFormato)       ' Devuelve "0,43"
```

En el lugar del comodín ***signo de dólar***, la función `Format` muestra el símbolo de moneda correspondiente definido por el sistema:

```
MiFormato = "#,##0.00 $"

MiCadena = Format(-1579.8, MiFormato)     ' Devuelve "-1.579,80 ?"
MiCadena = Format(1579.8, MiFormato)           ' Devuelve "1.579,80 ?"
MiCadena = Format(0.4, MiFormato)         ' Devuelve "0,40 ?"
MiCadena = Format(0.434, MiFormato)       ' Devuelve "0,43 ?"
```

Las instrucciones de formato que VBA utiliza para fechas y horas no se admiten en StarOffice Basic.

Fechas y horas

StarOffice Basic ofrece el tipo de datos `Date` , que guarda los detalles de fecha y hora en formato binario.

Especificación de detalles de fecha y hora dentro del código de programa

Se puede asignar una fecha a una variable de fecha mediante la asignación de una cadena simple:

```
Dim MiFecha As Date

MiFecha = "1.1.2002"
```

Esta asignación funciona correctamente porque StarOffice Basic convierte automáticamente el valor de fecha definido como cadena en una variable de fecha. No obstante, este tipo de asignación puede provocar errores, ya que los valores de fecha y hora se definen y muestran de forma distinta en cada país.

StarOffice Basic utiliza la configuración específica del país definida en el sistema operativo para la conversión de una cadena en un valor de fecha, por lo que la expresión indicada sólo funciona correctamente en el caso de que dicha configuración se ajuste a la expresión de la cadena.

```
Para evitar este problema, es conveniente utilizar la función
DateSerial que asigna un valor fijo a una variable de fecha:Dim MiVariable
As Date

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

Los parámetros de la función deben estar en el orden: año, mes, día. La función garantiza que la asignación es el valor correcto, independientemente de la configuración específica del país

La función `TimeSerial` formatea las horas igual que la función `DateSerial` las fechas:

```
Dim MiVariable As Date

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

Los parámetros deben especificarse en el orden: horas, minutos, segundos.

Extracción de detalles de fecha y hora

Las funciones siguientes constituyen el complemento de las funciones `DateSerial` y `TimeSerial`:

- **`Day(MiFecha)`**: devuelve el día del mes de `MiFecha`
- **`Month(MiFecha)`**: devuelve el mes de `MiFecha`
- **`Year(MiFecha)`**: devuelve el año de `MiFecha`
- **`WeekDay(MiFecha)`**: devuelve el número del día de la semana de `MiFecha`
- **`Hora(MiHora)`**: devuelve las horas de `MiHora`
- **`Minute(MiHora)`**: devuelve los minutos de `MiHora`
- **`Second(MiHora)`**: devuelve los segundos de `MiHora`

Estas. funciones extraen las secciones de fecha u hora de una variable de tipo `Date` especificada. El ejemplo

```
Dim MiFecha As Date

' ... Inicialización de MiFecha

If Year(MiFecha) = 2003 Then

      ' ... La fecha especificada pertenece al año 2003
End If
```

comprueba si la fecha guardada en la variable `MiFecha` corresponde al año 2003.

De forma similar, el ejemplo

```
Dim MiHora As Date

' ... Inicialización de MiHora

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

      ' ... La fecha especificada pertenece al año 2003
End If
```

comprueba si `MiHora` está entre las `12` y las `14` horas.

La función `Weekday` devuelve el número del día de la semana correspondiente a la fecha indicada:

```
Dim MiFecha As Date
Dim MiDiaDeLaSemana As String

' ... inicializar MiFecha

Select Case WeekDay(MiFecha)
case 1
      MiDiaDeLaSemana = "Domingo"
case 2
      MiDiaDeLaSemana = "Lunes"
case 3
      MiDiaDeLaSemana = "Martes"
case 4
      MiDiaDeLaSemana = "Miércoles"
case 5
      MiDiaDeLaSemana = "Jueves"
case 6
      MiDiaDeLaSemana = "Viernes"
case 7
      MiDiaDeLaSemana = "Sábado"
End Select
```

Nota: El domingo se considera el primer día de la semana.

Recuperación de la fecha y hora del sistema

StarOffice Basic dispone de las siguientes funciones para recuperar la hora y la fecha del sistema:

- **`Date`**: devuelve la fecha actual
- **`Time`**: devuelve la hora actual
- **`Now`**: devuelve el instante temporal actual (valor combinado de fecha y hora)

Archivos y directorios

El trabajo con archivos representa una de las tareas básicas de una aplicación. La API de StarOffice ofrece una amplia gama de objetos que permiten crear, abrir y modificar documentos de Office. Dichas funciones se comentan en detalle en el capítulo 4. A pesar de ello, en ciertos casos se deberá

acceder directamente al sistema de archivos, buscar en directorios o editar archivos de texto. La biblioteca de ejecución de StarOffice Basic ofrece varias funciones fundamentales para efectuar dichas tareas.

StarOffice 6.x ha dejado de incorporar algunas funciones de archivos y directorios específicas de DOS o las ha limitado. Por ejemplo, no se admiten las funciones `ChDir`, `ChDrive` y `CurDir`. Ciertas propiedades específicas de DOS ya no se utilizan en las funciones que aceptan como parámetros propiedades de archivos (por ejemplo, para diferenciar entre archivos ocultos y archivos de sistema). Este cambio ha sido necesario para garantizar la máxima independencia de la plataforma de StarOffice.

Administración de archivos

Búsqueda en directorios

En StarOffice Basic, la función `Dir` es responsable de buscar archivos y subdirectorios dentro de directorios. La primera vez que se solicita, es necesario asignarle como primer parámetro una cadena con la ruta de los directorios en los que se debe buscar. El segundo parámetro de `Dir` especifica el archivo o directorio que se debe buscar. StarOffice Basic devuelve el nombre de la primera entrada de directorio encontrada. Para recuperar la siguiente entrada deberá llamar a la función `Dir` sin parámetros. Si la función `Dir` no encuentra más entradas, devolverá una cadena vacía.

En el ejemplo siguiente se muestra cómo se puede utilizar la función `Dir` para solicitar todos los archivos contenidos en un directorio. El procedimiento guarda los nombres de archivo individuales en la variable `TodosLosArchivos` y luego la muestra en un cuadro de mensaje.

```
Sub MostrarArchivos
      Dim SiguienteArchivo As String
      Dim TodosLosArchivos As String

      TodosLosArchivos = ""
      SiguienteArchivo = Dir("C:\", 0)

      While SiguienteArchivo  <> ""
            TodosLosArchivos = TodosLosArchivos & Chr(13) &  SiguienteArchivo
            SiguienteArchivo  = Dir
      Wend

      MsgBox TodosLosArchivos
End Sub
```

El `0` utilizado como segundo parámetro de la función `Dir` garantiza que ésta devolverá únicamente nombres de archivo e ignorará los directorios. Se pueden especificar los parámetros siguientes:

- `0` : devuelve archivos normales
- `16` : subdirectorios

El ejemplo siguiente es prácticamente idéntico al anterior, pero la función `Dir` recibe como parámetro el valor 16 que indica que devuelva los directorios de una carpeta en lugar de los nombres de archivo.

```
Sub MostrarDirectorios
      Dim SiguienteDirectorio As String
      Dim TodosLosDirectorios As String
      TodosLosDirectorios = ""
      SiguienteDirectorio = Dir("C:\", 16)
      While SiguienteDirectorio <> ""
            TodosLosDirectorios = TodosLosDirectorios & Chr(13) &  SiguienteDirectorio
            SiguienteDirectorio  = Dir
      Wend
      MsgBox TodosLosDirectorios
End Sub
```

A diferencia de VBA, al llamar a la función `Dir` en StarOffice Basic con el parámetro `16`, dicha función devuelve únicamente los subdirectorios de una carpeta. (En VBA, la función devuelve también los nombres de los archivos estándar, por lo que es necesario una comprobación posterior para recuperar únicamente los directorios).

Las opciones que VBA ofrece para buscar específicamente archivos con las propiedades *oculto*, *sistema, archivo* y *nombre de volumen* no existe en StarOffice Basic, debido a que las funciones de sistemas de archivos correspondientes no están disponibles en todos los sistemas operativos.

En las especificaciones de ruta indicadas en `Dir` se pueden utilizar los comodines `*` y `?`, tanto en VBA como en StarOffice Basic. No obstante, en StarOffice Basic, el comodín `*` sólo puede ser el último carácter de un nombre de archivo o de una extensión; éste no es el caso en VBA.

Creación y supresión de directorios

StarOffice Basic ofrece la función `MkDir` para crear directorios.

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

Esta función crea directorios y subdirectorios. Si es necesario se pueden crear todos los directorios dentro de una jerarquía. Por ejemplo, si sólo existe el directorio `C:\SubDir1`, la llamada

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

crea tanto el directorio `C:\SubDir1\SubDir2` como el `C:\SubDir1\SubDir2\SubDir3`.

La función `RmDir` se utiliza para borrar directorios.

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

Si el directorio contiene subdirectorios o archivos, ***éstos se borrarán también.*** Tenga cuidado al utilizar `RmDir`.

En VBA, las funciones `MkDir` y `RmDir` sólo hacen referencia al directorio actual. En cambio, en StarOffice Basic `MkDir` y `RmDir` se pueden utilizar para crear o borrar niveles enteros de directorios.

En VBA, `RmDir` genera un mensaje de error en el caso de que el directorio contenga un archivo. En StarOffice Basic se borran el directorio ***y todos los archivos*** que contenga.

Copia, cambio de nombre, supresión y comprobación de la existencia de archivos

La llamada

```
FileCopy(Origen, Destino)
```

crea una copia del archivo `Origen` con el nombre `Destino`.

Con la ayuda de la función

```
Name NombreAntiguo As NombreNuevo
```

se puede cambiar el nombre del archivo `NombreAntiguo` por `NombreNuevo`. La sintaxis de la palabra clave `As`, así como el hecho de que no se utilice una coma, están relacionados con los fundamentos del lenguaje Basic.

La llamada

```
Kill(NombreArchivo)
```

borra el archivo `NombreArchivo`. Si desea borrar todo un directorio (incluidos sus archivos) utilice la función `RmDir`.

La función `FileExists` se utiliza para comprobar si un archivo existe:

```
If FileExists(NombreArchivo) Then
      MsgBox "el archivo existe."
End If
```

Lectura y cambio de las propiedades del archivo

Cuando se trabaja con archivos, a veces es necesario poder establecer las propiedades de éstos, la hora en la que se modificaron por última vez, la longitud.

La llamada

```
Dim Atrib As Integer
Atrib = GetAttr(NombreArchivo)
```

devuelve algunas de las propiedades de un archivo. El valor de retorno se devuelve en forma de máscara de bits; los valores posibles son los siguientes:

- 1 : archivo de sólo lectura

16 : nombre de un directorio

El ejemplo

```
Dim MascaraArchivo As Integer
Dim DescripcionArchivo As String

MascaraArchivo = GetAttr("prueba.txt")

If (MascaraArchivo AND 1) > 0 Then
      DescripcionArchivo = DescripcionArchivo & " sólo lectura "
End IF

If (MascaraArchivo AND 16) > 0 Then
      DescripcionArchivo = DescripcionArchivo & " directorio "
End IF

If DescripcionArchivo = "" Then
      DescripcionArchivo = " normal "
End IF

MsgBox DescripcionArchivo
```

determina la máscara de bits del archivo `prueba.txt` y comprueba si se trata de un archivo de sólo lectura o de un directorio. Si no es ninguno de estos dos casos, se asigna a la variable `DescripcionArchivo` la cadena "normal".

Los indicadores que se utilizan en VBA para consultar las propiedades del archivo, *oculto*, *sistema, archivo* y *nombre de volumen*, no se admiten en StarOffice Basic, ya que son específicos de Windows y no están disponibles en otros sistemas operativos, o lo están sólo de modo parcial.

La función `SetAttr` permite modificar las propiedades de un archivo. La llamada

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

se puede utilizar para asignar a un archivo el estado de sólo lectura. Si un archivo es de sólo lectura, se puede suprimir dicho estado mediante la llamada:

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

La fecha y la hora de la última modificación efectuada en un archivo se obtienen mediante la función `FileDateTime`. El formato de la fecha responde a la configuración específica del país vigente en el sistema.

```
FileDateTime("prueba.txt")    ' devuelve la fecha y hora de la última modificación
efectuada al archivo.
```

La función `FileLen` determina la longitud de un archivo en bytes (en forma de un valor de tipo entero largo).

FileLen("prueba.txt") ' devuelve la longitud del archivo en bytes

Escritura y lectura de archivos de texto

StarOffice Basic ofrece una gran variedad de métodos para leer y escribir archivos. Las siguientes indicaciones se refieren al trabajo con archivos de texto (***no*** con documentos de texto).

Escritura de archivos de texto

Antes de poder acceder a un archivo de texto, es necesario abrirlo. Para ello se necesita un ***manejador de archivo*** libre; dicho manejador identifica claramente el archivo para los subsiguientes accesos.

La función `FreeFile` se utiliza para crear un manejador de archivo libre que se utiliza como parámetro para la instrucción `Open` que abre el archivo. Para abrir un archivo de forma que se pueda especificar como archivo de texto, la llamada a `Open` es la siguiente:

```
Open NombreArchivo For Output As #NumArchivo
```

`NombreArchivo` es una cadena que contiene el nombre del archivo. `NumArchivo` es el manejador creado por la función `FreeFile`.

Una vez abierto el archivo, la instrucción `Print` puede describirse línea por línea:

```
Print #NumArchivo, "Esta es una línea de prueba."
```

`NumArchivo` hace también referencia al manejador de archivo. El segundo parámetro especifica el texto que debe guardarse como línea del archivo de texto.

Una vez completado el proceso de escritura, se debe cerrar el archivo mediante una llamada a `Close`:

```
Close #NumArchivo
```

De nuevo, es necesario especificar el manejador de archivo.

En el ejemplo siguiente se muestra cómo se abre, describe y cierra un archivo de texto:

```
Dim NumArchivo As Integer
Dim LineaActual As String
Dim NombreArchivo As String

NombreArchivo = "c:\datos.txt"                          ' Definir el nombre del archivo
NumArchivo = FreeFile                                   ' Establecer un manejador de
archivo libre

Open NombreArchivo For Output As #NumArchivo                    ' Abrir el archivo en
modo de escritura
Print #NumArchivo, "Esta es una línea de texto"          ' Guardar línea
Print #NumArchivo, "Esta es otra línea de texto"   ' Guardar línea
Close #NumArchivo                                       ' Cerrar archivo
```

Lectura de archivos de texto

Los archivos de texto se leen de la misma forma que se escriben. La instrucción `Open` que se utiliza para abrir el archivo contiene la expresión `For Input` en lugar de `For Output` y, en lugar de la orden `Print`, para escribir datos se debe utilizar la orden `Line Input` para leerlos.

Finalmente, cuando se llama a un archivo de texto, la instrucción

```
eof(NumArchivo)
```

se utiliza para comprobar que se ha llegado al final del archivo.

En el ejemplo siguiente se muestra el procedimiento para leer un archivo de texto:

```
Dim NumArchivo As Integer
Dim LineaActual As String
Dim Archivo As String
Dim Mensaje as String

' Definir nombre de archivo
NombreArchivo = "c:\datos.txt"

' Establecer un manejador de archivo libre
NumArchivo = Freefile

' Abrir el archivo (en modo de lectura)
Open NombreArchivo For Input As NumArchivo

' Comprobar si se ha llegado al final del archivo
Do While not eof(NumArchivo)

      ' Leer línea
      Line Input #NumArchivo, LineaActual
      If LineaActual <>"" then
            Mensaje = Mensaje & LineaActual & Chr(13)
      end if

Loop

' Cerrar archivo
Close #NumArchivo

Msgbox Mensaje
```

Las líneas individuales se recuperan en un bucle `Do While`, se guardan en la variable `Mensaje` y se muestran al final en un cuadro de mensaje.

Cuadros de mensaje y de entrada

StarOffice Basic ofrece las funciones `MsgBox` y `InputBox` para comunicarse con el usuario de forma sencilla.

Presentación de mensajes

`MsgBox` muestra un cuadro de información simple con uno o más botones. En su variante más sencilla

```
MsgBox "Esto es un fragmento de información."
```

el cuadro `MsgBox` contiene simplemente texto y el botón Aceptar.

El aspecto del cuadro de información se puede modificar mediante un parámetro. Éste permite agregar botones adicionales, definir el botón preasignado y añadir un símbolo de información. Los valores de selección de botones son:

- 0: botón Aceptar
- 1: botones Aceptar y Cancelar
- 2: botones Cancelar y Reintentar
- 3: botones Sí, No y Cancelar
- 4: botones Sí y No
- 5: botones Reintentar y Cancelar

Para establecer un botón como predeterminado, sume uno de los valores siguientes al valor del parámetro de la lista de selecciones de botones. Por ejemplo, para crear los botones Sí, No y Cancelar (valor 3) y que Cancelar sea el botón predeterminado (valor 512), el valor del parámetro deberá ser 3 + 512 = 515.

- 0: el primer botón es el predeterminado
- 256: el segundo botón es el predeterminado
- 512: el tercer botón es el predeterminado

Finalmente, se dispone de los siguientes símbolos de información que pueden mostrarse sumando los valores relevantes:

- 16: signo de stop
- 32: signo de interrogación
- 48: signo de admiración
- 64: símbolo de consejo

La llamada

```
MsgBox "¿Desea continuar?",  292
```

muestra un cuadro de información con los botones Sí y no (valor 4), siendo el segundo de ellos (no) el valor predeterminado (valor 256); el cuadro contiene además un signo de interrogación (valor 32), 4+256+32=292

Si un cuadro de información contiene varios botones, se debe consultar el valor de retorno para determinar el botón pulsado. Los valores de retorno disponibles en este caso son los siguientes:

- 1: Aceptar
- 2: Cancelar
- 4: Reintentar
- 5: Ignorar
- 6: Sí
- 7: No

En el ejemplo anterior, la comprobación del valor de retorno puede tener el aspecto siguiente:

```
If MsgBox ("¿Desea continuar?",  292) = 6 Then
      ' Se ha pulsado el botón Sí
Else
      ' Se ha pulsado el botón No
End IF
```

Aparte del texto de información y del parámetro para organizar el cuadro de información, `MsgBox` permite especificar un tercer parámetro que define el texto del título del cuadro:

```
MsgBox "¿Desea continuar?",  292, "Título de la ventana"
```

Si no se especifica ningún título, el título predeterminado es •gsoffice•h.

Cuadro de entrada para solicitar cadenas de texto sencillas

La función `InputBox` solicita al usuario cadenas de caracteres sencillas. Se trata de una alternativa simple para la configuración de diálogos. `InputBox` recibe tres parámetros estándar:

- un texto de información,
- un título para el cuadro,
- un valor predeterminado que se puede mostrar en el área de entrada.

```
ValEntrada = InputBox("Introduzca un valor:", "Prueba", "valor predeterminado")
```

Como valor de retorno, `InputBox` ofrece la cadena que el usuario ha introducido.

Otras funciones

Beep

La función `Beep` hace que el sistema emita un sonido que se puede utilizar para advertir al usuario de una acción incorrecta. `Beep` no tiene ningún parámetro:

```
Beep  ' emite un tono informativo
```

Shell

Los programas externos se pueden ejecutar mediante la función `Shell`.

```
Shell(NombreRuta, EstiloVentana, Parametros)
```

`NombreRuta` define la ruta del programa que se debe ejecutar. `EstiloVentana` define la ventana en la que se inicia el programa. Se pueden especificar los valores siguientes:

- 0: el programa recibe el foco y se inicia en una ventana oculta.
- 1: el programa recibe el foco y se inicia en una ventana de tamaño normal.
- 2: el programa recibe el foco y se inicia en una ventana minimizada.
- 3: el programa recibe el foco y se inicia en una ventana maximizada.
- 4: el programa se inicia en una ventana de tamaño normal, sin recibir el foco.
- 6: El programa se inicia en una ventana minimizada; el foco permanece en la ventana actual.
- 10: el programa se inicia en modo de pantalla completa.

El tercer parámetro, `Parametros,` permite transmitir parámetros de línea de órdenes al programa que se va a iniciar.

Wait

La función `Wait` interrumpe la ejecución del programa durante un tiempo especificado. El período de espera se debe indicar en milisegundos. La orden

```
Wait 2000
```

especifica una interrupción de 2 segundos (2000 milisegundos).

Environ

La función `Environ` devuelve las variables de entorno del sistema operativo. En función del sistema y de la configuración, dichas variables almacenan diversos tipos de datos. La llamada

```
Dim DirTemporal

DirTemporal=Environ ("TEMP")
```

determina las variables de entorno del directorio temporal del sistema operativo.

CAPÍTULO 4

Introducción a la API de StarOffice

La API de StarOffice es una interfaz de programación universal para acceder a StarOffice que se puede utilizar para crear, abrir, modificar e imprimir documentos de StarOffice. Ofrece la opción de ampliar el ámbito funcional de StarOffice mediante macros personales, así como escribir cuadros de diálogo personalizados.

La API de StarOffice no sólo se puede utilizar con StarOffice Basic; otros lenguajes de programación como Java o C++ pueden también emplearla. Esto es posible gracias a una técnica denominada UNO (***Universal Network Objects***, Objetos de red universales), que ofrece una interfaz para diversos lenguajes de programación.

En este capítulo se explica el uso de la API de StarOffice Basic en StarOffice Basic con la ayuda de UNO y se describen los conceptos principales de UNO desde el punto de vista del programador de StarOffice Basic. En los siguientes capítulos se detalla el trabajo con las diversas partes de la API de StarOffice.

Objetos de red universales (UNO)

StarOffice ofrece una interfaz de programación, UNO (Objetos de red universales), una interfaz orientada a objeto que StarOffice subdivide en diversos objetos que ,a su vez, garantizan el acceso controlado mediante programa al paquete StarOffice.

Puesto que StarOffice Basic es un lenguaje de programación por procedimientos, ha sido necesario agregarle diversas construcciones lingüísticas que hagan posible el uso de UNO.

Para poder utilizar un Objeto de red universal en StarOffice Basic, se necesita una declaración de variables para el objeto asociado; ésta se efectúa mediante la instrucción `Dim` (consulte el capítulo 2). Se debe utilizar la designación de tipo `Object` para declarar una variable de objeto:

```
Dim Obj As Object
```

La llamada declara una variable de objeto denominada `Obj`.

La variable de objeto creada se debe inicializar para poder utilizarla. Para ello se emplea la función `createUnoService`:

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

Esta llamada asigna a la variable `Obj` una referencia al objeto recién creado. `com.sun.star.frame.Desktop` parece un tipo de objeto; sin embargo, en la terminología de UNO, no se denomina tipo sino ***•gservicio•h***. De acuerdo con la filosofía de UNO, `Obj` se describe

como ***una referencia a un objeto que admite el servicio*** `com.sun.star.frame.Desktop`. El término •gservicio•h utilizado en StarOffice Basic se corresponde por tanto con los términos ***tipo*** y ***clase*** empleados en otros lenguajes de programación.

No obstante, hay una diferencia importante: un Objeto de red universal puede admitir varios servicios simultáneamente que, a su vez, pueden admitir otros servicios de forma que un único objeto proporciona una amplia gama de servicios. Por ejemplo, es posible que el objeto mencionado anteriormente, basado en el servicio `com.sun.star.frame.Desktop`, incluya también otros servicios para cargar documentos o finalizar un programa.

Mientras que la estructura de un objeto en VBA la define la clase a la que pertenece, en StarOffice Basic son los servicios que el objeto admite los que definen dicha estructura. Un objeto de VBA tiene asignada siempre una única clase. En cambio, un objeto de StarOffice Basic puede admitir varios servicios.

Propiedades y métodos

Un objeto de StarOffice Basic ofrece una gama de propiedades y métodos que se pueden llamar a través del objeto.

Propiedades

Las ***propiedades*** son como las propiedades de un objeto; por ejemplo, `Filename` (nombre de archivo) y `Title` (título) para un objeto `Document`.

Las propiedades se establecen mediante simples asignaciones:

```
Document.Title = "Manual de programación de StarOffice 6.0"
Document.Filename = "progman.sxv"
```

Una propiedad, igual que una variable normal, tiene un tipo que define los valores que puede almacenar. Las propiedades `Filename` y `Title` indicadas son de tipo cadena de caracteres.

Propiedades reales y propiedades imitadas

Casi todas las propiedades de un objeto en StarOffice Basic se definen como tales en la descripción UNO del servicio. Aparte de estas propiedades "reales", en StarOffice Basic hay también propiedades que constan de dos métodos en el nivel de UNO que se utilizan, respectivamente, para consultar el valor de la propiedad y para establecer ésta (métodos `get` y `set`). Una propiedad se ha imitado de forma virtual a partir de dos métodos. Los objetos de carácter en UNO, por ejemplo, ofrecen los métodos `getPosition` y `setPosition` a través de los cuales se puede consultar y modificar el punto asociado. El programador de StarOffice Basic puede acceder a los valores a través de la propiedad `Position`. Independientemente de ello, también están disponibles los métodos originales (en nuestro ejemplo, `getPosition` y `setPosition`).

Métodos

Los métodos pueden concebirse como funciones relacionadas directamente con un objeto y a través de las cuales puede llamarse a éste. El objeto `Document` de este ejemplo ofrece un método `Save`, que puede llamarse de la siguiente forma:

```
Document.Save()
```

Los métodos, igual que las funciones, pueden llevar asociados parámetros y valores de retorno. La sintaxis de las llamadas a métodos es similar a la de las funciones clásicas. La llamada

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

especifica también el parámetro `True` para el objeto de documento al solicitar el método Save. Una vez finalizado el método, `Save` guarda un valor de retorno en la variable `Ok`.

Módulo, servicios e interfaces

StarOffice ofrece centenares de servicios que, para mostrar una visión simplificada, se han combinado en módulos. Éstos no tienen ninguna otra importancia funcional para los programadores de StarOffice Basic. Únicamente al especificar un nombre de servicio tiene importancia el nombre del módulo, ya que éste debe aparecer también en el nombre del servicio. El nombre completo de un servicio incluye la expresión `com.sun.star`, que especifica que se trata de un servicio de StarOffice, seguida por los nombre del módulo, como, por ejemplo, `frame,` y del servicio en sí, como `Desktop`. En el ejemplo mencionado, el nombre completo sería:

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

Aparte de los términos de módulo y servicio, UNO introduce el de ***•ginterfaz•h.*** Aunque los programadores de Java pueden estar familiarizados con dicho término, éste no se utiliza en Basic.

Una interfaz combina diversos métodos. Aunque en un sentido estricto, un servicio de UNO no admite métodos, sino interfaces, las cuales proporcionan distintos métodos. En otras palabras, los métodos se asignan (como combinaciones) al servicio en forma de interfaces. Este detalle puede ser de interés especialmente para los programadores de Java- o C++, puesto que, en dichos lenguajes, la interfaz es necesaria para solicitar un método. En StarOffice Basic, es irrelevante. En él, los métodos se llaman directamente a través del objeto pertinente.

No obstante, para comprender la API resulta útil tener presente la asignación de métodos a diversas interfaces, ya que muchas de ellas se utilizan en distintos servicios. Si está familiarizado con una interfaz, puede transmitir sus conocimientos de un servicio a otro.

Algunas de las principales interfaces se utilizan con tanta frecuencia que vuelven a aparecer al final de este capítulo, asociadas a distintos servicios.

Herramientas para trabajar con UNO

La cuestión fundamental sigue siendo qué objetos (objeto servicios, para seguir con la terminología de UNO) admiten qué propiedades, métodos e interfaces y cómo éstos pueden determinarse. Aparte de este manual, hay otras formas de obtener más información acerca de objetos: el método `supportsService`, los métodos de depuración, la Guía del desarrollador y la referencia de la API.

El método supportsService

Algunos de los objetos UNO admiten el método `supportsService` que permite establecer si un objeto admite un servicio específico o no. La llamada

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

por ejemplo, determina si el objeto `TextElement` admite el servicio `com.sun.star.text.Paragraph`.

Propiedades de depuración

Todos los objetos UNO de StarOffice Basic ***saben*** qué propiedades, métodos e interfaces contienen. Dichos objetos contienen propiedades que devuelven una lista de estos métodos, interfaces y propiedades. Las propiedades correspondientes son:

`DBG_properties` : devuelve una cadena que contiene todas las propiedades de un objeto

`DBG_methods` : devuelve una cadena que contiene todos los métodos de un objeto

`DBG_supportetInterfaces` : devuelve una cadena que contiene todas las interfaces admitidas por un objeto.

El siguiente fragmento de programa muestra de qué forma se pueden utilizar `DBG_properties` y `DBG_methods` en aplicaciones reales. Dicho fragmento crea en primer lugar el servicio `com.sun.star.frame.Desktop` y luego muestra las propiedades y métodos admitidos en cuadros de mensaje.

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

MsgBox Obj.DBG_Propierties
MsgBox Obj.DBG_methods
```

Cuando utilice `DBG_properties`, tenga en cuenta que la función devuelve todas las propiedades que un servicio específico puede potencialmente admitir. No obstante, no se garantiza que el objeto pueda utilizarlas. Antes de hacer una llamada a las propiedades se deberá utilizar la función `IsEmpty` para comprobar si están disponibles.

Guía de referencia de la API

La guía de referencia de la API de StarOffice contiene información adicional acerca de los servicios disponibles y sus interfaces, métodos y propiedades. Se puede acceder a dicha guía en www.openoffice.org:

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

Resumen de algunas de las principales interfaces

Algunas de las interfaces de StarOffice aparecen en diversos lugares de la API de StarOffice. Dichas interfaces definen métodos para tareas abstractas que pueden aplicarse a la resolución de problemas muy diversos. Aquí encontrará una descripción resumida de las interfaces más comunes.

El origen de los objetos se explica más adelante en este manual. En este momento se tendrán únicamente en cuenta algunos de los aspectos abstractos de los objetos para los cuales la API de StarOffice ofrece varias interfaces.

Creación de objetos dependientes del contexto

La API de StarOffice ofrece dos opciones para la creación de objetos. Una es la función `createUnoService` mencionada al inicio de este capítulo. `createUnoService` crea un objeto que se puede utilizar de forma universal. Dichos objetos y servicios se denominan también ***servicios independientes del contexto***.

Además de éstos, también hay ***servicios dependientes del contexto*** cuyos objetos sólo pueden utilizarse en conjunción con otro objeto. Por ejemplo, un objeto de dibujo para un documento de hoja de cálculo sólo puede existir conjuntamente con este tipo de documento.

Interfaz com.sun.star.lang.XMultiServiceFactory

Los objetos dependientes del contexto se suelen crear mediante un método de objeto del que dependen. El método `createInstance`, definido en la interfaz `XMultiServiceFactory`, se utiliza específicamente en los objetos de documento.

El objeto de dibujo mencionado arriba se podría crear, por ejemplo, de la siguiente forma a partir de un objeto de hoja de cálculo:

```
Dim FormaRectangular As Object

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

De forma similar se puede crear una plantilla de párrafo en un documento de texto:

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

Acceso a objetos subordinados mediante el nombre

Las interfaces XNameAccess y XNameContainer se utilizan en objetos que contienen objetos subordinados a los que se puede asignar un nombre convencional.

Mientras que XNamedAccess permite acceder a los objetos individuales, XNameContainer se encarga de la inserción, modificación y supresión de elementos.

Interfaz com.sun.star.container.XNameAccess

En ejemplo de uso de XNameAccess lo ofrece el objeto de hoja de una hoja de cálculo; éste combina todas las páginas que contiene aquélla. Para acceder a las páginas individuales se utiliza el método getByName de XNameAccess:

```
Dim Hojas As Object
Dim Hoja As Object

Hojas = Spreadsheet.Sheets
Hoja = Sheets.getByName("Hoja1")
```

El método getElementNames proporciona una lista con los nombres de todos los elementos. Devuelve como resultado un campo de datos que contiene dichos nombres. En el ejemplo siguiente se muestra cómo se pueden determinar y mostrar los nombres de los elementos en un bucle:

```
Dim Hojas As Object
Dim NombresHojas
Dim I As Integer

Hojas = Spreadsheet.Sheets
NombresHojas = Sheets.getElementNames

For I=LBound(NombresHojas) To UBound(NombresHojas)
      MsgBox NombresHojas(I)
Next I
```

El método hasByName de la interfaz XNameAccess revela si dentro del objeto básico hay un objeto subordinado con un nombre específico. En el ejemplo siguiente se muestra, por tanto, un mensaje en el que se informa al usuario si el objeto Spreadsheet contiene una página con el nombre Hoja1.

```
Dim Hojas As Object

Hojas = Spreadsheet.Sheets
If Hojas.HasByName("Hoja1") Then
      MsgBox " Hoja1 está disponible"
Else
      MsgBox " Hoja1 no está disponible"
End If
```

Interfaz com.sun.star.container.XNameContainer

La interfaz `XNameContainer` se encarga de la inserción, supresión y modificación de los elementos subordinados de un objeto básico. Las funciones responsables de ello son `insertByName`, `removeByName` y `replaceByName`.

A continuación se muestra un ejemplo práctico . En el ejemplo se llama a un documento de texto que contiene un objeto `FamiliasDeEstilos` y se utiliza éste para, a su vez, hacer disponibles las plantillas de párrafo (ParagraphStyles) del documento.

```
Dim FamiliasDeEstilos As Objects
Dim EstilosDeParrafo As Objects
Dim EstiloNuevo As Object

FamiliasDeEstilos = Textdoc.StyleFamilies
EstilosDeParrafo = StyleFamilies.getByName("ParagraphStyles")

EstilosDeParrafo.insertByName("EstiloNuevo", EstiloNuevo)
EstilosDeParrafo.replaceByName("EstiloQueSeModifica", EstiloNuevo)
EstilosDeParrafo.removeByName("EstiloAntiguo")
```

La línea `insertByName` inserta el estilo `EstiloNuevo` con el mismo nombre en el objeto `EstilosDeParrafo`. La línea `replaceByName` cambia el objeto `EstiloQueSeModifica` a `EstiloNuevo`. Finalmente, la línea `removeByName` suprime el objeto `EstiloAntiguo` de `EstilosDeParrafo`.

Acceso a objetos subordinados a través del índice

Las interfaces `XIndexAccess` y `XIndexContainer` se utilizan en objetos que contienen objetos subordinados a los que se puede acceder a través de un índice.

`XIndexAccess` proporciona métodos para acceder a objetos individuales. `XIndexContainer` proporciona métodos para insertar y suprimir elementos.

Interfaz com.sun.star.container.XIndexAccess

`XIndexAccess` ofrece los métodos `getByIndex` y `getCount` para llamar a los objetos subordinados. `getByIndex` proporciona un objeto con un índice específico. `getCount` devuelve el número de objetos disponibles.

```
Dim Hojas As Object
Dim Hoja As Object
Dim I As Integer

Hojas = Spreadsheet.Sheets

For I = 0 to Hojas.getCount() - 1
      Hoja = Hojas.getByIndex(I)
      ' Editar hoja
Next I
```

El ejemplo supone un bucle que se ejecuta secuencialmente a través de todos los elementos y guarda una referencia a cada uno de ellos en la variable de objeto `Hoja`. Al trabajar con índices, tenga en cuenta que `getCount` devuelve el número de elementos. Sin embargo, los elementos en `getByIndex` están numerados a partir del `0`. Por tanto, la variable de cuenta del bucle va de `0` a `getCount()-1`.

Interfaz com.sun.star.container.XIndexContainer

La interfaz `XIndexContainer` ofrece las funciones `insertByIndex` y `removeByIndex`. Los parámetros están estructurados de la misma forma que las funciones correspondientes de `XNameContainer`.

Acceso iterativo a objetos subordinados

En ciertos casos, un objeto puede contener una lista de objetos subordinados a la que no puede accederse mediante nombre ni mediante índice. En tales situaciones es adecuado utilizar las interfaces `XEnumeration` y `XenumerationAccess` que proporcionan un mecanismo mediante el cual se pueden recorrer paso a paso todos los elementos subordinados de un objeto sin necesidad de dirigirse a ellos de forma directa.

Interfaces com.sun.star.container.XEnumeration y XenumerationAccess

El objeto básico debe ofrecer la interfaz `XEnumerationAccess` que contiene únicamente un método `createEnumeration`. Éste devuelve un objeto auxiliar que, a su vez, proporciona a la interfaz `XEnumeration` los métodos `hasMoreElements` y `nextElement`. Dichos métodos permiten acceder a los objetos subordinados.

El ejemplo siguiente recorre todos los párrafos de un texto:

```
Dim EnumeracionDeParrafos As Object
Dim Parrafo As Object

EnumeracionDeParrafos = Textdoc.Text.createEnumeration

While EnumeracionDeParrafos.hasMoreElements()
      Parrafo = ParagraphElements.nextElement()
Wend
```

En el ejemplo se crea en primer lugar el objeto auxiliar `EnumeracionDeParrafos` que devuelve gradualmente mediante un bucle los párrafos individuales del texto. El bucle finaliza en el momento en que el método `hasMoreElements` devuelve el valor `False` que indica que se ha llegado al final del texto.

CAPÍTULO 5

Trabajo con documentos de StarOffice

La API de StarOffice está estructurada de forma que casi todas sus partes se puedan utilizar de forma universal para distintas tareas. Esto incluye las interfaces y servicios para crear, abrir, guardar, convertir e imprimir documentos y para administrar plantillas. Puesto que dichas funciones están disponibles en todo tipo de documentos, se explican en primer lugar en el capítulo.

El objeto StarDesktop

Durante el trabajo con documentos, dos son los servicios que se utilizan con mayor frecuencia:

- El servicio `com.sun.star.frame.Desktop`, similar al servicio de núcleo de StarOffice. Ofrece las funciones del objeto marco de StarOffice que abarca todas las ventanas del documento. Este servicio permite asimismo crear, abrir e importar objetos.
- El servicio `com.sun.star.document.OfficeDocument` proporciona la funcionalidad básica para los objetos de documento individuales. Dicho servicio contiene los métodos utilizados para guardar, exportar e imprimir documentos.

El servicio `com.sun.star.frame.Desktop` se abre automáticamente al iniciar StarOffice. Para ello, StarOffice crea un objeto al que se puede acceder mediante el nombre global `StarDesktop`.

La interfaz más importante del objeto `StarDesktop` es `com.sun.star.frame.XComponentLoader` que incluye básicamente el método `loadComponentFromURL`, responsable de crear, importar y abrir documentos.

El nombre del objeto `StarDesktop` procede de StarOffice 5 que incluía todas las variables de documento en una aplicación común denominada StarDesktop. En la versión actual de StarOffice ya no se usa un StarDesktop visible. No obstante, se ha conservado el nombre StarDesktop para el objeto marco de StarOffice debido a que indica claramente que se trata de un objeto básico para la totalidad de la aplicación.

El objeto `StarDesktop` adquiere el papel de sucesor del objeto `Application` de StarOffice 5 que se consideraba anteriormente el objeto raíz. Sin embargo, a diferencia del antiguo objeto `Application`, StarDesktop es principalmente responsable de abrir documentos nuevos. Las funciones de control del aspecto en pantalla de StarOffice Basic que residían en el objeto `Application` (por ejemplo, `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) han dejado de utilizarse.

Mientras que Word usa `Application.ActiveDocument` y Excel `Application.ActiveWorkbook`, para acceder al documento activo, StarOffice usa `StarDesktop` para esta tarea. En StarOffice Basic 6, el acceso al objeto de documento activo se efectúa mediante la propiedad `StarDesktop.CurrentComponent`.

Información básica acerca de los documentos en StarOffice

Al trabajar con documentos de StarOffice, resulta útil tener en cuenta algunos de los aspectos básicos de la administración de documentos en StarOffice. Estos aspectos incluyen la estructura de los nombres de archivo de los documentos de StarOffice, así como el formato en el que se guardan los archivos.

Nombres de archivo en notación de URL

StarOffice es una aplicación diseñada como independiente de la plataforma, por lo que sus nombres de archivo utilizan notación URL (independiente del sistema operativo) según la normativa Internet Standard RFC 1738. Los nombres de archivos estándar de este sistema se inician con el prefijo

`file:///`

seguido por la ruta local. Si el nombre del archivo contiene subdirectorios, éstos se separan mediante una *barra*, en lugar de la contrabarra utilizada en Windows. La ruta siguiente hace referencia al archivo `prueba.sxw` situado en el directorio `doc` de la `unidad C:`.

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

Para convertir nombres de archivo locales en URL, StarOffice ofrece la función `ConvertToUrl`. Para convertir una URL en un nombre de archivo local, ofrece la función `ConvertFromUrl`.

```
MsgBox ConvertToUrl("C:\doc\prueba.sxw")
            ' proporciona file:///C:/doc/prueba.sxw

MsgBox ConvertFromUrl("file:///C:/doc/prueba.sxw")
            '  proporciona (en Windows) c:\doc\prueba.sxw
```

En el ejemplo se convierte un nombre de archivo local en una URL y se muestra ésta en un cuadro de mensaje. A continuación se convierte una URL en un nombre de archivo local y se muestra también dicho nombre.

La normativa Internet Standard RFC 1738 en la que se basa esta funcionalidad permite utilizar los caracteres `0-9`, `a-z` y `A-Z`. El resto de caracteres se insertan en la URL en forma de códigos de escape. Para ello se convierten en su valor hexadecimal en el juego de caracteres ISO 8859-1 (ISO-Latin) y se les antepone un signo de porcentaje. Por consiguiente, un espacio en un nombre de archivo se convierte en `%20` en la URL.

Formato de archivo XML

Desde la versión 6.0, StarOffice utiliza un formato de archivo basado en XML. La adopción de XML ofrece al usuario la opción de abrir y editar los archivos en otros programas.

Compresión de archivos

Puesto que XML se basa en archivos de texto estándar, los archivos resultantes suelen ser de gran tamaño. Por tanto, StarOffice los comprime y los guarda en forma de archivos ZIP.

Una opción del método `storeAsURL` permite al usuario guardar directamente los archivos XML originales. Consulte Opciones del método storeAsURL en la página 78.

Creación, apertura e importación de documentos

Los documentos se abren, importan y crean mediante el método

```
StarDesktop.loadComponentFromURL(URL, Frame, _
                        IndicadoresBusqueda, PropiedadesArchivo)
```

El primer parámetro de `loadComponentFromURL` especifica la URL del archivo asociado.

`loadComponentFromURL` espera, como segundo parámetro, un nombre para el objeto marco (Frame) de la ventana que StarOffice crea internamente para su administración. Se suele especificar el nombre predefinido `_blank`, lo que garantiza la creación de una ventana nueva por parte de StarOffice. Otra posibilidad es especificar `_hidden`, con lo que el documento correspondiente se cargará pero permanecerá invisible.

Estos parámetros permiten al usuario abrir un documento de StarOffice, ya que los dos últimos parámetros pueden tener asignados comodines (valores dummy):

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

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

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

La llamada anterior abre el archivo prueba.sxw y lo muestra en una ventana nueva.

De esta forma se puede abrir en StarOffice Basic cualquier número de documentos y, a continuación, editarlos utilizando los objetos de documento devueltos.

`StarDesktop.loadComponentFromURL` reemplaza los métodos `Documents.Add` y `Documents.Open` de la antigua API de StarOffice Basic.

Sustitución del contenido de la ventana de documento

Los valores named `_blank` y `_hidden` para el parámetro `Frame` garantizan que StarOffice creará una ventana nueva en cada llamada a `loadComponentFromURL`. En algunas situaciones es útil reemplazar el contenido de una ventana. En tal caso, el objeto marco de la ventana debe tener un nombre explícito; tenga en cuenta que éste no puede comenzar con un carácter de subrayado. Además, el parámetro `IndicadoresBusqueda` se debe asignar para que se cree el marco de trabajo correspondiente, si no existe ya. La constante correspondiente para `IndicadoresBusqueda` es:

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

En el ejemplo siguiente se muestra cómo sustituir el contenido de una ventana abierta con la ayuda del parámetro Frame y de `IndicadoresBusqueda`:

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

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

Url = "file:///C:/prueba.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MiMarco", _
              IndicadoresBusqueda, Dummy)

MsgBox "Pulse Aceptar para mostrar el segundo documento."

Url = "file:///C:/prueba2.sxw"
Doc = StarDesktop.loadComponentFromURL(Url, "MiMarco", _
              IndicadoresBusqueda, Dummy)
```

En el ejemplo, en primer lugar, se abre el archivo `prueba.sxw` en una ventana nueva con el nombre de marco `MiMarco`. Una vez confirmado el cuadro de mensaje, se sustituye el contenido de la ventana por el archivo `prueba2.sxw`.

Opciones del método loadComponentFromURL

El cuarto parámetro de la función `loadComponentFromURL` es un campo de datos de tipo `PropertyValue` que ofrece a StarOffice diversas opciones para abrir y crear documentos. El campo de datos debe proporcionar una estructura `PropertyValue` para cada opción en la que el nombre de la opción y el valor asociado se guarden en formato de cadena de caracteres.

`loadComponentFromURL` admite las opciones siguientes:

- **`AsTemplate (booleano)`**: si es True, carga un documento nuevo sin título de la URL especificada. Si es False, se carga un archivo de plantilla para editarlo.
- **`CharacterSet (String)`**: define en qué juego de caracteres se basa un documento.
- **`FilterName (String)`**: especifica un filtro especial para la función `loadComponentFromURL`. Los nombres de filtro disponibles están definidos en el archivo `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **`FilterOptions (String)`**: define opciones de filtro adicionales.
- **`JumpMark (String)`**: una vez abierto un documento, salta a la posición definida en JumpMark.
- **`Password (String)`**: transfiere una contraseña para un archivo protegido.
- **`ReadOnly (booleano)`**: carga un documento de sólo lectura.

En el ejemplo siguiente se muestra de qué forma un archivo de texto separado por comas de StarOffice Calc puede abrirse mediante la opción `FilterName`.

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

Url = "file:///C:/csv.doc"

PropiedadesArchivo(0).Name = "FilterName"
PropiedadesArchivo(0).Value = "scalc: Text - txt - csv (StarOffice Calc)"

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

El campo de datos `PropiedadesArchivo` cubre específicamente un valor porque registra una única opción. La propiedad `Filtername` define si StarOffice debe usar un filtro de texto de StarOffice Calc para abrir archivos.

Creación de documentos nuevos

StarOffice crea automáticamente un documento nuevo si el documento especificado en la URL es una plantilla.

Otra posibilidad, en el caso de necesitar únicamente un documento vacío sin ninguna adaptación, es especificar una URL `private:factory`:

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

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

La llamada crea un documento vacío de StarOffice Writer.

Objetos documento

La función `loadComponentFromURL` presentada en la sección anterior devuelve un objeto documento. Éste admite el servicio `com.sun.star.document.OfficeDocument` que, a su vez, proporciona dos interfaces principales:

- la interfaz `com.sun.star.frame.XStorable`, responsable de guardar documentos
- la interfaz `com.sun.star.view.XPrintable` con los métodos para imprimir documentos.

Al cambiar a StarOffice 6 verá que el ámbito funcional de los objetos documento no ha cambiado en general. Por ejemplo, los objetos documento siguen proporcionando métodos para guardar e imprimir documentos. Sin embargo, los nombres y parámetros de los métodos han cambiado.

Guardado y exportación de documentos

Los documentos de StarOffice se guardan directamente a través del objeto documento. Para ello se dispone del método `store` de la interfaz `com.sun.star.frame.XStorable`:

```
Doc.store()
```

Esta llamada funciona si ya se ha asignado al documento un espacio en la memoria. Este no es el caso de los documentos nuevos. Para estos se utiliza el método `storeAsURL` que está también definido en `com.sun.star.frame.XStorable` y se puede utilizar para definir la ubicación del documento:

```
Dim URL As String
Dim Dummy()

Url = "file:///C:/prueba3.sxw"

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

Además de los métodos anteriores, `com.sun.star.frame.XStorable` proporciona también otros métodos de ayuda para guardar documentos. Éstas son:

- **`hasLocation()`** : especifica si se ha asignado una URL al documento.
- **`isReadonly()`** : especifica si un documento tiene protección de sólo lectura.
- **`isModified()`** : especifica si un documento se ha modificado desde la última vez que se guardó.

El código para guardar un documento se puede ampliar con estas opciones, de forma que sólo se guarde si el objeto se ha modificado y sólo se solicite el nombre de archivo si es necesario:

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

El ejemplo comprueba, primero, si el documento pertinente se ha modificado desde la última vez que se guardó. El proceso de guardar prosigue sólo en ese caso. Si ya se ha asignado una URL al documento y no se trata de un documento de sólo lectura, se guarda con la URL existente. Si no tiene asignada una URL o se ha abierto en estado de sólo lectura, se guarda con una URL nueva.

Opciones del método storeAsURL

De forma similar al método `loadComponentFromURL`, es posible especificar algunas opciones con formato del campo de datos `PropertyValue` mediante el método `storeAsURL`. Dichas opciones determinan el procedimiento utilizado por StarOffice para guardar un documento. `storeAsURL` ofrece las opciones siguientes:

- **`CharacterSet (String)`**: define en qué juego de caracteres se basa un documento.
- **`FilterName (String)`**: especifica un filtro especial para la función `loadComponentFromURL`. Los nombres de filtro disponibles están definidos en el archivo `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **`FilterOptions (String)`**: define opciones de filtro adicionales.

- **`Overwrite (booleano)`**: permite sobrescribir un archivo sin consultarlo.
- **`Password (String)`**: transfiere una contraseña para un archivo protegido.
- **`Unpacked (booleano)`**: guarda el documento (sin comprimir) en subdirectorios.

En el ejemplo siguiente se muestra cómo se puede utilizar la opción `Overwrite` en conjunción con `storeAsURL`:

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

' ... Inicializar Doc

Url = "file:///c:/prueba3.sxw"

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

Doc.storeAsURL(Url, PropiedadesArchivo())
```

A continuación, en el ejemplo se guarda `Doc` con el nombre de archivo especificado si ya existe un archivo con ese nombre.

Impresión de documentos

De forma similar al proceso de guardar, los documentos se imprimen directamente a través del objeto documento. Para ello se incluye el método `print` de la interfaz `com.sun.star.view.Xprintable`. En su forma más sencilla, la llamada a `print` es:

```
Dim Dummy()

Doc.print(Dummy())
```

Como en el método `loadComponentFromURL`, el parámetro `Dummy` es un campo de datos `PropertyValue` a través del cual StarOffice puede especificar diversas opciones de impresión.

Opciones del método print

El método `print` espera recibir un parámetro del campo de datos `PropertyValue` que refleje los parámetros de configuración del diálogo de impresión de StarOffice:

- **`CopyCount (entero)`** : especifica el número de copias que se deben imprimir.
- **`FileName (String)`** : imprime el documento en el archivo especificado.
- **`Collate (booleano)`**: indica a la impresora que intercale las páginas de las copias.
- **`Sort (booleano)`** : ordena las páginas cuando se imprimen varias copias (`CopyCount` > 1).
- **`Pages (String)`**: contiene la lista de las páginas que se deben imprimir (la sintaxis es la misma que se usa en el diálogo de imprimir).

En el ejemplo siguiente se muestra cómo imprimir varias páginas de un documento mediante la opción `Pages`:

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

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

Doc.print(PropiedadesImpresion())
```

Selección y configuración de la impresora

La interfaz `com.sun.star.view.XPrintable` ofrece la propiedad `Printer` para seleccionar la impresora. Esta propiedad recibe un campo de datos `PropertyValue` con la siguiente configuración:

- **`Name (String)`** : especifica el nombre de la impresora.
- **`PaperOrientation (Enum)`** : especifica la orientación del papel (valor `com.sun.star.view.PaperOrientation.PORTRAIT` para formato vertical, `com.sun.star.view.PaperOrientation.LANDSCAPE` para formato horizontal).
- **`PaperFormat (Enum)`** : especifica el formato del papel (por ejemplo, com.sun.star.view.PaperFormat.A4 para DIN A4 o `com.sun.star.view.PaperFormat.Letter` para cartas de formato EE.UU.).
- **`PaperSize (tamaño)`** : especifica el tamaño del papel en centésimas de milímetro.

En el ejemplo siguiente se muestra cómo cambiar una impresora y establecer el tamaño del papel mediante la propiedad `Printer`.

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

TamanoPapel.Width = 20000    ' corresponde a 20 cm
TamanoPapel.Height = 20000   ' corresponde a 20 cm

PropiedadesImpresora(0).Name="Name"
PropiedadesImpresora(0).Value="Mi HP Laserjet"

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

Doc.Printer = PropiedadesImpresora()
```

El ejemplo define un objeto denominado `TamanoPapel` con el tipo `com.sun.star.awt.Size`. Esta acción es necesaria para especificar el tamaño del papel. A continuación crea un campo de datos para dos entradas de tipo `PropertyValue` denominado `PropiedadesImpresora`. Este campo de datos se inicializa con los valores que se deben establecer y asignar a la propiedad `Printer`. Desde el punto de vista de UNO, Printer no es una propiedad real, sino una propiedad ***imitada***.

Plantillas

Las plantillas son listas con nombre que contienen atributos de formato. Abarcan todas las aplicaciones de StarOffice y simplifican las tareas de formato de forma significativa. Si el usuario cambia uno de los atributos de una plantilla, StarOffice ajusta automáticamente todas las secciones de documento que dependan de dicho atributo. Por ejemplo, el usuario puede cambiar la fuente de todos los encabezamientos de nivel uno mediante una única modificación en el documento. Según los tipos de documento, StarOffice reconoce una amplia gama de tipos de plantilla.

StarOffice Writer admite

- plantillas de caracteres,
- plantillas de párrafo,
- plantillas de marco,
- plantillas de página
- plantillas de numeración

StarOffice Calc admite

- plantillas de celda
- plantillas de página

StarOffice Impress admite

- plantillas de elemento de carácter
- plantillas de presentación

En la terminología de StarOffice, los distintos tipos de plantilla se denominan `StyleFamilies` (familias de estilos), debido al servicio `com.sun.star.style.StyleFamily` en el que se basan. Se accede a las `StyleFamilies` a través del objeto documento:

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosDeCeldas As Object

Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDeCelda = StyleFamilies.getByName("CellStyles")
```

El ejemplo utiliza la propiedad `StyleFamilies` de un documento de hoja de cálculo para establecer una lista que contenga todas las plantillas de celda disponibles.

Se puede acceder a las plantillas individuales directamente mediante un índice:

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosDeCeldas As Object
Dim EstiloDeCelda As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDeCelda = StyleFamilies.getByName("CellStyles")

For I = 0 To EstilosDeCelda.Count - 1
      EstiloDeCelda = EstilosDeCelda(I)
      MsgBox CellStyle.Name
Next I
```

El bucle agregado muestra los nombres de todos los estilos de celda sucesivamente en un cuadro de mensaje.

Detalles acerca de diversas opciones de formato

Cada un de los tipos de plantilla ofrece una amplia gama de propiedades de formato individuales. A continuación se resumen las propiedades de formato más importantes y el lugar donde se explican:

- Propiedades de carácter, capítulo 6, Documentos de texto:, servicio `com.sun.star.style.CharacterProperties`
- Propiedades de párrafo, capítulo 6, Documentos de texto:, servicio `com.sun.star.style.Paragraph`
- Propiedades de celda, capítulo 7, Documentos de hoja de cálculo, servicio `com.sun.star.table.CellProperties`
- Propiedades de página, capítulo 7, Documentos de hoja de cálculo, servicio `com.sun.star.table.PageStyle`
- Propiedades de elemento de carácter, capítulo 7, Documentos de hoja de cálculo, servicios diversos

Las propiedades de formato no están en absoluto restringidas a las aplicaciones en las que se explican, sino que pueden utilizarse de forma universal. Por ejemplo, la mayoría de las propiedades de página descritas en el capítulo 7 pueden utilizarse tanto en StarOffice Calc como en StarOffice Writer.

Para obtener más información acerca del trabajo con plantillas consulte la sección ***Valores predeterminados de las propiedades de carácter y de párrafo*** en el capítulo 6, Documentos de texto:.

CAPÍTULO 6

Documentos de texto:

Aparte de cadenas de caracteres puras, los documentos de texto contienen también información de formato que puede aparecer en cualquier lugar del texto. En el caso de las tablas, la estructura es mucho más complicada, ya que éstas no sólo contienen cadenas unidimensionales, sino también campos bidimensionales. La mayoría de los programas de procesamiento de textos ofrecen también la opción de insertar objetos de dibujo, marcos de texto y otros objetos dentro de un texto que pueden estar tanto fuera del flujo de texto como situados en cualquier lugar de la página.

En este capítulo se presentan las interfaces y los servicios fundamentales de los documentos de texto. La primera sección trata sobre la estructura de los documentos de texto y el uso de los programas en StarOffice Basic para efectuar tareas iterativas en un documento de StarOffice. Se centra en párrafos, fragmentos de párrafo y su formato.

La segunda sección trata sobre el trabajo eficiente con documentos de texto para el que StarOffice ofrece varios objetos de ayuda, como `TextCursor`, que se añaden a los especificados en la primera sección.

La tercera sección trata de otros temas distintos al trabajo con textos; se centra en las tablas, marcos de texto, marcadores, listas de contenidos, etc.

La información sobre cómo crear, abrir, guardar e imprimir documentos se describe en el capítulo 5, ya que no sólo es válida para los documentos de texto, sino para otros tipos de documentos.

La estructura de los documentos de texto

Un documento puede contener esencialmente cuatro tipos de información:

- el texto en sí
- plantillas para el formato de caracteres, párrafos y páginas
- elementos no estrictamente textuales, como tablas, imágenes y objetos de dibujo
- parámetros de configuración globales del documento de texto

Esta sección se centra especialmente en el texto y en las opciones de formato asociadas con aquél.

El diseño de la API de StarOffice 6.x para StarOffice Writer es distinto de la versión anterior. La versión antigua de la API estaba centrada en el trabajo con el objeto `Selection`, claramente orientado hacia la idea de la interfaz de usuario para usuarios finales, basado en resaltar mediante el ratón.

La API de StarOffice 6.x ha reemplazado estas conexiones entre las interfaces del usuario y del programador. Por tanto, éste puede acceder en paralelo a todas las partes de una aplicación y trabajar con objetos de distintas subsecciones de un documento al mismo tiempo. El antiguo objeto `Selection` ya no está disponible.

Párrafos y fragmentos de párrafo

El núcleo de un documento de texto lo constituye una serie de párrafos sin nombre ni índice, por lo que no hay posibilidad de acceder ***directamente*** a los párrafos individuales; no obstante, se pueden recorrer secuencialmente con la ayuda del objeto `Enumeration` descrito en el capítulo 4. Esto permite editar los párrafos.

Cuando se trabaja con el objeto `Enumeration` se debe tener en cuenta una característica especial: no sólo devuelve párrafos, sino también tablas (estrictamente, en StarOffice Writer, una tabla es un tipo especial de párrafo). Para poder acceder a un objeto devuelto se debe comprobar si éste admite el servicio `com.sun.star.text.Paragraph` para párrafos o el servicio `com.sun.star.text.TextTable` para tablas.

En el ejemplo siguiente se recorre el contenido de un documento de texto mediante un bucle y se utiliza un mensaje en cada caso para informar al usuario si el objeto en cuestión es un párrafo o una tabla.

```
Dim Doc As Object
Dim Enum As Object
Dim ElementoTexto As Object

' Crear objeto documento
Doc = StarDesktop.CurrentComponent

' Crear objeto enumeración
Enum = Doc.Text.createEnumeration

' recorrer en bucle todos los elementos del texto
While Enum.hasMoreElements
	ElementoTexto = Enum.nextElement

	If ElementoTexto.supportsService("com.sun.star.text.TextTable") Then
		MsgBox "El bloque actual contiene una tabla."
	End If

	If ElementoTexto.supportsService("com.sun.star.text.Paragraph") Then
		MsgBox "El bloque actual contiene un párrafo."
	End If
Wend
```

En el ejemplo se crea un objeto documento `Doc` que hace referencia al documento de StarOffice actual. Con la ayuda de `Doc` se crea un objeto `Enumeration` que recorre las partes individuales del texto (párrafos y tablas) y asigna el elemento actual al objeto `ElementoTexto`. El ejemplo utiliza el método `supportsService` para comprobar si `ElementoTexto` es un párrafo o una tabla para mostrar el mensaje correspondiente.

Párrafos

El servicio `com.sun.star.text.Paragraph` ofrece acceso al contenido de un párrafo. El texto del párrafo se puede recuperar y modificar mediante la propiedad `String`:

```
Dim Doc As Object
Dim Enum As Object
Dim ElementoTexto As Object

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

While Enum.hasMoreElements
      ElementoTexto = Enum.nextElement

      If ElementoTexto.supportsService("com.sun.star.text.Paragraph") Then
            TextElement.String = Replace(TextElement.String, "dos", "2")
            TextElement.String = Replace(TextElement.String, "tres", "3")
            TextElement.String = Replace(TextElement.String, "cuatro", "4")
      End If
Wend
```

En el ejemplo se abre el documento de texto actual y se recorre con la ayuda del objeto `Enumeration`. Se utiliza la propiedad `TextElement.String` en todos los párrafos para acceder a los párrafos pertinentes y se sustituyen las cadenas `dos`, `tres` y `cuatro` por los caracteres `2`, `3` y `4`. La función `Replace` que se utiliza para la sustitución no está dentro del ámbito lingüístico estándar de StarOffice Basic. Se trata de un uso de la función de ejemplo descrita en el capítulo 3, sección ***Buscar y reemplazar***.

El contenido del procedimiento descrito aquí para acceder a los párrafos de un texto se puede comparar con la lista `Paragraphs`, utilizada en VBA, que se encuentra en los objetos `Range` y `Document` de VBA. Mientras que, en VBA, se accede a los párrafos por su número (por ejemplo, mediante la llamada `Paragraph(1)`), en StarOffice Basic se debe utilizar el objeto `Enumeration` descrito anteriormente.

Las listas `Characters`, `Sentences` y `Words` de VBA no tienen una contrapartida directa en StarOffice Basic. Sí existe la opción de cambiar a un objeto `TextCursor` (cursor de texto) que permite desplazarse en el nivel de caracteres, frases y palabras (consulte *El objeto TextCursor*).

Partes de párrafos

El ejemplo anterior puede modificar el texto según se solicite, pero a veces también puede destruir el formato.

El motivo es que un párrafo consta a su vez de subobjetos individuales cada uno de ellos con su propia información de formato. Si un párrafo, por ejemplo, contiene una palabra en negrita, el párrafo se representará en StarOffice en forma de tres fragmentos de párrafo: el fragmento anterior a la negrita, la palabra en negrita y, finalmente, el fragmento posterior a la negrita, que vuelve a tener el aspecto normal.

Si se modifica el texto del párrafo mediante la propiedad `String` del párrafo, StarOffice en primer lugar borra los fragmentos de párrafo antiguos e inserta un fragmento nuevo. El formato de las secciones anteriores se pierde.

Para impedirlo, el usuario puede acceder a los fragmentos de párrafo asociados en lugar de al párrafo entero. Los párrafos proporcionan para ello su propio objeto `Enumeration`. En el ejemplo

siguiente se muestra un bucle doble que recorre todos los párrafos de un texto y los fragmentos de párrafo contenidos en ellos y aplica los procesos de sustitución del ejemplo anterior:

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim ElementoTexto As Object
Dim FragmentoTexto As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' bucle para todos los párrafos
While Enum1.hasMoreElements
	ElementoTexto = Enum1.nextElement
	If ElementoTexto.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = ElementoTexto.createEnumeration

		' bucle para todos los subpárrafos
		While Enum2.hasMoreElements
			FragmentoTexto = Enum2.nextElement
			MsgBox "'" & FragmentoTexto.String & "'"
			FragmentoTexto.String = Replace(FragmentoTexto.String, "dos", "2")
			FragmentoTexto.String = Replace(FragmentoTexto.String, "tres", "3")
			FragmentoTexto.String = Replace(FragmentoTexto.String, "cuatro",
"4")
		Wend

	End If
Wend
```

El ejemplo se ejecuta en todo el documento de texto mediante un doble bucle. El bucle exterior hace referencia a los párrafos del texto. El bucle interior procesa los fragmentos de estos párrafos. El código de ejemplo modifica el contenido de los fragmentos de párrafo mediante la propiedad `String` del fragmento, como sucedía en el ejemplo anterior con los párrafos. Pero, como los fragmentos de párrafo se editan directamente, su información de formato se conserva al sustituir la cadena.

Formato

Hay diversas formas de dar formato a un texto. La más sencilla es asignar las propiedades de formato directamente a la secuencia de texto; este procedimiento se denomina ***formato directo*** y se utiliza especialmente con documentos breves, porque es el propio usuario quien asigna los parámetros de formato mediante el ratón: se puede, por ejemplo, destacar una palabra determinada de un texto poniéndola en negrita o centrar una línea.

Otra forma de asignar formato al texto es mediante el uso plantillas. Este procedimiento se denomina ***formato indirecto*** y permite al usuario asignar una plantilla predeterminada al fragmento de texto pertinente; Si el diseño de éste se modifica posteriormente, el usuario no tiene más que modificar la plantilla y StarOffice cambiará el aspecto de todos los fragmentos de texto que utilizan ésta.

En VBA, las propiedades de formato de un objeto suelen abarcar diversos subobjetos (por ejemplo,

Range.Font, Range.Borders, Range.Shading, Range.ParagraphFormat). Se accede a las propiedades mediante expresiones en cascada (por ejemplo, Range.Font.AllCaps). En StarOffice Basic, en cambio, las propiedades de formato son accesibles directamente a través de los objetos adecuados (TextCursor, Paragraph, etc). En las dos secciones siguientes podrá ver un resumen de las propiedades de carácter y de párrafo disponibles en StarOffice.

En la antigua API de StarOffice, el formato de un texto se efectuaba esencialmente a través del objeto Selection y sus objetos subordinados (por ejemplo, Selection.Font, Selection.Paragraph y Selection.Border). En la API nueva, las propiedades de formato se encuentran en cada objeto (Paragraph, TextCursor, etc) y pueden aplicarse directamente. A continuación se enumeran las propiedades de carácter y párrafo disponibles.

Propiedades de carácter

Las propiedades de formato que hacen referencia a caracteres individuales se denominan propiedades de carácter. Entre éstas se encuentran las negritas y las fuentes. Los objetos que permitan establecer propiedades de carácter deben admitir el servicio com.sun.star.style.CharacterProperties. StarOffice reconoce una amplia gama de servicios que admiten este servicio. Entre ellos se encuentran los servicios para párrafos com.sun.star.text.Paragraph y los servicios para fragmentos de párrafo com.sun.star.text.TextPortion.

El servicio com.sun.star.style.CharacterProperties no ofrece ninguna interfaz, pero en cambio proporciona un conjunto de propiedades a través de las cuáles se pueden definir y llamar las propiedades de carácter. En la referencia de la API de StarOffice se puede consultar una lista completa de las propiedades de carácter. Las más importantes se describen en la lista siguiente:

- **CharFontName (String)**: nombre de la fuente seleccionada.
- **CharColor (Long)**: color del texto.
- **CharHeight (Float)**: altura del carácter en puntos (pt).
- **CharUnderline (grupo de Constant)**: tipo de carácter de subrayado (constantes de acuerdo con com.sun.star.awt.FontUnderline).
- **CharWeight (grupo de Constant)**: peso de la fuente (constantes de acuerdo con com.sun.star.awt.FontWeight).
- **CharBackColor (Long)**: color del fondo.
- **CharKeepTogether (booleano)**: supresión de salto de línea automático.
- **CharStyleName (String)**: nombre de la plantilla de caracteres.

Propiedades de párrafo

Si la información de formato no se refiere a caracteres individuales, sino a todo el párrafo, se considera una propiedad de párrafo. Esto incluye la distancia del párrafo a los bordes de la página, así como el interlineado. Las propiedades de párrafo están disponibles mediante el servicio `com.sun.star.style.ParagraphProperties`.

Las propiedades de párrafo están disponibles en diversos objetos. Todos los objetos que admiten el servicio `com.sun.star.text.Paragraph` admiten también las propiedades de párrafo de `com.sun.star.style.ParagraphProperties`.

En la referencia de la API de StarOffice se puede consultar una lista completa de las propiedades de párrafo. Las más comunes son:

- **`ParaAdjust (enum)`:** orientación de texto vertical (constantes de acuerdo con `com.sun.star.style.ParagraphAdjust`).
- **`ParaLineSpacing (struct)`:** interlineado (estructura de acuerdo con `com.sun.star.style.LineSpacing`).
- **`ParaBackColor (Long)`:** color del fondo.
- **`ParaLeftMargin (Long)`:** margen izquierdo en centésimas de milímetro.
- **`ParaRightMargin (Long)`:** margen derecho en centésimas de milímetro.
- **`ParaTopMargin (Long)`:** margen superior en centésimas de milímetro.
- **`ParaBottomMargin (Long)`:** margen inferior en centésimas de milímetro.
- **`ParaTabStops (matriz de struct)`:** tipo y posición de tabuladores (matriz de estructura de los tipos `com.sun.star.style.TabStop`).

`ParaStyleName (String)`: nombre de la plantilla de párrafo.

Ejemplo: exportación simple a HTML

En el ejemplo siguiente se muestra la forma de trabajar con información de formato. En él se recorre un documento de texto y se crea un archivo HTML sencillo. Cada párrafo se registra para ello en su propio elemento HTML <P>. Los fragmentos de párrafo que se muestran en negrita se marcan para exportar con el elemento HTML <B>.

```
Dim NumArchivo As Integer, NombreArchivo As String, LineaActual As String

Dim Doc As Object
Dim Enum1 As Object, Enum2 As Object
Dim ElementoTexto As Object, FragmentoTexto As Object

NombreArchivo = "c:\texto.html"
NumArchivo = Freefile
Open NombreArchivo For Output As #NumArchivo
Print #NumArchivo, "<HTML><BODY>"

Doc = StarDesktop.CurrentComponent
```

```
Enum1 = Doc.Text.createEnumeration

' bucle para todos los párrafos
While Enum1.hasMoreElements
       ElementoTexto = Enum1.nextElement
       If ElementoTexto.supportsService("com.sun.star.text.Paragraph") Then
              Enum2 = ElementoTexto.createEnumeration
              LineaActual = "<P>"

              ' bucle para todos los fragmentos de párrafo
              While Enum2.hasMoreElements
                     FragmentoTexto = Enum2.nextElement
                     If FragmentoTexto.CharWeight = com.sun.star.awt.FontWeight.BOLD THEN
                            LineaActual = LineaActual & "<B>" & FragmentoTexto.String &
"</B>"
                     Else
                            LineaActual = LineaActual & FragmentoTexto.String
                     End If
              Wend

              ' enviar la línea
              LineaActual = LineaActual & "</P>"
              Print #NumArchivo, LineaActual

       End If
Wend

' escribir pie de HTML
Print #NumArchivo, "</BODY></HTML>"
Close #NumArchivo
```

La estructura básica del ejemplo está orientada como los ejemplos de recorrido de los fragmentos de párrafo de un texto comentados con anterioridad. Se han agregado las funciones para escribir el archivo HTML, así como un código que comprueba el peso de la fuente de los fragmentos de párrafo correspondientes y escribe fragmentos de párrafo con la etiqueta HTML pertinente.

Valores predeterminados de las propiedades de carácter y de párrafo

El ***formato*** directo tiene siempre prioridad sobre el ***indirecto***. En otras palabras, el formato mediante plantillas tiene asignada una prioridad inferior que el formato directo del texto.

No es fácil establecer si el formato de una sección del texto es directo o indirecto. Las barras de símbolos que proporciona StarOffice muestran las propiedades de texto más comunes, como la fuente, el peso y el tamaño. No obstante, no está del todo claro si la configuración correspondiente está basada en una plantilla o se ha asignado directamente.

StarOffice Basic ofrece el método `getPropertyState`, mediante el cual los programadores pueden comprobar cómo se asignó el formato a una propiedad determinada. Este método toma como parámetro el nombre de la propiedad y devuelve una constante que proporciona información acerca del origen del formato. Las respuestas posibles, definidas en la enumeración `com.sun.star.beans.PropertyState`, son las siguientes:

- **com.sun.star.beans.PropertyState.DIRECT_VALUE:** la propiedad está definida directamente en el texto (formato directo),
- **com.sun.star.beans.PropertyState.DEFAULT_VALUE:** la propiedad se ha definido mediante una plantilla (formato indirecto)
- **com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE:** la propiedad es ambigua. Este estado surge, por ejemplo, al consultar la propiedad de negrita de un párrafo que contiene palabras en negrita y palabras con la fuente normal.

En el ejemplo siguiente se muestra cómo editar las propiedades de formato en StarOffice. En él se buscan fragmentos de párrafo de un texto que se hayan identificado como en negrita con formato directo. Si se encuentra un fragmento de párrafo relevante, se suprime el formato directo mediante el método `setPropertyToDefault` y se asigna una plantilla de caracteres `MiNegrita` al fragmento de párrafo correspondiente.

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim ElementoTexto As Object
Dim FragmentoTexto As Object

Doc = StarDesktop.CurrentComponent
Enum1 = Doc.Text.createEnumeration

' bucle para todos los párrafos
While Enum1.hasMoreElements
      ElementoTexto = Enum1.nextElement
      If ElementoTexto.supportsService("com.sun.star.text.Paragraph") Then
            Enum2 = ElementoTexto.createEnumeration

            ' bucle para todos los fragmentos de párrafo
            While Enum2.hasMoreElements
                  FragmentoTexto = Enum2.nextElement
                  If FragmentoTexto.CharWeight = _
                     com.sun.star.awt.FontWeight.BOLD AND _
                     FragmentoTexto.getPropertyState("CharWeight") = _
                     com.sun.star.beans.PropertyState.DIRECT_VALUE Then

                         FragmentoTexto.setPropertyToDefault("CharWeight")
                         FragmentoTexto.CharStyleName = "MiNegrita"

                  End If
            Wend

      End If
Wend
```

Edición de documentos de texto

En la sección anterior ya se ha comentado una amplia gama de opciones para editar documentos de texto centradas en los servicios `com.sun.star.text.TextPortion` y `com.sun.star.text.Paragraph`, que ofrecen acceso a fragmentos de párrafo y a párrafos enteros. Dichos servicios son útiles para aplicaciones en las que se debe editar el contenido de un texto en un pase de un bucle. No obstante, esto no basta en muchos casos. StarOffice ofrece el servicio `com.sun.star.text.TextCursor` para la realización de tareas más complejas, que incluyen desplazarse hacia atrás en un documento o desplazarse por palabras o frases en lugar de `TextPortions`.

El objeto TextCursor

Un `TextCursor` de la API de StarOffice es comparable al cursor visible de un documento de StarOffice. Marca un punto específico dentro de un documento de texto y puede desplazarse en todas direcciones mediante órdenes. Sin embargo, los objetos `TextCursor` disponibles en StarOffice Basic no deben confundirse con el cursor visible. Se trata de dos cosas muy distintas.

Advertencia: la terminología es distinta de la utilizada en VBA. En términos del ámbito de la función, el objeto `Range` de VBA se puede comparar con el objeto `TextCursor` de StarOffice y no (como podría sugerir su nombre) con el objeto `Range` de StarOffice.

El objeto `TextCursor` de StarOffice, por ejemplo, ofrece métodos para desplazarse en el texto y modificarlo que se incluyen en el objeto `Range` de VBA (por ejemplo, `MoveStart`, `MoveEnd`, `InsertBefore`, `InsertAfter`). Las contrapartidas del objeto `TextCursor` de StarOffice se describen en las próximas secciones.

Desplazamiento dentro de un texto

El objeto `TextCursor` de StarOffice Basic actúa independientemente del cursor visible en un documento de texto. Un cambio de posición controlado por programa de un objeto `TextCursor` no influye en absoluto en el cursor visible. Incluso pueden abrirse varios objetos `TextCursor` en el mismo documento, independientes entre sí, y utilizarse en diversas posiciones.

Para crear un objeto `TextCursor` se utiliza la llamada `createTextCursor`:

```
Dim Doc As Object
Dim Cursor As Object

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

El objeto `Cursor` así creado admite el servicio `com.sun.star.text.TextCursor` que, a su vez, ofrece una amplia variedad de métodos para desplazarse en documentos de texto. En el ejemplo siguiente se mueve en primer lugar el objeto `TextCursor` diez caracteres a la izquierda y a continuación tres caracteres a la derecha:

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

Un `TextCursor` puede utilizarse para destacar un área. Esta acción es comparable a destacar un punto en el texto mediante el ratón. El parámetro `False` de la llamada de función anterior especifica si se debe destacar el área sobre la cual pasa el cursor al moverse. Por ejemplo, el objeto `TextCursor` del ejemplo siguiente

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

primero se mueve diez caracteres hacia la derecha sin destacarlos y luego vuelve hacia atrás tres caracteres y los destaca. El área destacada por el `TextCursor` empieza por tanto a partir del séptimo carácter del texto y finaliza después del décimo.

Los principales métodos que ofrece el servicio `com.sun.star.text.TextCursor` para el desplazamiento son:

- **`goLeft (numero, expandir)`** : salta número caracteres a la izquierda.
- **`goRight (numero, expandir)`** : salta número caracteres a la derecha.
- **`gotoStart (expandir)`** : salta al principio del documento de texto.
- **`gotoEnd (expandir)`** : salta al final del documento de texto.
- **`gotoRange (TextRange, expandir)`**: salta al objeto `TextRange` especificado.
- **`gotoStartOfWord (expandir)`**: salta al principio de la palabra actual.
- **`gotoEndOfWord (expandir)`**: salta al final de la palabra actual.
- **`gotoNextWord (expandir)`**: salta al principio de la palabra siguiente.
- **`gotoPreviousWord (expandir)`**: salta al principio de la palabra anterior.
- **`isStartOfWord ()`**: devuelve `True` si el `TextCursor` está al principio de una palabra.
- **`isEndOfWord ()`**: devuelve `True` si el `TextCursor` está al final de una palabra.
- **`gotoStartOfSentence (expandir)`**: salta al principio de la frase actual.
- **`gotoEndOfSentence (expandir)`**: salta al final de la frase actual.
- **`gotoNextSentence (expandir)`**: salta al principio de la frase siguiente.
- **`gotoPreviousSentence (expandir)`**: salta al principio de la frase anterior.
- **`isStartOfSentence ()`**: devuelve `True` si el `TextCursor` está al principio de una frase.
- **`isEndOfSentence ()`**: devuelve `True` si el `TextCursor` está al final de una frase.
- **`gotoStartOfParagraph (expandir)`**: salta al principio del párrafo actual.
- **`gotoEndOfParagraph (expandir)`**: salta al final del párrafo actual.
- **`gotoNextParagraph (expandir)`**: salta al principio del párrafo siguiente.
- **`gotoPreviousParagraph (expandir)`**: salta al principio del párrafo anterior.
- **`isStartOfParagraph ()`**: devuelve `True` si el `TextCursor` está al principio de un párrafo.
- **`isEndOfParagraph ()`**: devuelve `True` si el `TextCursor` está al final de un párrafo.

El texto se divide en frases mediante símbolos de frase. Por ejemplo, los puntos se interpretan como símbolos que indican el final de una frase.

El parámetro `expandir` es un valor booleano que especifica si el área que se recorre en el desplazamiento se debe desatacar o no. Todos los métodos de desplazamiento devuelven un parámetro que indica si el desplazamiento se ha efectuado satisfactoriamente o si la acción ha finalizado por falta de texto.

A continuación se enumeran diversos métodos para editar las áreas destacadas mediante un `TextCursor` y que admiten también el servicio `com.sun.star.text.TextCursor`:

- **`collapseToStart ()`**: restablece el destacado y la posición de `TextCursor` al principio del área destacada anteriormente.
- **`collapseToEnd ()`**: restablece el destacado y la posición de `TextCursor` al final del área destacada anteriormente.
- **`isCollapsed ()`**: devuelve `True` si el `TextCursor` no cubre actualmente ninguna zona destacada.

Formato de texto con TextCursor

El servicio `com.sun.star.text.TextCursor` admite todas las propiedades de carácter y párrafo presentadas al inicio de este capítulo.

En el ejemplo siguiente se muestra el uso de dichas propiedades junto con un `TextCursor`. Recorre un documento completo y formatea la primera palabra de cada frase con negrita.

```
Dim Doc As Object
Dim Cursor As Object
Dim Proseguir As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

      Cursor.gotoEndOfWord(True)
      Cursor.CharWeight = com.sun.star.awt.FontWeight.BOLD
      Proseguir = Cursor.gotoNextSentence(False)
      Cursor.gotoNextWord(False)

Loop While Proseguir
```

En el ejemplo se crea en primer lugar un objeto documento para el texto que se acaba de abrir. A continuación recorre iterativamente todo el texto, frase por frase, destaca todas las primeras palabras y las formatea en negrita.

Recuperación y modificación del contenido de texto

Si un `TextCursor` contiene un área de texto destacada, ese texto estará disponible a través de la propiedad `String` del objeto `TextCursor`. En el ejemplo siguiente se utiliza la propiedad `String` para mostrar las primeras palabras de una frase en un cuadro de mensaje:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proseguir As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

      Cursor.gotoEndOfWord(True)
      MsgBox Cursor.String
      Proseguir = Cursor.gotoNextSentence(False)
      Cursor.gotoNextWord(False)

Loop While Proseguir
```

De forma similar se puede modificar la primera palabra de cada frase mediante la propiedad `String`:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proseguir As Boolean

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor

Do

      Cursor.gotoEndOfWord(True)
      Cursor.String = "Arr"
      Proseguir = Cursor.gotoNextSentence(False)
      Cursor.gotoNextWord(False)

Loop While Proseguir
```

Si el objeto `TextCursor` contiene un área destacada, una asignación a la propiedad `String` sustituye el texto contenido en dicha área por el texto nuevo. Si no hay ningún área destacada, el texto se inserta en la posición actual de `TextCursor`.

Inserción de códigos de control

En algunas situaciones es necesario modificar la estructura del texto de un documento, no el texto en sí. StarOffice ofrece códigos de control para ello. Dichos códigos se insertan en el texto y alteran su estructura. Los códigos de control se definen en el grupo de constantes `com.sun.star.text.ControlCharacter`. StarOffice dispone de los siguientes códigos de control:

- **PARAGRAPH_BREAK**: salto de párrafo.
- **LINE_BREAK**: salto de línea dentro de un párrafo.
- **SOFT_HYPHEN**: punto posible de separación silábica.
- **HARD_HYPHEN**: punto obligatorio de separación silábica.
- **HARD_SPACE**: espacio protegido que no se expande ni se comprime en texto justificado.

Para insertar los códigos de control no sólo es necesario el cursor, sino también los objetos documento de texto asociados. En el ejemplo siguiente se inserta un párrafo después del vigésimo carácter de un texto:

```
Dim Doc As Object
Dim Cursor As Object
Dim Proseguir 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)
```

El parámetro `False` de la llamada al método `insertControlCharacter` garantiza que el área actualmente destacada por `TextCursor` se conserve después de la operación de inserción. Si se pasa el parámetro `True`, `insertControlCharacter` reemplazará el texto actual.

Búsqueda de fragmentos de texto

En muchas ocasiones se debe buscar un término específico dentro de un texto para, a continuación, editarlo en la posición correspondiente. Todos los documentos de StarOffice ofrecen una interfaz especial para ello cuyo funcionamiento se ajusta siempre al mismo principio: Antes de efectuar una búsqueda se debe crear lo que suele denominarse un `SearchDescriptor` (descriptor de búsqueda) que define lo que StarOffice busca en el documento. Un `SearchDescriptor` es un objeto que admite el servicio `com.sun.star.util.SearchDescriptor` y puede crearse mediante el método `createSearchDescriptor` de un documento:

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

Una vez creado el `SearchDescriptor`, éste recibe el texto que se debe buscar:

```
DescBusqueda.searchString="cualquier texto"
```

Según su funcionalidad, el `SearchDescriptor` es similar al diálogo de búsqueda de StarOffice. Los parámetros de configuración necesarios para efectuar una búsqueda se pueden establecer en el objeto `SearchDescriptor` de forma parecida a como se hace en la ventana de búsqueda.

El servicio `com.sun.star.util.SearchDescriptor` proporciona las propiedades:

- **SearchBackwards (booleano)**: busca hacia atrás en vez de hacia delante.
- **SearchCaseSensitive (booleano)**: distingue entre mayúsculas y minúsculas durante la búsqueda.
- **SearchRegularExpression (booleano)**: trata la expresión de búsqueda como expresión regular.
- **SearchStyles (booleano)**: busca en el texto la plantilla de párrafo especificada.
- **SearchWords (booleano)**: busca únicamente palabras completas.

La función de StarOffice `SearchSimilarity` (o •gbúsqueda difusa•h) está también disponible en StarOffice Basic. Con esta función, StarOffice busca una expresión que puede ser similar aunque no exactamente igual que la expresión de búsqueda. El número de caracteres adicionales, suprimidos y modificados para estas expresiones se puede definir de forma individual. Las propiedades asociadas del servicio `com.sun.star.util.SearchDescriptor` son:

- **SearchSimilarity (booleano)** efectúa una búsqueda de similitud.
- **SearchSimilarityAdd (Short)**: número de caracteres que se pueden agregar para una búsqueda de similitud.
- **SearchSimilarityExchange (Short)**: número de caracteres que se pueden reemplazar como parte de una búsqueda de similitud.
- **SearchSimilarityRemove (Short)**: número de caracteres que se pueden suprimir como parte de una búsqueda de similitud.
- **SearchSimilarityRelax (booleano)**: tiene en cuenta para la expresión de búsqueda todas las reglas de desviación simultáneamente.

Una vez preparado el `SearchDescriptor` como se ha solicitado se puede aplicar al documento de texto. Los documentos de StarOffice ofrecen para ello los métodos `findFirst` y `findNext`:

```
Encontrado = Doc.findFirst (DescBusqueda)

Do While Encontrado
      ' Editar resultados de búsqueda
      Encontrado = Doc.findNext( Encontrado.End, Search)
Loop
```

El ejemplo busca todas las apariciones mediante un bucle y devuelve un objeto `TextRange` que hace referencia al fragmento de texto encontrado.

Ejemplo: búsqueda por similitud

En este ejemplo se muestra cómo buscar la palabra "beneficio" en un texto y formatear los resultados en negrita. Se utiliza una búsqueda por similitud, que no sólo encuentra la palabra "beneficio", sino también su plural "beneficios" y variaciones como "beneficioso". Las expresiones encontradas pueden diferir de la expresión de búsqueda en un máximo de dos caracteres:

```
Dim DescBusqueda As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

DescBusqueda = Doc.createSearchDescriptor
DescBusqueda.SearchString="beneficio"
DescBusqueda.SearchSimilarity = True
DescBusqueda.SearchSimilarityAdd = 2
DescBusqueda.SearchSimilarityExchange = 2
DescBusqueda.SearchSimilarityRemove = 2
DescBusqueda.SearchSimilarityRelax = False

Encontrado = Doc.findFirst (DescBusqueda)

Do While Encontrado
Encontrado.CharWeight = com.sun.star.awt.FontWeight.BOLD
Encontrado = Doc.findNext( Encontrado.End, Search)
Loop
```

La idea básica de la función de buscar y reemplazar de StarOffice es parecida a la de VBA. Ambas interfaces ofrecen un objeto, a través del cuál se pueden definir las propiedades de buscar y reemplazar, que se aplica luego al área de texto requerida para efectuar la acción. Mientras que en VBA se puede acceder al objeto auxiliar responsable a través de la propiedad `Find` del objeto `Range`, en StarOffice Basic dicho objeto se crea mediante la llamada `createSearchDescriptor` o `createReplaceDescriptor` del objeto documento. Incluso las propiedades y métodos de búsqueda son distintos.

Igual que en la API antigua de StarOffice, en la nueva las operaciones de buscar y reemplazar texto se efectúan mediante el objeto documento. Mientras que anteriormente existía un objeto denominado `SearchSettings` especial para definir las opciones de búsqueda, las búsquedas en el nuevo objeto se efectúan utilizando un objeto `SearchDescriptor`, o un objeto `ReplaceDescriptor` para reemplazar texto automáticamente. Estos objetos no sólo contienen las opciones, sino también el texto de búsqueda actual y, si es necesario, el texto sustituto asociado. Los objetos descriptor se crean mediante el objeto documento, se completan según las necesidades específicas y se devuelven al objeto documento en forma de parámetros para los métodos de búsqueda.

Reemplazo de fragmentos de texto

Como sucede con la función de búsqueda, la de sustitución de StarOffice también está disponible en StarOffice Basic. Ambas se gestionan de forma idéntica. En un proceso de sustitución de texto también se necesita en primer lugar un objeto que almacene los parámetros del proceso. Dicho objeto se denomina `ReplaceDescriptor` (descriptor de sustitución) y admite el servicio `com.sun.star.util.ReplaceDescriptor`. Todas las propiedades de `SearchDescriptor` descritas en los párrafos anteriores se aplican también al `ReplaceDescriptor`. Por ejemplo, durante un proceso de sustitución también se puede activar la distinción entre mayúsculas y minúsculas, así como efectuar búsquedas por similitud.

En el ejemplo siguiente se muestra el uso de `ReplaceDescriptors` para una búsqueda dentro de un documento de StarOffice.

```
Dim I As Long
Dim Doc As Object
Dim Reemplazar As Object
Dim PalabrasBritanicas(5) As String
Dim PalabrasEEUU(5) As String

PalabrasBritanicas() = Array("colour", "neighbour", "centre", "behaviour", _
      "metre", "through")

PalabrasEEUU() = Array("color", "neighbor", "center", "behavior", _
      "meter", "thru")

Doc = StarDesktop.CurrentComponent
Reemplazar = Doc.createReplaceDescriptor

For O = 0 To 5
      Reemplazar.SearchString = PalabrasBritanicas(I)
      Reemplazar.ReplaceString = PalabrasEEUU(I)
      Doc.replaceAll(Reemplazar)
Next n
```

Las expresiones para buscar y reemplazar se establecen mediante las propiedades `SearchString` y `ReplaceString` de los `ReplaceDescriptors`. El proceso de sustitución en sí se implementa finalmente mediante el método `replaceAll` del objeto documento, que sustituye todas las apariciones de la expresión de búsqueda.

Ejemplo: buscar y reemplazar texto con expresiones regulares

La función de sustitución de StarOffice es especialmente eficaz si se utiliza con expresiones regulares que ofrecen la opción de definir una expresión de búsqueda variable con comodines y caracteres especiales, en lugar de con un valor fijo.

Las expresiones regulares admitidas por StarOffice se describen con detalle en la sección de ayuda en línea de StarOffice. Éstos son algunos ejemplos:

- Un punto dentro de una expresión de búsqueda significa cualquier carácter. La expresión de búsqueda `car.a` puede así significar tanto `carpa` como `carta`.

- El carácter ^ marca el inicio de un párrafo. Así, se pueden buscar todas las apariciones del nombre `Pedro` al principio de un párrafo mediante la expresión de búsqueda `^Pedro`.
- El carácter $ marca un final de párrafo. Así, se pueden buscar todas las apariciones del nombre `Pedro` al final de un párrafo mediante la expresión de búsqueda `Pedro$`.
- El signo * indica que el carácter anterior puede estar repetido cualquier número de veces. Se puede combinar con el punto como comodín para cualquier carácter. Por ejemplo, la expresión `temp.*a` puede significar tanto `templanza` como `temperatura`.

En el ejemplo siguiente se muestra cómo suprimir las líneas vacías de un documento de texto con la ayuda de la expresión regular ^$:

```
Dim Doc As Object
Dim Reemplazar As Object
Dim I As Long

Doc = StarDesktop.CurrentComponent
Reemplazar = Doc.createReplaceDescriptor

Reemplazar.SearchRegularExpression = True
Reemplazar.SearchString = "^$"
Reemplazar.ReplaceString = ""

Doc.replaceAll(Reemplazar)
```

Documentos de texto: no sólo texto

Hasta ahora, en este capítulo se ha tratado únicamente de párrafos de texto y fragmentos de párrafos. Pero los documentos de texto también pueden contener otros objetos, como, tablas, dibujos, campos de texto y directorios. Y todos ellos pueden estar situados (anclados) en cualquier lugar dentro del texto.

Gracias a estas características comunes, todos estos objetos de StarOffice admiten un servicio básico común denominado `com.sun.star.text.TextContent` con las propiedades siguientes:

- **`AnchorType (Enum)`**: determina el tipo de anclaje de un objeto `TextContent` (los valores predeterminados según la enumeración `com.sun.star.text.TextContentAnchorType`).
- **`AnchorTypes (secuencia de Enum)`**: enumeración de todos los `AnchorTypes` que admiten un objeto `TextContent` específico.
- **`TextWrap (Enum)`**: determina el tipo de ajuste de texto alrededor de un objeto `TextContent` (valores predeterminados según la enumeración `com.sun.star.text.WrapTextMode`).

Los objetos `TextContent` comparten también algunos métodos (específicamente, métodos para crear, insertar y borrar objetos).

- Para crear un nuevo objeto `TextContent` se usa el método `createInstance` del objeto documento.
- Para **insertar** un objeto se emplea el método `insertTextContent` del objeto de texto.
- `Los objetos TextContent` se **borran** mediante el método `removeTextContent`.

En las secciones siguientes se presentan diversos ejemplos de uso de dichos métodos.

Tablas

En el ejemplo siguiente se crea una tabla con la ayuda del método `createInstance` descrito anteriormente.

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

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

Tabla = Doc.createInstance("com.sun.star.text.TextTable")
Tabla.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Tabla, False)
```

Una vez creada la tabla, se establece el número de filas y columnas solicitado mediante una llamada a `initialize` y a continuación se inserta en el documento de texto mediante `insertTextContent`.

Como se puede ver en el ejemplo, el método `insertTextContent` no sólo espera el objeto `Content` que se debe insertar, sino dos parámetros más:

- un objeto `Cursor` que determina la posición de inserción
- una variable booleana que especifica si el objeto `Content` debe sustituir la actual selección del cursor (valor `True`) o si se debe insertar antes de la selección actual (`False`)

Al crear e insertar tablas en un documento de texto, StarOffice Basic utiliza objetos similares a los disponibles en VBA: el objeto documento y un objeto `TextCursor` en StarOffice Basic o el objeto `Range`, la contrapartida en VBA. Mientras que el método `Document.Tables.Add` asume la tarea de crear y configurar la tabla en VBA, en StarOffice Basic ésta se crea, como se muestra en el ejemplo anterior, mediante `createInstance` y se inicializa e inserta en el documento con `insertTextContent`.

Las tablas insertadas en un documento de texto se pueden determinar mediante un bucle simple. Para ello se utiliza el método `getTextTables()` del objeto documento:

```
Dim Doc As Object
Dim TablasTexto As Object
Dim Tabla As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
TablasTexto = Doc.getTextTables()

For I = 0 To TablasTexto.count - 1
      Tabla = TablasTexto(I)

      ' Editando tabla
Next I
```

Las tablas de texto están disponibles en StarOffice 6 a través de la lista `TextTables` del objeto documento. Esta lista sustituye a la antigua lista de tablas que proporcionaba el objeto `Selection`. En el ejemplo anterior se muestra cómo crear una tabla de texto. Las opciones de acceso a las tablas de texto se describen en la sección siguiente.

Edición de tablas

Una tabla consta de filas individuales que, a su vez, contienen las diversas celdas. Estrictamente, en StarOffice no hay columnas de tabla. Ésta se generan de forma implícita al disponer las filas (una debajo de otra) juntas. No obstante, para simplificar el acceso a las tablas, StarOffice ofrece algunos métodos que funcionan con columnas y que son útiles si la tabla no contiene celdas unidas.

Consideremos en primer lugar las propiedades de la tabla en sí. Éstas se definen en el servicio `com.sun.star.text.TextTable`. He aquí una lista de las propiedades más importantes del objeto tabla:

- **`BackColor (Long)`:** color del fondo.
- **`BottomMargin (Long)`:** margen inferior en centésimas de milímetro.
- **`LeftMargin (Long)`:** margen izquierdo en centésimas de milímetro.
- **`RightMargin (Long)`:** margen derecho en centésimas de milímetro.
- **`TopMargin (Long)`:** margen superior en centésimas de milímetro.
- **`RepeatHeadline (booleano)`**: el encabezamiento de la tabla se repite en cada página.
- **`Width (Long)`:** ancho absoluto de la tabla en centésimas de milímetro.

Filas

Una tabla consiste en una lista de filas. En el ejemplo siguiente se muestra cómo recuperar y dar formato a las filas de una tabla.

```
Dim Doc As Object
Dim Tabla As Object
Dim Cursor As Object
Dim Filas As Object
Dim Fila As Object
Dim I As Integer
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Tabla = Doc.createInstance("com.sun.star.text.TextTable")
Tabla.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Tabla, False)
Filas = Tabla.getRows
For I = 0 to Filas.getCount() - 1
      Fila = Filas.getByIndex(I)
      Fila.BackColor = &HFF00FF
Next
```

En el ejemplo se crea en primer lugar una lista con todas filas mediante una llamada a `Table.getRows`. Los métodos `getCount` y `getByIndex` de la interfaz `com.sun.star.table.XtableRows` permiten procesar la lista. El método `getByIndex` devuelve un objeto fila que admite el servicio `com.sun.star.text.TextTableRow`.

Estos son los métodos esenciales de la interfaz `com.sun.star.table.XtableRows`:

- **`getByIndex(Integer)`**: devuelve un objeto fila para el índice especificado.
- **`getCount()`**: devuelve el número de objetos fila.
- **`insertByIndex(índice, número)`**: inserta número filas en la tabla a partir de la posición `índice`.
- **`removeByIndex(índice, número)`**: borra número filas de la tabla a partir de la posición `índice`.

Mientras que los métodos `getByIndex` y `getCount` están disponibles para todas las tablas, los métodos `insertByIndex` y `removeByIndex` sólo se pueden utilizar en tablas que no contengan celdas unidas.

El servicio `com.sun.star.text.TextTableRow` ofrece las siguientes propiedades:

- **`BackColor (Long)`**: color de fondo de la fila.
- **`Height (Long)`**: altura de la fila en centésimas de milímetro.
- **`IsAutoHeight (booleano)`**: la altura de la tabla se adapta al contenido de forma dinámica.
- **`VertOrient (const)`**: orientación vertical del marco de texto; detalles de la orientación vertical del texto dentro de la tabla (valores de acuerdo con `com.sun.star.text.VertOrientation`)

Columnas

El acceso a las columnas, como en el caso de las filas, se lleva a cabo mediante los métodos `getByIndex`, `getCount`, `insertByIndex` y `removeByIndex` del objeto `Column`, al que se accede a través de `getColumns`. No obstante, dichos métodos sólo pueden emplearse en tablas que no contengan celdas unidas. En StarOffice Basic no es posible formatear las celdas por columnas. Para ello se deben formatear las celdas de forma individual.

Celdas

Cada una de las celdas de un documento de StarOffice tiene un nombre exclusivo. Si el cursor de StarOffice se encuentra en una celda, en la barra de estado se puede leer el nombre de la misma. La celda superior izquierda se suele denominar `A1` y la inferior derecha se suele denominar `Xn`, siendo `X` la letra correspondiente a la última columna y `n` el número correspondiente a la última fila. Los objetos celda están disponibles a través del método `getCellByName()` del objeto tabla. En el ejemplo siguiente se muestra un bucle que recorre todas las celdas de una tabla y escribe en cada una de ellas las coordenadas de fila y columna correspondientes.

```
Dim Doc As Object
Dim Tabla As Object
Dim Cursor As Object
Dim Filas As Object
Dim IndiceFila As Integer
Dim Columnas As Object
Dim IndiceColumna As Integer
Dim NombreCelda As String
Dim Celda As Object

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

Tabla = Doc.createInstance("com.sun.star.text.TextTable")
Tabla.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Tabla, False)

Filas = Tabla.getRows
Columnas = Tabla.getColumns

For IndiceFila = 1 to Filas.getCount()
      For IndiceColumna = 1 To Columnas.getCount()
            NombreCelda = Chr(64 + IndiceColumna) & IndiceFila
            Celda = Tabla.getCellByName(NombreCelda)
            Celda.String = "fila: " & CStr(IndiceFila) + ", columna: " & CStr
(IndiceColumna)
      Next
Next
```

La celda de una tabla es similar a un texto estándar. Admite la interfaz `createTextCursor` para crear un objeto `TextCursor` asociado.

```
CursorCelda = Celda.createTextCursor()
```

Por consiguiente, todas las opciones de formato para caracteres individuales y párrafos están automáticamente disponibles.

En el ejemplo siguiente se efectúa una búsqueda a través de todas las tablas de un documento de texto y se aplica el formato alineación derecha a todas las celdas con valores numéricos, mediante la propiedad de párrafo correspondiente.

```
Dim Doc As Object
Dim TablasTexto As Object
Dim Tabla As Object
Dim NombresCelda
Dim Celda As Object
Dim CursorCelda As Object
Dim I As Integer
Dim J As Integer

Doc = StarDesktop.CurrentComponent
```

```
TablasTexto = Doc.getTextTables()

For I = 0 To TablasTexto.count - 1
      Tabla = TablasTexto(I)
      NombresCelda = Tabla.getCellNames()

      For J = 0 to UBound(NombresCelda)
            Celda = Tabla.getCellByName(NombresCelda(J))
            If IsNumeric(Celda.String) Then
                  CursorCelda = Celda.createTextCursor()
                  CursorCelda.paraAdjust = com.sun.star.style.ParagraphAdjust.RIGHT
            End If
      Next
Next
```

En el ejemplo se crea una lista `TextTables` que contiene todas las tablas de un texto que se recorren en un bucle. StarOffice crea a continuación una lista con los nombres de celda asociados para cada una de dichas tablas. Las celdas se recorren mediante un bucle. Si una celda contiene un valor numérico, el ejemplo cambia el formato de la misma. Para ello crea en primer lugar un objeto `TextCursor` que hace referencia al contenido de la celda de la tabla y a continuación adapta las propiedades de párrafo de la celda.

Marcos de texto

Los marcos de texto se consideran objetos `TextContent`, como las tablas y las imágenes. Pueden constar esencialmente de texto estándar, pero es posible situarlos en cualquier lugar de la página y no se incluyen en el flujo del texto.

Como sucede con todos los objetos `TextContent`, en los marcos de texto se debe distinguir entre la creación propiamente dicha y la inserción en el documento.

```
Dim Doc As Object
Dim TablasTexto As Object
Dim Cursor As Object
Dim Marco As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Marco = Doc.createInstance("com.sun.star.text.TextFrame")

Doc.Text.insertTextContent(Cursor, Marco, False)
```

El marco de texto se crea mediante el método `createInstance` del objeto documento. El marco de texto así creado se puede insertar en el documento mediante el método `insertTextContent` del objeto `Text`. Al hacerlo se debe especificar el nombre del servicio `com.sun.star.text.TextFrame` adecuado.

Un objeto `Cursor`, que se ejecuta también al insertarse, determina la posición de inserción del marco de texto.

Los marcos de texto son el equivalente en StarOffice del marco de posición utilizado en Word. Mientras que VBA utiliza para ello el método `Document.Frames.Add`, la creación se lleva a cabo mediante el procedimiento indicado anteriormente, con la ayuda de un objeto TextCursor así como del método `createInstance` del objeto documento.

Los objetos marco de texto ofrecen diversas propiedades que afectan a la posición y al comportamiento del marco. La mayoría de ellas se definen en el servicio `com.sun.star.text.BaseFrameProperties` que también admite cada uno de los servicios TextFrame. Las propiedades fundamentales son:

- **`BackColor (Long)`**: color de fondo del marco de texto.
- **`BottomMargin (Long)`**: margen inferior en centésimas de milímetro.
- **`LeftMargin (Long)`**: margen izquierdo en centésimas de milímetro.
- **`RightMargin (Long)`**: margen derecho en centésimas de milímetro.
- **`TopMargin (Long)`**: margen superior en centésimas de milímetro.
- **`Height (Long)`**: altura del marco de texto en centésimas de milímetro.
- **`Width (Long)`**: ancho del marco de texto en centésimas de milímetro.
- **`HoriOrient (const)`**: orientación horizontal del marco de texto (de acuerdo con `com.sun.star.text.HoriOrientation`).

`VertOrient (const)`: orientación vertical del marco de texto (de acuerdo con `com.sun.star.text.VertOrientation`).

En el ejemplo siguiente se crea un marco de texto utilizando las propiedades mencionadas:

```
Dim Doc As Object
Dim TablasTexto As Object
Dim Cursor As Object
Dim Marco As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Cursor.gotoNextWord(False)
Marco = Doc.createInstance("com.sun.star.text.TextFrame")

Marco.Width = 3000
Marco.Height = 1000
Marco.AnchorType = com.sun.star.text.TextContentAnchorType.AS_CHARACTER
Marco.TopMargin = 0
Marco.BottomMargin = 0
Marco.LeftMargin = 0
Marco.RightMargin = 0
Marco.BorderDistance = 0
Marco.HoriOrient = com.sun.star.text.HoriOrientation.NONE
Marco.VertOrient = com.sun.star.text.VertOrientation.LINE_TOP

Doc.Text.insertTextContent(Cursor, Marco, False)
```

En el ejemplo se crea un `TextCursor` como marca de inserción para el marco de texto que se sitúa entre la primera y la segunda palabra del texto. El marco de texto se crea utilizando `Doc.createInstance`. Las propiedades de los objetos marco de texto se establecen en los valores iniciales pertinentes.

Es conveniente observar la interacción entre las propiedades `AnchorType` (del servicio `TextContent`) y `VertOrient` (del servicio `BaseFrameProperties`). `AnchorType` recibe el valor `AS_CHARACTER`. Por consiguiente, el marco de texto se inserta directamente en el flujo de texto y se comporta como un carácter. Puede, por ejemplo, desplazarse a la línea siguiente en caso de salto de línea. El valor `LINE_TOP` de la propiedad `VertOrient` garantiza que el borde superior del marco de texto se encuentre situado a la misma altura que el borde superior del carácter.

Una vez completada la inicialización, el marco de texto se inserta en el documento de texto mediante una llamada a `insertTextContent`.

Para editar el contenido de un marco de texto, el usuario utiliza el objeto `TextCursor,` que ya se ha tratado en numerosas ocasiones y que está también disponible en los marcos de texto.

```
Dim Doc As Object
Dim TablasTexto As Object
Dim Cursor As Object
Dim Marco As Object
Dim CursorMarco As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Marco = Doc.createInstance("com.sun.star.text.TextFrame")

Marco.Width = 3000
Marco.Height = 1000

Doc.Text.insertTextContent(Cursor, Marco, False)

CursorMarco = Marco.createTextCursor()
CursorMarco.CharWeight = com.sun.star.awt.FontWeight.BOLD
CursorMarco.paraAdjust = com.sun.star.style.ParagraphAdjust.CENTER
CursorMarco.String = " Esto es una pequeña prueba"
```

En el ejemplo se crea un marco de texto, se inserta en el documento actual y se abre un `TextCursor` para dicho marco. Este cursor se utiliza para poner en negrita la fuente del marco y para establecer en centrada la alineación del párrafo. Finalmente se asigna al marco de texto la cadena **•gEsto es una pequeña prueba•h.**

Campos de texto

Los campos de texto son objetos `TextContent` porque proporcionan caracteres lógicos adicionales que van más allá del puro texto. Los campos de texto se pueden insertar en un documento de texto con los mismos métodos utilizados para otros objetos `TextContent`:

```
Dim Doc As Object
Dim CampoFechaHora As Object
Dim Cursor As Object
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

CampoFechaHora = Doc.createInstance("com.sun.star.text.TextField.DateTime")
CampoFechaHora.IsFixed = False
CampoFechaHora.IsDate = True
Doc.Text.insertTextContent(Cursor, CampoFechaHora, False)
```

En el ejemplo se inserta un campo de texto con la fecha actual al principio del documento de texto actual. El valor `True` de la propiedad `IsDate` hace que se muestre únicamente la fecha, no la hora. El valor `False` de `IsFixed` garantiza la actualización automática de la fecha al abrir el documento.

Mientras que en VBA el tipo de un campo se especifica mediante un parámetro del método `Document.Fields.Add`, el nombre del servicio responsable del tipo de campo en cuestión es el que lo define en StarOffice Basic.

Anteriormente, el acceso a los campos de texto se efectuaba mediante numerosos métodos que StarOffice ofrecía a través del antiguo objeto `Selection` (por ejemplo, `InsertField`, `DeleteUserField`, `SetCurField`).

En StarOffice 6, el concepto de administración de los campos es orientado al objeto. Para crear un campo de texto, se debe crear e inicializar en primer lugar un campo de texto utilizando las propiedades requeridas. A continuación dicho campo de texto se inserta en el documento mediante el método `insertTextContent`. En el ejemplo anterior se puede ver un texto fuente correspondiente. En las secciones siguientes se describen los tipos de campo más importantes y sus propiedades.

Además de insertar campos de texto, a veces es importante buscar campos en un documento. En el ejemplo siguiente se muestra cómo recorrer todos los campos de texto de un documento mediante un bucle y comprobar de qué tipo son.

```
Dim Doc As Object
Dim EnumCamposTexto As Object
Dim CampoTexto As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent

EnumCamposTexto = Doc.getTextFields.createEnumeration

While EnumCamposTexto.hasMoreElements()

      CampoTexto = EnumCamposTexto.nextElement()

      If CampoTexto.supportsService("com.sun.star.text.TextField.DateTime") Then
            MsgBox "Fecha/hora"
      ElseIf CampoTexto.supportsService("com.sun.star.text.TextField.Annotation") Then
```

```
            MsgBox "Anotación"
      Else
            MsgBox "desconocido"
      End If

Wend
```

El punto de partida para establecer qué campos de texto están presentes es la lista `TextFields` del objeto documento. El ejemplo crea un objeto `Enumeration` a partir de dicha lista que permite consultar todos los campos de texto uno por uno en un bucle. El servicio admitido por los campos de texto encontrados se comprueba mediante el método `supportsService`. Si el campo es de fecha / hora o de anotación, el tipo del campo se muestra en un cuadro de información. Si, por el contrario, el ejemplo encuentra un campo de otro tipo, la información que se muestra es • gdesconocido•h.

A continuación se enumeran los tipos de campos de texto más importantes y sus propiedades asociadas. Para ver una lista completa de todos los campos consulte, en la referencia de la API, el módulo `com.sun.star.text.TextField`. (En el nombre de servicio de un campo de texto en StarOffice Basic se deben usar mayúsculas y minúsculas, como aparecía en el ejemplo anterior.)

Número de páginas, palabras y caracteres

Los campos de texto

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

devuelven el número de páginas, palabras o caracteres de un texto. Admiten la propiedad siguiente:

- **`NumberingType (const)`**: formato de numeración (directrices de acuerdo con las constantes de `com.sun.star.style.NumberingType`).

Página actual

El número de la página actual se puede insertar en un documento mediante el campo de texto `com.sun.star.text.TextField.PageNumber`. Se pueden especificar las propiedades siguientes:

- **`NumberingType (const)`** : formato de numeración (directrices de acuerdo con las constantes de `com.sun.star.style.NumberingType`).
- **`Offset (short)`**: desplazamiento sumado al número de páginas (se puede especificar un valor negativo).

En el ejemplo siguiente se muestra cómo insertar el número de páginas en el pie de un documento.

```
Dim Doc As Object
Dim CampoFechaHora As Object
Dim EstilosPagina As Object
Dim PaginaEstandar As Object
```

```
Dim CursorPie As Object
Dim NumeroPagina As Object

Doc = StarDesktop.CurrentComponent

NumeroPagina = Doc.createInstance("com.sun.star.text.TextField.PageNumber")
NumeroPagina.NumberingType = com.sun.star.style.NumberingType.ARABIC

EstilosPagina = Doc.StyleFamilies.getByName("PageStyles")

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

CursorPie = StdPage.FooterTextLeft.Text.createTextCursor()
PaginaEstandar.FooterTextLeft.Text.insertTextContent(CursorPie, NumeroPagina, False)
```

En el ejemplo se crea en primer lugar un campo de texto que admite el servicio `com.sun.star.text.TextField.PageNumber`. Puesto que las líneas de encabezamiento y pie están definidas como parte de las plantillas de página de StarOffice, se establece en primer lugar mediante la lista `PageStyles`.

Para garantizar que la línea de pie de página esté visible, la propiedad `FooterIsOn` se establece en `True`. A continuación se inserta el campo de texto en el documento mediante el objeto de texto asociado de la línea de pie de la izquierda.

Anotaciones

Los campos de anotación (`com.sun.star.text.TextField.Annotation`) se indican mediante un pequeño símbolo amarillo en el texto. Al hacer clic en dicho símbolo se abre un campo de texto, en el que se puede almacenar un comentario relativo al punto actual del texto. Las propiedades de un campo de anotación son:

- **`Author (String)`**: nombre del autor.
- **`Content (String)`**: texto del comentario.
- **`Date (Date)`**: fecha en la que se efectúa la anotación.

Fecha / hora

Un campo de fecha / hora (`com.sun.star.text.TextField.DateTime`) representa la fecha o la hora actuales. Admite las propiedades siguientes:

- **`IsFixed (booleano)`**: si es `True`, los detalles temporales de la inserción permanecen sin cambios; si es False, dichos detalles se actualizan cada vez que se abre el documento.
- **`IsDate (booleano)`**: si es `True`, el campo muestra la fecha actual; sino, muestra la hora actual.
- **`DateTimeValue (struct)`** : contenido actual del campo (estructura `com.sun.star.util.DateTime`)
- **`NumberFormat (const)`**: formato en el que se muestra la fecha u hora.

Nombre / número de capítulo

El nombre del capítulo actual está disponible en un campo de texto de tipo `com.sun.star.text.TextField.Chapter`. El formato se puede definir mediante dos propiedades.

- **`ChapterFormat (const)`**: determina si se muestra el nombre del capítulo o su número (de acuerdo con `com.sun.star.text.ChapterFormat`)
- **`Level (Integer)`**: determina el nivel del capítulo cuyo nombre o número debe mostrarse. El valor 0 representa el máximo nivel disponible.

Marcadores

Los marcadores (servicio `com.sun.star.text.Bookmark`) son objetos `TextContent`. Los marcadores se crean e insertan utilizando el concepto descrito anteriormente:

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

Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor()

Marcador = Doc.createInstance("com.sun.star.text.Bookmark")
Marcador.Name = "Mis marcadores"
Doc.Text.insertTextContent(Cursor, Marcador, True)
```

En el ejemplo se crea un `Cursor` que marca la posición de inserción del marcador y, a continuación, el objeto marcador en sí (`Bookmark`). A continuación se asigna al marcador un nombre y se inserta en el documento en la posición del cursor mediante `insertTextContent`.

Para acceder a los marcadores de un texto se utiliza la lista denominada `Bookmarks`. Se puede acceder a ellos por número o por nombre.

En este ejemplo se muestra cómo buscar un marcador dentro de un texto y cómo insertar texto en su posición.

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

Doc = StarDesktop.CurrentComponent

Marcador = Doc.Bookmarks.getByName("Mis marcadores")

Cursor = Doc.Text.createTextCursorByRange(Marcador.Anchor)
Cursor.String = "Aquí está el marcador"
```

En este ejemplo se utiliza el método `getByName` para buscar el marcador requerido a partir de su nombre. A continuación, la llamada `createTextCursorByRange` crea un `Cursor` que se sitúa en la posición de anclaje del marcador. Finalmente, el cursor inserta en ese punto el texto requerido.

CAPÍTULO 7

Documentos de hoja de cálculo

StarOffice Basic ofrece una amplia interfaz para la creación y edición de hojas de cálculo por control de programa. En este capítulo se describe el control de los servicios, métodos y propiedades pertinentes de los documentos de la hoja de cálculo.

En la primera sección se comenta la estructura básica de los documentos de la hoja de cálculo y se muestra cómo acceder al contenido de las celdas individuales y editarlo.

La segunda sección se centra en cómo editar hojas de cálculo de la forma más eficiente a partir del concepto de área de celdas y en cómo buscar y reemplazar el contenido de las celdas.

El objeto Range permite designar cualquier área de la tabla; en la nueva API se ha ampliado.

La estructura de los documentos basados en tablas (hojas de cálculo)

El objeto documento de una hoja de cálculo se basa en el servicio `com.sun.star.sheet.SpreadsheetDocument`. Cada uno de estos documentos puede contener varias hojas. En este manual, los términos ***documento basado en tablas*** o ***documento hoja de cálculo*** se refieren al documento completo, mientras que ***hoja de cálculo*** (u ***hoja*** para abreviar) hace referencia a una hoja (tabla) del documento.

VBA y StarOffice Basic utilizan una terminología distinta para las hojas de cálculo y su contenido. Mientras que en VBA el objeto documento se denomina *Workbook* (libro de trabajo) y sus páginas individuales *Worksheets* (hojas de trabajo), en StarOffice Basic se denominan *SpreadsheetDocument* (documento hoja de cálculo) y *Sheet* (hoja).

Hojas de cálculo

Se puede acceder a las hojas individuales de un documento mediante la lista `Sheets`.

En los ejemplos siguientes se muestra cómo acceder a una hoja a través de su número o de su nombre.

Ejemplo 1: acceso a través del número (la numeración empieza en 0)

```
Dim Doc As Object
Dim Hoja As Object

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

Ejemplo 2: acceso a través del nombre

```
Dim Doc As Object
Dim Hoja As Object

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

En el primer ejemplo se accede a la hoja a través de su número (la numeración empieza en 0). En el segundo ejemplo se accede por su nombre mediante el método `getByName`.

El objeto `Sheet` que obtiene el método `getByName` admite el servicio `com.sun.star.sheet.Spreadsheet`. Aparte de proporcionar diversas interfaces para editar el contenido, este servicio ofrece las propiedades siguientes:

- **`IsVisible (booleano)`**: la hoja de cálculo es visible.
- **`PageStyle (String)`**: nombre de la plantilla de página de la hoja de cálculo.

Creación, borrado y cambio de nombre de las hojas

La lista `Sheets` de un documento `hoja de cálculo` se utiliza también para crear, borrar y cambiar el nombre de hojas individuales. En el ejemplo siguiente se utiliza el método `hasByName` para comprobar si existe una hoja denominada ***MiHoja***. En tal caso, el método determina la referencia de objeto correspondiente mediante el método `getByName` y guarda dicha referencia en una variable `Hoja`. Si la hoja correspondiente no existe, se crea mediante la llamada `createInstance` y se inserta en el documento hoja de cálculo mediante el método `insertByName`.

```
Dim Doc As Object
Dim Hoja As Object

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

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

Los métodos `getByName` e `insertByName` pertenecen a la interfaz `com.sun.star.container.XnameContainer` descrita en el capítulo 4.

Filas y columnas

Cada hoja contiene una lista de sus filas y columnas a las que se puede acceder a través de las propiedades `Rows` y `Columns` del objeto hoja de cálculo; admiten los servicios `com.sun.star.table.TableColumns` o `com.sun.star.table.TableRows`.

En el ejemplo siguiente se crean dos objetos que hacen referencia a la primera fila y a la primera columna de una hoja y se almacenan las referencias en las variables de objeto `PrimeraColumna` y `PrimeraFila`.

```
Dim Doc As Object
Dim Hoja As Object
Dim PrimeraFila As Object
Dim PrimeraColumna As Object

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

PrimeraColumna = Hoja.Columns(0)
PrimeraFila = Hoja.Rows(0)
```

Los objetos columna admiten el servicio `com.sun.star.table.TableColumn` que tiene las propiedades siguientes:

- **`Width (Long)`:** ancho de una columna en centésimas de milímetro.
- **`OptimalWidth (booleano)`**: establece el ancho óptimo de una columna.
- **`IsVisible (booleano)`**: muestra la columna.
- **`IsStartOfNewPage (booleano)`**: al imprimir efectúa un salto de página antes de una columna.

El ancho de una columna sólo se optimiza si se establece la propiedad `OptimalWidth` en `True`. Si se modifica el ancho de una celda individual, el ancho de la columna que la contiene no se modifica. En términos de su función, `OptimalWidth` es más un método que una propiedad.

Los objetos fila se basan en el servicio `com.sun.star.table.TableRow` que tiene las propiedades siguientes:

- **`Height (Long)`:** altura de la fila en centésimas de milímetro.
- **`OptimalHeight (booleano)`**: establece la altura óptima de una fila.
- **`IsVisible (booleano)`**: muestra la fila.
- **`IsStartOfNewPage (booleano)`**: al imprimir efectúa un salto de página antes de la fila.

Si se establece la propiedad `OptimalHeight` de una fila en `True`, la altura de ésta cambia automáticamente al modificar la altura de una celda de la fila. La optimización automática prosigue mientras no se asigne a la fila una altura absoluta mediante la propiedad `Height`.

En el ejemplo siguiente se activa la optimización automática de altura para las cinco primeras filas de la hoja y se hace invisible la segunda columna.

```
Dim Doc As Object
Dim Hoja As Object
Dim Fila As Object
Dim Columna As Object
Dim I As Integer

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

For I = 0 To 4
      Fila = Hoja.Rows(I)
      Fila.OptimalHeight = True
Next I

Columna = Hoja.Columns(1)
Columna.IsVisible = False
```

En StarOffice Basic se puede acceder a las listas `Rows` y `Columns` mediante un índice. A diferencia de VBA, el índice de la primera columna es 0 y no 1.

Inserción y borrado de filas y columnas

Los objetos `Rows` y `Columns` de una hoja permiten tanto acceder a filas y columnas como insertarlas y borrarlas.

```
Dim Doc As Object
Dim Hoja As Object
Dim NuevaColumna As Object

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

Hoja.Columns.insertByIndex(3, 1)
Hoja.Columns.removeByIndex(5, 1)
```

Este ejemplo utiliza el método `insertByIndex` para insertar una columna nueva en la cuarta posición de la hoja (índice 3; la numeración se inicia en 0). El segundo parámetro especifica el número de columnas que se debe insertar (en el ejemplo, una).

El método `removeByIndex` borra la sexta columna (índice 5). De nuevo, el segundo parámetro especifica el número de columnas que se desea borrar.

Los métodos para insertar y borrar filas utilizan la función objeto `Rows` de la misma forma que los métodos para la edición de columnas emplean el objeto `Columns`.

Celdas

Una hoja de cálculo consiste en una lista bidimensional de celdas, cada una de ellas definida por sus posiciones X e Y con respecto a la celda superior izquierda, cuyas coordenadas de posición son (0,0).

En el ejemplo siguiente se crea un objeto que hace referencia a la celda superior izquierda y se inserta un texto en la celda:

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object

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

Celda = Hoja.getCellByPosition(0, 0)
Celda.String = "Prueba"
```

Además de las coordenadas numéricas, cada una de las celdas de una hoja tiene un nombre; por ejemplo, la celda superior izquierda (0,0) de una hoja de cálculo se denomina `A1`. La letra `A` significa la columna y el número `1` la fila. Es importante no confundir el ***nombre*** y la ***posición*** de una celda porque la numeración de las filas empieza en el 1, pero la numeración de la posición empieza en el 0.

En StarOffice, una celda de una tabla puede estar vacía o contener texto, números o fórmulas. El contenido almacenado en la celda no determina el tipo de ésta; lo establece la propiedad del objeto utilizado para introducir aquél. Los números se pueden insertar y recuperar mediante la propiedad `Value`, el texto utiliza la propiedad `String` y las fórmulas la propiedad `Formula`.

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object

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

Celda = Hoja.getCellByPosition(0, 0)
Celda.Value = 100

Celda = Hoja.getCellByPosition(0, 1)
Celda.String = "Prueba"

Celda = Hoja.getCellByPosition(0, 2)
Celda.Formula = "=A1"
```

En el ejemplo se inserta un número, un texto y una fórmula en los campos A1 al A3.

Las propiedades `Value`, `String` y `Formula` reemplazan al método `PutCell` para establecer los valores de una celda.

StarOffice trata el contenido de una celda introducido mediante la propiedad `String` como texto, aunque dicho contenido sea un número. Los números introducidos de esta forma están alineados a

la izquierda en lugar de a la derecha. Se debe tener en cuenta también la diferencia entre texto y números al utilizar fórmulas:

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object

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

Celda = Hoja.getCellByPosition(0, 0)
Celda.Value = 100

Celda = Hoja.getCellByPosition(0, 1)
Celda.String = 1000

Celda = Hoja.getCellByPosition(0, 2)
Celda.Formula = "=A1+A2"

MsgBox Celda.Value
```

Aunque la celda A1 contiene el valor 100 y la celda A2 contiene el valor 1000, la fórmula A1+A2 devuelve el valor 100, debido a que el contenido de la celda A2 se ha introducido como cadena, no como número.

Para comprobar si una celda contiene un número o una cadena, utilice la propiedad `Type`:

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets(0)
Celda = Hoja.getCellByPosition(1,1)

Celda.Value = 1000

Select Case Celda.Type
Case com.sun.star.table.CellContentType.EMPTY
      MsgBox "Contenido:" Vacío"
Case com.sun.star.table.CellContentType.VALUE
      MsgBox "Contenido:" Valor"
Case com.sun.star.table.CellContentType.TEXT
      MsgBox "Contenido:" Texto"
Case com.sun.star.table.CellContentType.FORMULA
      MsgBox "Contenido:" Fórmula"
End Select
```

La propiedad `Celda.Type` devuelve un valor para la enumeración `com.sun.star.table.CellContentType` que identifica el tipo de contenido de una celda. Los valores posibles son:

- **`EMPTY`**: ningún valor

- **VALUE**: número
- **TEXT**: cadena de caracteres
- **FORMULA**: fórmula

Inserción, borrado, copia y desplazamiento de celdas

Aparte de modificar directamente el contenido de la celda, StarOffice Basic ofrece también una interfaz para insertar, borrar, copiar o unir celdas. La interfaz (`com.sun.star.sheet.XRangeMovement`) está disponible a través del objeto hoja de cálculo y ofrece cuatro métodos para modificar el contenido de las celdas.

El método `insertCell` se utiliza para insertar celdas en una hoja.

```
Dim Doc As Object
Dim Hoja As Object
Dim DireccionAreaCeldas As New com.sun.star.table.CellRangeAddress

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

DireccionAreaCeldas.Sheet = 0
DireccionAreaCeldas.StartColumn = 1
DireccionAreaCeldas.StartRow = 1
DireccionAreaCeldas.EndColumn = 2
DireccionAreaCeldas.EndRow = 2

Hoja.insertCells(DireccionAreaCeldas, com.sun.star.sheet.CellInsertMode.DOWN)
```

En este ejemplo se inserta un área de celdas de dos filas por dos columnas en la segunda columna y fila (ambas identificadas por el número 1) de la primera hoja (número 0) de la hoja de cálculo. Los posibles valores del área de celdas especificada se desplazan hacia abajo.

Para definir el área de celdas que se desee insertar se utiliza la estructura `com.sun.star.table.CellRangeAddress`. En esta estructura se incluyen los siguientes valores:

- **`Sheet (short)`**: número de la hoja (la numeración se inicia en 0).
- **`StartColumn (long)`**: primera columna del área de celdas (la numeración se inicia en 0).
- **`StartRow (long)`**: primera fila del área de celdas (la numeración se inicia en 0).
- **`EndColumn (long)`**: última columna del área de celdas (la numeración se inicia en 0).
- **`EndRow (long)`**: última fila del área de celdas (la numeración se inicia en 0).

La estructura `CellRangeAddress` completa se debe pasar como primer parámetro al método `insertCells`. El segundo parámetro de `insertCells` contiene un valor de la enumeración `com.sun.star.sheet.CellInsertMode` y define lo que se debe hacer con los valores situados a partir de la posición de inserción. La enumeración `CellInsertMode` reconoce los valores siguientes:

- **NONE**: los valores actuales permanecen en la posición que ocupan.

- **DOWN**: las celdas situadas en la posición de inserción y debajo de ella se desplazan hacia abajo.
- **RIGHT**: las celdas situadas en la posición de inserción y a la derecha de ésta se desplazan hacia la derecha.
- **ROWS**: las celdas situadas después de la posición de inserción se desplazan hacia abajo.
- **COLUMNS**: las columnas situadas después de la posición de inserción se desplazan hacia la derecha.

El método `removeRange` es la contrapartida del método `insertCells`. Este método borra de la hoja el área de celdas definida en la estructura `CellRangeAddress`.

```
Dim Doc As Object
Dim Hoja As Object
Dim DireccionAreaCeldas As New com.sun.star.table.CellRangeAddress

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

DireccionAreaCeldas.Sheet = 0
DireccionAreaCeldas.StartColumn = 1
DireccionAreaCeldas.StartRow = 1
DireccionAreaCeldas.EndColumn = 2
DireccionAreaCeldas.EndRow = 2

Hoja.removeRange(DireccionAreaCeldas, com.sun.star.sheet.CellDeleteMode.UP)
```

En este ejemplo se suprime de la hoja el área de celdas B2:C3 y las celdas de debajo de dicha área se desplazan hacia arriba dos filas. El tipo de supresión se define mediante uno de los siguientes valores de la enumeración `com.sun.star.sheet.CellDeleteMode`:

- **NONE**: los valores actuales permanecen en la posición que ocupan.
- **UP:** las celdas situadas en la posición de inserción y debajo de ella se desplazan hacia arriba.
- **LEFT**: las celdas situadas en la posición de inserción y a la derecha de ésta se desplazan hacia la izquierda.
- **ROWS**: las celdas situadas después de la posición de inserción se desplazan hacia arriba.
- **COLUMNS**: las columnas situadas después de la posición de inserción se desplazan hacia la izquierda.

La interfaz `XRangeMovement` proporciona dos métodos adicionales para desplazar (`moveRange`) o copiar (`copyRange`) áreas de celdas. En el ejemplo siguiente se desplaza el área B2:C3 de modo que empiece en la posición A6:

```
Dim Doc As Object
Dim Hoja As Object
Dim DireccionAreaCeldas As New com.sun.star.table.CellRangeAddress
Dim DireccionCelda As New com.sun.star.table.CellAddress

Doc = StarDesktop.CurrentComponent
```

```
Hoja = Doc.Sheets(0)

DireccionAreaCeldas.Sheet = 0
DireccionAreaCeldas.StartColumn = 1
DireccionAreaCeldas.StartRow = 1
DireccionAreaCeldas.EndColumn = 2
DireccionAreaCeldas.EndRow = 2

DireccionCelda.Sheet = 0
DireccionCelda.Column = 0
DireccionCelda.Row = 5

Hoja.moveRange(DireccionCelda, DireccionAreaCeldas)
```

Aparte de la estructura `CellRangeAdress` , el método `moveRange` espera recibir una estructura `com.sun.star.table.CellAddress` para definir el origen del área de destino del desplazamiento. El método `CellAddress` proporciona los valores siguientes:

- **`Sheet (short)`**: número de la hoja de cálculo (la numeración se inicia en 0).
- **`Column (long)`**: número de la columna designada (la numeración se inicia en 0).
- **`Row (long)`**: número de la fila designada (la numeración se inicia en 0).

El método `moveRange` siempre sobrescribe el contenido de las celdas del área de destino. A diferencia del método `InsertCells`, el método `removeRange` no proporciona un parámetro para efectuar desplazamientos automáticos.

El método `copyRange` funciona de la misma forma que el método `moveRange`, salvo que `copyRange` inserta una copia del área de celdas en lugar de desplazarla.

Desde el punto de vista de su función, los métodos `insertCell`, `removeRange` y `copyRange` de StarOffice Basic son comparables a los métodos de VBA `Range.Insert`, `Range.Delete` y `Range.Copy`. Mientras que en VBA los métodos se aplican al objeto `Range` correspondiente, en StarOffice Basic se aplican al objeto `Sheet` asociado.

Formato

Los documentos hoja de cálculo proporcionan métodos y propiedades para dar formato a celdas y páginas.

Propiedades de la celda

Las opciones de formato de celdas son abundantes; por ejemplo, especificar el tipo de fuente del texto y su tamaño. Las celdas admiten los servicios `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties` cuyas .principales propiedades se describen en el capítulo 6 (***Documentos de texto:***). Los formatos de celda especiales los gestiona el servicio `com.sun.star.table.CellProperties`. Las principales propiedades de este servicio se describen en las próximas secciones.

Las propiedades especificadas se pueden aplicar a celdas individuales y a áreas de celdas.

El objeto `CellProperties` de la API de StarOffice es comparable al objeto `Interior` de VBA, que define también propiedades específicas de celdas.

Color de fondo y sombreado

El servicio `com.sun.star.table.CellProperties` proporciona las siguientes propiedades para definir los colores de fondo y los sombreados:

- **`CellBackColor (Long)`**: color de fondo de la celda.
- **`IsCellBackgroundTransparent (booleano)`**: establece el color de fondo en transparente.
- **`ShadowFormat (struct)`**: especifica el sombreado de las celdas (estructura de acuerdo con `com.sun.star.table.ShadowFormat`).

La estructura `com.sun.star.table.ShadowFormat` y las especificaciones detalladas de sombreado de celdas tienen la estructura siguiente:

- **`Location (enum)`**: posición de la sombra (valor procedente de la estructura `com.sun.star.table.ShadowLocation`).
- **`ShadowWidth (Short)`**: tamaño de la sombra en centésimas de milímetro.
- **`IsTransparent (booleano)`**: define la sombra como transparente.
- **`Color (Long)`**: color de la sombra.

En el ejemplo siguiente se escribe el número 1000 en la celda B2, se cambia el color de fondo a rojo mediante la propiedad `CellBackColor` y se crea una sombra de color gris claro que se desplaza 1 mm hacia la izquierda y hacia abajo.

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object
Dim FormatoSombra As New com.sun.star.table.ShadowFormat

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets(0)
Celda = Hoja.getCellByPosition(1,1)

Celda.Value = 1000

Celda.CellBackColor = RGB(255, 0, 0)

FormatoSombra.Location = com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
FormatoSombra.ShadowWidth = 100
FormatoSombra.Color = RGB(160, 160, 160)

Celda.ShadowFormat = FormatoSombra
```

Justificación

StarOffice ofrece diversas funciones para modificar la justificación de un texto en una celda.

Las propiedades siguientes definen la justificación horizontal y vertical de un texto:

- **HoriJustify (enum)**: justificación horizontal del texto (valor de `com.sun.star.table.CellHoriJustify`)
- **VertJustify (enum)**: justificación vertical del texto (valor de `com.sun.star.table.CellVertJustify`)
- **Orientation (enum)**: orientación del texto (valor de acuerdo con `com.sun.star.table.CellOrientation`)
- **IsTextWrapped (booleano)**: permite saltos de línea automáticos dentro de la celda
- **RotateAngle (Long)**: ángulo de rotación del texto en centésimas de grado

En el ejemplo siguiente se muestra la forma de "apilar" el contenido de una celda de forma que los caracteres individuales se impriman uno sobre el otro en la esquina superior izquierda de la celda. Los caracteres no se giran.

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets(0)
Celda = Hoja.getCellByPosition(1,1)

Celda.Value = 1000

Celda.HoriJustify = com.sun.star.table.CellHoriJustify.LEFT
Celda.VertJustify = com.sun.star.table.CellVertJustify.TOP
Celda.Orientation = com.sun.star.table.CellOrientation.STACKED
```

Formatos de número, fecha y texto

StarOffice ofrece una amplia variedad de formatos de fecha y hora predefinidos. Cada uno posee un número interno que se utiliza para asignar el formato a las celdas mediante la propiedad `NumberFormat`. StarOffice proporciona los métodos `queryKey` y `addNew` para poder acceder a los formatos numéricos actuales y crear formatos numéricos propios. Para acceder a los métodos se emplea la siguiente llamada:

```
FormatosNumericos = Doc.NumberFormats
```

Los formatos se especifican mediante una cadena de formato de estructura similar a la función de formato de StarOffice Basic. No obstante, hay una diferencia fundamental: mientras que el formato de la orden espera abreviaturas inglesas y puntos o caracteres decimales como separadores de miles, se deben utilizar las abreviaturas específicas del país en la estructura del formato de la orden para el objeto NumberFormats.

En el ejemplo siguiente se da formato a la celda B2 de modo que los números se muestren con tres decimales y se utilice la coma como separador de miles.

```
Dim Doc As Object
Dim Hoja As Object
Dim Celda As Object
```

```
Dim FormatosNumericos As Object
Dim CadenaFormatoNumerico as String
Dim IdFormatoNumerico As Long
Dim ConfiguracionLocal As New com.sun.star.lang.Locale

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets(0)
Celda = Hoja.getCellByPosition(1,1)

Celda.Value = 23400.3523565

ConfiguracionLocal.Language = "en"
ConfiguracionLocal.Country = "us"

FormatosNumericos = Doc.NumberFormats
CadenaFormatoNumerico = "#,##0.000"

IdFormatoNumerico = FormatosNumericos.queryKey(CadenaFormatoNumerico, ConfiguracionLocal,
True)
If IdFormatoNumerico = -1 Then
      IdFormatoNumerico = FormatosNumericos.addNew(CadenaFormatoNumerico,
ConfiguracionLocal)
End If

MsgBox IdFormatoNumerico
Celda.NumberFormat = IdFormatoNumerico
```

El diálogo **Formatear celdas** de StarOffice Calc proporciona una visión general de las diversas opciones del formato de celdas.

Propiedades de la página

Las propiedades de la página son las opciones de formato que sirven para situar el contenido del documento en una página, así como los elementos visuales que se repiten en todas las páginas. Incluyen:

- Formatos del papel
- Márgenes de la página
- Encabezamientos y pies de página.

El procedimiento de definición de los formatos de página es distinto al de las otras modalidades de formato. Mientras que los elementos celda, párrafo y carácter se pueden formatear directamente, los formatos de la página se pueden definir y aplicar de forma indirecta mediante estilos de página. Por ejemplo, los encabezamientos y los pies de página se agregan al estilo de página.

En las secciones siguientes se describen las principales opciones de formato para las páginas de la hoja de cálculo. Muchos de los estilos están también disponibles para los documentos de texto. Las propiedades de las páginas válidas para ambos tipos de documentos se definen en el servicio `com.sun.star.style.PageProperties`. Las propiedades que sólo se aplican a documentos de hoja de cálculo se definen en el servicio `com.sun.star.sheet.TablePageStyle`.

Las propiedades de la página (márgenes, bordes, etc.) de un documento de Microsoft Office se definen mediante un objeto `PageSetup` en el nivel de los objetos `Worksheet` (Excel) o `Document` (Word). En StarOffice, dichas propiedades se definen utilizando un estilo de página que a su vez está vinculado con el documento asociado.

Fondo de la página

El servicio `com.sun.star.style.PageProperties` define las siguientes propiedades del fondo de la página:

- **`BackColor (long)`:** color del fondo
- **`BackGraphicURL (String)`**: URL de la imagen del fondo que se desea utilizar
- **`BackGraphicFilter (String)`**: nombre del filtro para interpretar la imagen del fondo
- **`BackGraphicLocation (Enum)`**: posición de la imagen del fondo (de acuerdo con la enumeración `com.sun.star.style.GraphicLocation`)
- **`BackTransparent (booleano)`**: hace el fondo transparente

Formato de la página

El formato de la página se define utilizando las propiedades siguientes del servicio `com.sun.star.style.PageProperties`:

- **`IsLandscape (booleano)`**: formato horizontal
- **`Width (Long)`:** ancho de la página en centésimas de milímetro
- **`Height (Long)`:** altura de la página en centésimas de milímetro
- **`PrinterPaperTray (String)`**: nombre de la bandeja de papel de la impresora que desee utilizar

En el ejemplo siguiente se establece el tamaño del papel del estilo de página "Default" en el formato DIN A5 horizontal (altura 14,8 cm, ancho 21 cm):

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosPagina As Object
Dim PaginaPredeterminada As Object

Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDePagina = StyleFamilies.getByName("PageStyles")
PaginaPredeterminada = PageStyles.getByName("Default")

PaginaPredeterminada.IsLandscape = True
PaginaPredeterminada.Width = 21000
PaginaPredeterminada.Height = 14800
```

Márgenes , bordes y sombreados de la página

El servicio `com.sun.star.style.PageProperties` ofrece las propiedades siguientes para ajustar los márgenes, los bordes y los sombreados de la página:

- **`LeftMargin (long)`**: ancho del margen izquierdo de la página en centésimas de milímetro
- **`RightMargin (long)`**: ancho del margen derecho de la página en centésimas de milímetro
- **`TopMargin (long)`**: ancho del margen superior de la página en centésimas de milímetro
- **`BottomMargin (long)`**: ancho del margen inferior de la página en centésimas de milímetro
- **`LeftBorder (struct)`**: especificaciones de la línea izquierda del borde de la página (estructura `com.sun.star.table.BorderLine`)
- **`RightBorder (struct)`**: especificaciones de la línea derecha del borde de la página (estructura `com.sun.star.table.BorderLine`)
- **`TopBorder (struct)`**: especificaciones de la línea superior del borde de la página (estructura `com.sun.star.table.BorderLine`)
- **`BottomBorder (struct)`**: especificaciones de la línea superior del borde de la página (estructura `com.sun.star.table.BorderLine`)
- **`LeftBorderDistance (long)`**: distancia entre el borde izquierdo de la página y el contenido de la página en centésimas de milímetro
- **`RightBorderDistance (long)`**: distancia entre el borde derecho de la página y el contenido de la página en centésimas de milímetro
- **`TopBorderDistance (long)`**: distancia entre el borde superior de la página y el contenido de la página en centésimas de milímetro
- **`BottomBorderDistance (long)`**: distancia entre el borde inferior de la página y el contenido de la página en centésimas de milímetro
- **`ShadowFormat (struct)`**: especificaciones de sombreado del área de contenido de la página (estructura `com.sun.star.table.ShadowFormat`)

En el ejemplo siguiente se establecen en 1 centímetro los bordes izquierdo y derecho del estilo de página "Default".

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosPagina As Object
Dim PaginaPredeterminada As Object

Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDePagina = StyleFamilies.getByName("PageStyles")
PaginaPredeterminada = PageStyles.getByName("Default")

PaginaPredeterminada.LeftMargin = 1000
```

```
PaginaPredeterminada.RightMargin = 1000
```

Encabezamientos y pies de página

Los encabezamientos y pies de página de un documento forman parte de las propiedades de la página y se definen mediante el servicio `com.sun.star.style.PageProperties`. Las propiedades utilizadas para dar formato a los encabezamientos son:

- **`HeaderIsOn (booleano)`**: se activa el encabezamiento
- **`HeaderLeftMargin (long)`**: distancia entre el encabezamiento y el margen izquierdo de la página en centésimas de milímetro
- **`HeaderRightMargin (long)`**: distancia entre el encabezamiento y el margen derecho de la página en centésimas de milímetro
- **`HeaderBodyDistance (long)`**: distancia entre el encabezamiento y el cuerpo principal del documento en centésimas de milímetro
- **`HeaderHeight (Long)`**: altura del encabezamiento en centésimas de milímetro
- **`HeaderIsDynamicHeight (Boolean)`**: la altura del encabezado se adapta automáticamente al contenido
- **`HeaderLeftBorder (struct)`**: detalles del borde izquierdo del marco que rodea al encabezamiento (estructura `com.sun.star.table.BorderLine`)
- **`HeaderRightBorder (struct)`**: detalles del borde derecho del marco que rodea al encabezamiento (`estructura` com.sun.star.table.BorderLine)
- **`HeaderTopBorder (struct)`**: detalles del borde superior del marco que rodea al encabezamiento (`estructura` com.sun.star.table.BorderLine)
- **`HeaderBottomBorder (struct)`**: detalles del borde inferior del marco que rodea al encabezamiento (`estructura` com.sun.star.table.BorderLine)
- **`HeaderLeftBorderDistance (long)`**: distancia entre el borde izquierdo y el contenido del encabezamiento en centésimas de milímetro
- **`HeaderRightBorderDistance (long)`**: distancia entre el borde derecho y el contenido del encabezamiento en centésimas de milímetro
- **`HeaderTopBorderDistance (long)`**: distancia entre el borde superior y el contenido del encabezamiento en centésimas de milímetro
- **`HeaderBottomBorderDistance (long)`**: distancia entre el borde inferior y el contenido del encabezamiento en centésimas de milímetro
- **`HeaderIsShared (booleano)`**: el contenido de los encabezamientos en las páginas pares e impares es el mismo (consulte `HeaderText`, `HeaderTextLeft` y `HeaderTextRight`)
- **`HeaderBackColor (long)`:** color del fondo del encabezamiento
- **`HeaderBackGraphicURL (String)`**: URL de la imagen del fondo que se desee utilizar

- **HeaderBackGraphicFilter (String)**: nombre del filtro para interpretar la imagen del fondo del encabezamiento
- **HeaderBackGraphicLocation (Enum)** : ubicación de la imagen del fondo del encabezamiento (valor según la enumeración `com.sun.star.style.GraphicLocation`)
- **HeaderBackTransparent (booleano):** muestra transparente el fondo del encabezamiento
- **HeaderShadowFormat (struct)**: detalles de la sombra del encabezamiento (estructura `com.sun.star.table.ShadowFormat`)

Las propiedades utilizadas para dar formato a los pies de página son:

- **FooterIsOn (booleano)**: se activa el pie de página
- **FooterLeftMargin (long)**: distancia entre el pie de página y el margen izquierdo de la página en centésimas de milímetro
- **FooterRightMargin (long)**: distancia entre el pie de página y el margen derecho de la página en centésimas de milímetro
- **FooterBodyDistance (long)**: distancia entre el pie de página y el cuerpo principal del documento en centésimas de milímetro
- **FooterHeight (Long):** altura del pie de página en centésimas de milímetro
- **FooterIsDynamicHeight (Boolean)**: la altura del pie de página se adapta automáticamente al contenido
- **FooterLeftBorder (struct)**: detalles de la línea izquierda del marco que rodea al pie de página (estructura `com.sun.star.table.BorderLine`)
- **FooterRightBorder (struct)**: detalles de la línea derecha del marco que rodea al pie de página (`estructura` com.sun.star.table.BorderLine)
- **FooterTopBorder (struct)**: detalles de la línea superior del marco que rodea al pie de página (`estructura` com.sun.star.table.BorderLine)
- **FooterBottomBorder (struct)**: detalles de la línea inferior del marco que rodea al pie de página (`estructura` com.sun.star.table.BorderLine)
- **FooterLeftBorderDistance (long)**: distancia entre el borde izquierdo del pie de página y el contenido de éste en centésimas de milímetro
- **FooterRightBorderDistance (long)**: distancia entre el borde derecho del pie de página y el contenido de éste en centésimas de milímetro
- **FooterTopBorderDistance (long)**: distancia entre el borde superior del pie de página y el contenido de éste en centésimas de milímetro
- **FooterBottomBorderDistance (long)**: distancia entre el borde inferior del pie de página y el contenido de éste en centésimas de milímetro
- **FooterIsShared (booleano)**: el contenido de los pies de página en las páginas pares e impares es el mismo (consulte `FooterText`, `FooterTextLeft` y `FooterTextRight`)

- **FooterBackColor (long)**: color del fondo del pie de página
- **FooterGraphicURL (String)**: URL de la imagen del fondo que se desee utilizar
- **FooterBackGraphicFilter (String)**: nombre del filtro para interpretar la imagen del fondo del pie de página
- **FooterBackGraphicLocation (Enum)** - ubicación de la imagen del fondo del pie de página (valor según la enumeración `com.sun.star.style.GraphicLocation`)
- **FooterBackTransparent (booleano)**: muestra transparente el fondo del pie de página
- **FooterShadowFormat (struct)**: detalles de la sombra del pie de página (estructura `com.sun.star.table.ShadowFormat`)

Cambio del texto de los encabezamientos y pies de página

Se puede acceder al contenido de los encabezamientos y pies de página de una hoja de cálculo mediante las propiedades siguientes:

- **LeftPageHeaderContent (objeto)**: contenido de los encabezamientos de las páginas pares (servicio `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageHeaderContent (objeto)**: contenido de los encabezamientos de las páginas impares (servicio `com.sun.star.sheet.HeaderFooterContent`)
- **LeftPageFooterContent (objeto)**: contenido de los pies de página de las páginas pares (servicio `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageFooterContent (objeto)**: contenido de los pies de página de las páginas impares (servicio `com.sun.star.sheet.HeaderFooterContent`)

Si no necesita distinguir entre los encabezamientos o los pies de página de las páginas pares e impares (la propiedad `FooterIsShared` es `False`), defina las propiedades en las páginas impares.

Todos los objetos mencionados devuelven un objeto que admite el servicio `com.sun.star.sheet.HeaderFooterContent`. Mediante las propiedades (no genuinas) `LeftText`, `CenterText` y `RightText`, este servicio proporciona tres elementos de texto para los encabezamientos y pies de página de StarOffice Calc.

En el ejemplo siguiente se escribe el valor "Es una prueba." en el campo de texto izquierdo del encabezamiento de la plantilla "Default".

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosPagina As Object
Dim PaginaPredeterminada As Object
Dim TextoEnc As Object
Dim ContenidoEnc As Object
Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDePagina = StyleFamilies.getByName("PageStyles")
```

```
PaginaPredeterminada = PageStyles.getByName("Default")

PaginaPredeterminada.HeaderIsOn = True
ContenidoEnc = PaginaPredeterminada.RightPageHeaderContent
TextoEnc = ContenidoEnc.LeftText
TextoEnc.String = "Es una prueba."
PaginaPredeterminada.RightPageHeaderContent = ContenidoEnc
```

Observe la última línea del ejemplo: Una vez modificado el texto, el objeto `TextContent` se debe asignar de nuevo al encabezamiento para que el cambio surta efecto.

Los documentos de texto (StarOffice Writer) disponen de otra posibilidad para modificar el texto de los encabezamientos, ya que constan de un único bloque de texto. Las siguientes propiedades están definidas en el servicio `com.sun.star.style.PageProperties`:

- **`HeaderText (objeto)`**: objeto de texto con el contenido del encabezamiento (servicio `com.sun.star.text.XText`)
- **`HeaderTextLeft (objeto)`**: objeto de texto con el contenido de los encabezamientos de las páginas del lado izquierdo (servicio `com.sun.star.text.XText`)
- **`HeaderTextRight (objeto)`**: objeto de texto con el contenido de los encabezamientos de las páginas del lado derecho (servicio `com.sun.star.text.XText`)
- **`FooterLeft (objeto)`**: objeto de texto con el contenido del pie de página (servicio `com.sun.star.text.XText`)
- **`FooterTextLeft (objeto)`**: objeto de texto con el contenido de los pies de página de las páginas del lado izquierdo (servicio `com.sun.star.text.XText`)
- **`FooterTextRight (objeto)`**: objeto de texto con el contenido de los pies de página de las páginas del lado derecho (servicio `com.sun.star.text.XText`)

En el ejemplo siguiente se crea un encabezamiento para documentos de texto con el estilo de página "Default" al que se añade el texto "Es una prueba".

```
Dim Doc As Object
Dim Hoja As Object
Dim FamiliasDeEstilos As Object
Dim EstilosPagina As Object
Dim PaginaPredeterminada As Object
Dim TextoEnc As Object

Doc = StarDesktop.CurrentComponent
FamiliasDeEstilos = Doc.StyleFamilies
EstilosDePagina = StyleFamilies.getByName("PageStyles")
PaginaPredeterminada = PageStyles.getByName("Default")

PaginaPredeterminada.HeaderIsOn = True
TextoEnc = PaginaPredeterminada.HeaderText

TextoEnc.String = "Es una prueba."
```

En este ejemplo, el acceso se efectúa a través de la propiedad `HeaderText` del estilo de página, en lugar de usar el objeto `HeaderFooterContent`.

Centrado (sólo en hojas de cálculo)

El servicio `com.sun.star.sheet.TablePageStyle` sólo se utiliza en los estillos de página de StarOffice Calc y permite centrar en la página las áreas de celda que se desea imprimir. Este servicio ofrece las propiedades siguientes:

- **`CenterHorizontally (booleano)`**: el contenido de la tabla se centra horizontalmente
- **`CenterVertically (booleano)`**: el contenido de la tabla se centra verticalmente

Definición de los elementos que se deben imprimir (sólo hojas de cálculo)

Al formatear las hojas de cálculo se puede definir si los elementos de la página serán o no visibles. Para ello, el servicio `com.sun.star.sheet.TablePageStyle` ofrece las propiedades siguientes:

- **`PrintAnnotations (booleano)`**: imprime los comentarios de las celdas
- **`PrintGrid (booleano)`**: imprime las líneas de cuadrícula de las celdas
- **`PrintHeaders (booleano)`**: imprime los encabezamientos de la fila y la columna
- **`PrintCharts (booleano)`**: imprime los diagramas contenidos en una hoja
- **`PrintObjects (booleano)`**: imprime los objetos incrustados
- **`PrintDrawing (booleano)`**: imprime los objetos de dibujo
- **`PrintDownFirst (booleano)`**: si el contenido de una hoja abarca varias páginas, éstas se imprimen en primer lugar en orden vertical descendente y a continuación se avanza hacia la derecha.
- **`PrintFormulas (booleano)`**: se imprimen las fórmulas en lugar de los valores calculados
- **`PrintZeroValues (booleano)`**: se imprimen los valores cero

Edición eficiente de los documentos de hoja de cálculo

Mientras que en la sección anterior se ha descrito la estructura principal de los documentos de hoja de cálculo, en esta sección se describen los servicios que permiten acceder con facilidad a celdas individuales o áreas de celdas.

Áreas de celdas

Aparte de un objeto para las celdas individuales (servicio `com.sun.star.table.Cell`), StarOffice ofrece también objetos para representar áreas de celdas. Estos objetos `CellRange` se crean mediante la llamada `getCellRangeByName` del objeto hoja de cálculo:

```
Dim Doc As Object
Dim Hoja As Object
Dim AreaDeCeldas As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets.getByName("Hoja 1")
AreaDeCeldas = Hoja.getCellRangeByName("A1:C15")
```

El signo de dos puntos (:) se utiliza para especificar un área de celdas en un documento de hoja de cálculo. Por ejemplo, A1:C15 representa todas las celdas de las filas 1 a 15 en las columnas A, B y C.

La ubicación de las celdas individuales en un área de celdas se puede determinar mediante el método `getCellByPosition`, siendo (0, 0) las coordenadas de la celda superior izquierda del área de celdas. En el ejemplo siguiente se utiliza este método para crear un objeto de la celda C3.

```
Dim Doc As Object
Dim Hoja As Object
Dim AreaDeCeldas As Object
Dim Celda As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets.getByName("Hoja 1")
AreaDeCeldas = Hoja.getCellRangeByName("B2:D4")
Celda = AreaDeCeldas.getCellByPosition(1, 1)
```

Formato de áreas de celdas

Se puede aplicar formato a las áreas de celdas como si fuesen celdas individuales, mediante el servicio `com.sun.star.table.CellProperties`. Para obtener más información y ejemplos de este servicio, consulte la sección ***Formato***.

Cálculos con áreas de celdas

Se puede utilizar el método `computeFunction` para efectuar cálculos matemáticos en áreas de celdas. El método `computeFunction` recibe una constante como parámetro para describir la función matemática que se quiere usar. Las constantes asociadas se encuentran definidas en la enumeración `com.sun.star.sheet.GeneralFunction`.

Están disponibles los valores siguientes:

- **SUM**: suma de todos los valores numéricos
- **COUNT**: número total de valores (incluidos los no numéricos)
- **COUNTNUMS**: número total de valores numéricos
- **AVERAGE:** promedio de todos los valores numéricos
- **MAX**: valor numérico más alto
- **MIN**: valor numérico más bajo
- **PRODUCT:** producto de todos los valores numéricos
- **STDEV**: desviación estándar
- **VAR**: varianza
- **STDEVP**: desviación estándar basada en la población total
- **VARP**: varianza basada en la población total

En el ejemplo siguiente se calcula el promedio del área de celdas A1:C3 y se muestra el resultado en un cuadro de mensaje:

```
Dim Doc As Object
Dim Hoja As Object
Dim AreaDeCeldas As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets.getByName("Hoja 1")
AreaDeCeldas = Hoja.getCellRangeByName("A1:C3")

MsgBox AreaDeCeldas.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Borrado del contenido de las celdas

El método `clearContents` simplifica el proceso de borrar el contenido de las celdas y áreas de celdas, ya que borra un tipo de contenido específico de un área de celdas.

En el ejemplo siguiente se borran todas las cadenas de caracteres y la información de formato directo del área de celdas B2:C3.

```
Dim Doc As Object
Dim Hoja As Object
Dim AreaDeCeldas As Object
Dim Indicadores As Long

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets(0)
AreaDeCeldas = Hoja.getCellRangeByName("B2:C3")

Indicadores = com.sun.star.sheet.CellFlags.STRING + _
            com.sun.star.sheet.CellFlags.HARDATTR

CellRange.clearContents(Indicadores)
```

Los indicadores especificados en `clearContents` proceden de la lista de constantes `com.sun.star.sheet.CellFlags`. Ésta contiene los siguientes elementos:

- **`VALUE`**: valores numéricos que no tienen formato de fecha o de hora
- **`DATETIME`**: valores numéricos que tienen formato de fecha o de hora
- **`STRING`**: cadenas de caracteres
- **`ANNOTATION`**: comentarios vinculados a celdas
- **`FORMULA`**: fórmulas
- **`HARDATTR`**: formato directo de celdas
- **`STYLES`**: formato indirecto
- **`OBJECTS`**: objetos de dibujo vinculados a celdas
- **`EDITATTR`**: formatos de caracteres que se aplican únicamente a algunas partes de las celdas

También se pueden agregar constantes para borrar distintos tipos de información con una única llamada a `clearContents`.

Búsqueda y sustitución de los contenidos de las celdas

Los documentos de hoja de cálculo, como los de texto, ofrecen una función para buscar y reemplazar.

Los objetos de descriptor para buscar y reemplazar en documentos de hoja de cálculo no se crean directamente a través del objeto documento, sino a través de la lista `Sheets`. A continuación se muestra un ejemplo de un proceso de buscar y reemplazar:

```
Dim Doc As Object
Dim Hoja As Object
Dim DescriptorSustitucion As Object
Dim I As Integer

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

DescriptorSustitucion = Hoja.createReplaceDescriptor()
DescriptorSustitucion.SearchString = "es"
DescriptorSustitucion.ReplaceString = "era"

For I = 0 to Doc.Sheets.Count - 1
      Hoja = Doc.Sheets(I)
      Hoja.ReplaceAll(DescriptorSustitucion)
Next I
```

En este ejemplo se utiliza la primera página del documento para crear un `DescriptorSustitucion` y luego se aplica a todas las páginas mediante un bucle.

CAPÍTULO **8**

Dibujos y presentaciones

Este capítulo proporciona una introducción a la creación y edición de dibujos controladas por macros. En la primera sección se describe la estructura de los dibujos, incluidos los elementos básicos que los componen. En la segunda sección se analizan funciones de edición más complejas como agrupar, girar y escalar objetos.

En el capítulo 5, ***Trabajo con documentos de StarOffice,*** encontrará información acerca de cómo crear, abrir y guardar dibujos.

La estructura de los dibujos

StarOffice no pone límites al número de páginas de un documento de dibujo. El diseño de cada página puede ser independiente. Tampoco hay límites en el número de elementos de dibujo que pueda contener una página.

Sin embargo, la presencia de las ***capas*** complica un poco el esquema general. De forma predeterminada, cada documento de dibujo consta de las capas ***Diseño***, ***Controles*** y ***Líneas de dimensiones*** y todos los elementos de dibujo se agregan a la capa ***Diseño***. Hay la opción de agregar nuevas capas. Para obtener más información, consulte el manual StarOffice Developer's Guide.

Páginas

Las páginas de un documento de dibujo están disponibles mediante la lista `DrawPages`. Se puede acceder a las páginas individuales mediante su número o su nombre. Si un documento tiene una sola página denominada ***Diapositiva 1***, los siguientes ejemplos son idénticos.

Ejemplo 1:

```
Dim Doc As Object
Dim Pagina As Object

Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages(0)
```

Ejemplo 2:

```
Dim Doc As Object
Dim Pagina As Object

Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages.getByName("Diapositiva 1")
```

En el primer ejemplo se accede a la página a través de su número (la numeración empieza en 0). En el segundo ejemplo se accede por su nombre mediante el método getByName.

```
Dim sUrl As String, sFiltro As String
Dim sOpciones As String
Dim oHojas As Object, oHoja As Object

      oHojas = oDocumento.Sheets

      If oHojas.hasByName("Vinculo") Then
            oHoja = oHojas.getByName("Vinculo")
      Else
            oHoja = oDocumento.createInstance("com.sun.star.sheet.Spreadsheet")
            oHojas.insertByName("Vinculo", oHoja)
            oHoja.IsVisible = False
      End If
```

La llamada anterior devuelve un objeto de página que admite el servicio com.sun.star.drawing.DrawPage. Dicho servicio reconoce las propiedades siguientes:

- **BorderLeft (Long)**: borde izquierdo en centésimas de milímetro
- **BorderRight (Long)**: borde derecho en centésimas de milímetro
- **BorderTop (Long)**: borde superior en centésimas de milímetro
- **BorderBottom (Long)**: borde inferior en centésimas de milímetro
- **Width (Long)**: ancho de página en centésimas de milímetro
- **Height (Long)**: altura de página en centésimas de milímetro
- **Number (Short)**: número de páginas (la numeración empieza en 1), sólo lectura
- **Orientation (enum)**: orientación de la página (valor de acuerdo con com.sun.star.view.PaperOrientation)

La modificación de estos valores de configuración afecta a ***todas*** las páginas del documento.

En el ejemplo siguiente se define el tamaño de página del documento de dibujo que se acaba de abrir en 20 × 20 centímetros, con un margen de 0,5 centímetros:

```
Dim Doc As Object
Dim Pagina As Object

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

Pagina.BorderLeft = 500
Pagina.BorderRight = 500
Pagina.BorderTop = 500
Pagina.BorderBottom = 500

Pagina.Width = 20000
Pagina.Height = 20000
```

Propiedades básicas de los objetos de dibujo

Los objetos de dibujo incluyen formas (rectángulos, círculos, etc.), líneas y objetos de texto. Todos ellos comparten diversas características y admiten el servicio `com.sun.star.drawing.Shape`. Este servicio define las propiedades `Size` y `Position` de un objeto de dibujo.

StarOffice Basic ofrece también otros diversos servicios mediante los cuales se pueden modificar dichas propiedades, como formatos y aplicación de rellenos. Las opciones de formato disponibles dependen del tipo de objeto de dibujo.

En el ejemplo siguiente se crea e inserta un rectángulo en un documento de dibujo:

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

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

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

Pagina.add(FormaRectangular)
```

En este ejemplo se utiliza la llamada a `StarDesktop.CurrentComponent` para determinar qué documento está abierto. El objeto documento determinado devuelve la primera página del dibujo a través de la llamada `drawPages(0)`.

A continuación se inicializan las estructuras `Punto` y `Tamano` con el punto de origen (esquina izquierda) y el tamaño del objeto de dibujo. Las longitudes se especifican en centésimas de milímetro.

El código del programa utiliza la llamada `Doc.createInstance` para crear el objeto de dibujo rectángulo según se especifica en el servicio `com.sun.star.drawing.RectangleShape`. Finalmente, el objeto de dibujo se asigna a una página mediante la llamada `Pagina.add`.

Propiedades de relleno

En esta sección se describen cuatro servicios y, en cada caso, el programa de ejemplo utiliza un elemento rectangular que combina varios tipos de formato. Las propiedades de relleno se combinan en el servicio `com.sun.star.drawing.FillProperties`.

StarOffice reconoce cuatro tipos principales de formato para un área de relleno. El más simple es el relleno de un sólo color. Las opciones para definir gradientes de color y tramas permiten hacer intervenir otros colores. La cuarta variante es la opción de proyectar imágenes en el área de relleno.

El modo de relleno de un objeto de dibujo se define mediante la propiedad `FillStyle`. Los valores permitidos se definen en `com.sun.star.drawing.FillStyle`.

Rellenos de un solo color

La propiedad principal de los rellenos de un solo color es:

- **`FillColor (Long)`:** color de relleno del área.

Para utilizar el modo de relleno deberá asignar a la propiedad `FillStyle` el modo de relleno `SOLID`.

En el ejemplo siguiente se crea una forma rectangular y se rellena de color rojo (valor RGB 255, 0, 0):

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.FillStyle = com.sun.star.drawing.FillStyle.SOLID
FormaRectangular.FillColor = RGB(255,0,0)

Pagina.add(FormaRectangular)
```

Gradiente de color

Si establece el valor de la propiedad `FillStyle` en `GRADIENT` podrá aplicar un gradiente de color a cualquier área de un documento de StarOffice que pueda rellenarse.

Si desea aplicar un gradiente de color predefinido, puede asignar el nombre asociado de la propiedad `FillTransparenceGradientName`. Para definir un gradiente de color propio, deberá llenar una estructura `com.sun.star.awt.Gradient` para asignarla a la propiedad `FillGradient`. Esta propiedad ofrece las opciones siguientes:

- **`Style (Enum)`**: tipo de gradiente; por ejemplo, lineal o radial (valores predeterminados de acuerdo con `com.sun.star.awt.GradientStyle`)
- **StartColor (Long)**: color inicial del gradiente de color
- **EndColor (Long)**: color final del gradiente de color

- **Angle (Short)**: ángulo del gradiente de color en décimas de grado
- **XOffset (Short)**: coordenada x en la que empieza el gradiente de color, especificada en centésimas de milímetro
- **YOffset (Short)**: coordenada y en la que empieza el gradiente de color, especificada en centésimas de milímetro
- **StartIntensity (Short)**: intensidad del `StartColor` expresada en forma de porcentaje (en StarOffice Basic se pueden especificar valores superiores al 100%)
- **EndIntensity (Short)**: intensidad del `EndColor` expresada en forma de porcentaje (en StarOffice Basic se pueden especificar valores superiores al 100%)
- **StepCount (Short)**: número de graduaciones de color que StarOffice debe calcular para los gradientes

En el ejemplo siguiente se muestra el uso de los gradientes de color con la ayuda de la estructura `com.sun.star.awt.Gradient`:

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size
Dim Gradiente As New com.sun.star.awt.Gradient

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

Gradiente.Style = com.sun.star.awt.GradientStyle.LINEAR
Gradiente.StartColor = RGB(255,0,0)
Gradiente.EndColor = RGB(0,255,0)
Gradiente.StartIntensity = 150
Gradiente.EndIntensity = 150
Gradiente.Angle = 450
Gradiente.StepCount = 100

FormaRectangular.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
FormaRectangular.FillGradient = Gradiente

Pagina.add(FormaRectangular)
```

En este ejemplo se crea un gradiente de color lineal (`Style` = `LINEAR`). El gradiente empieza con el color rojo (`StartColor`) en la esquina superior izquierda y se extiende, con un ángulo de 45

grados (Angle), hasta el verde (EndColor) en la esquina inferior derecha. La intensidad de los colores inicial y final es del 150 por ciento (StartIntensity y EndIntensity), lo que se traduce en que los colores parecen más brillantes que los valores especificados en las propiedades StartColor y EndColor. El gradiente de color se representa mediante un centenar de colores individuales graduados (StepCount).

Tramas

Para crear un relleno de trama se debe asignar a la propiedad FillStyle el valor HATCH. El código de programa para definir la trama es muy similar al utilizado para gradientes de color. Nuevamente se utiliza una estructura auxiliar, com.sun.star.drawing.Hatch en este caso, para definir el aspecto de la trama. Las propiedades de la estructura utilizada para las tramas son:

- **Style (Enum)**: tipo de trama: simple, cuadrada o cuadrada con diagonales (valores predeterminados de acuerdo con com.sun.star.awt.HatchStyle)
- **Color (Long)**: color de las líneas
- **Distance (Long)**: distancia entre las líneas en centésimas de milímetro
- **Angle (Short)**: ángulo de la trama en décimas de grado

En el ejemplo siguiente se muestra el uso de una estructura de trama:

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size
Dim Trama As New com.sun.star.drawing.Hatch

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.FillStyle = com.sun.star.drawing.FillStyle.HATCH

Trama.Style = com.sun.star.drawing.HatchStyle.SINGLE
Trama.Color = RGB(64,64,64)
Trama.Distance = 20
Trama.Angle = 450

FormaRectangular.FillHatch = Trama

Pagina.add(FormaRectangular)
```

Este fragmento de código crea una estructura de trama simple (`HatchStyle` = `SINGLE`) cuyas líneas tiene un ángulo de 45 grados (`Angle`). El color de las líneas es gris oscuro (`Color`) y la separación entre ellas (`Distance`) es de 0,2 milímetros.

Bitmaps

Para utilizar como relleno la proyección de un bitmap, la propiedad `FillStyle` debe tener como valor `BITMAP`. Si el bitmap está disponible en StarOffice, únicamente deberá especificar su nombre en la propiedad `FillBitMapName` y su estilo de visualización (simple, mosaico o estirado) en la propiedad `FillBitmapMode` (valores predeterminados de acuerdo con `com.sun.star.drawing.BitmapMode`).

Si desea utilizar un archivo de bitmap externo, puede especificar su URL en la propiedad `FillBitmapURL`.

En el ejemplo siguiente se crea un rectángulo cuya área se rellena con un mosaico del bitmap Sky, disponible en StarOffice.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.FillStyle = com.sun.star.drawing.FillStyle.BITMAP

FormaRectangular.FillBitmapName = "Sky"
FormaRectangular.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT


Pagina.add(FormaRectangular)
```

Transparencia

Es posible ajustar la transparencia de cualquiera de los estilos de relleno. La forma más fácil de cambiar la transparencia de un elemento de dibujo es mediante la propiedad `FillTransparence`.

En el ejemplo siguiente se crea un rectángulo rojo con una transparencia del 50 por ciento.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.FillStyle = com.sun.star.drawing.FillStyle.SOLID
FormaRectangular.FillTransparence = 50
FormaRectangular.FillColor = RGB(255,0,0)

Pagina.add(FormaRectangular)
```

Para hacer que el relleno sea transparente, defina el valor de la propiedad `FillTransparence` en 100.

Aparte de la propiedad `FillTransparence`, el servicio `com.sun.star.drawing.FillProperties` ofrece también la propiedad `FillTransparenceGradient` que se utiliza para definir un gradiente que especifica la transparencia de un área de relleno.

Propiedades de líneas

Todos los objetos de dibujo que pueden llevar un borde admiten el servicio `com.sun.star.drawing.LineStyle`. Algunas de las propiedades que proporciona este servicio son:

- **`LineStyle (Enum)`**: tipo de línea (valores predeterminados de acuerdo con `com.sun.star.drawing.LineStyle`)
- **`LineColor (Long)`**: color de la línea
- **`LineTransparence (Short)`**: transparencia de la línea
- **`LineWidth (Long)`**: grosor de la línea en centésimas de milímetro
- **`LineJoint (Enum)`**: transiciones a puntos de conexión (valores predeterminados de acuerdo con `com.sun.star.drawing.LineJoint`)

En el ejemplo siguiente se crea un rectángulo con un borde sólido (`LineStyle` = `SOLID`) de 5 milímetros de grosor (`LineWidth`) y una transparencia del 50 por ciento. Los bordes izquierdo y derecho de la línea se extienden hasta sus puntos de intersección (`LineJoint` = `MITER`) para formar un ángulo recto.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.LineColor = RGB(128,128,128)
FormaRectangular.LineTransparence = 50
FormaRectangular.LineWidth = 500
FormaRectangular.LineJoint = com.sun.star.drawing.LineJoint.MITER

FormaRectangular.LineStyle = com.sun.star.drawing.LineStyle.SOLID

Pagina.add(FormaRectangular)
```

Además de las propiedades enumeradas, el servicio `com.sun.star.drawing.LineStyle` ofrece opciones para dibujar líneas de puntos y de guiones. Si desea más información, consulte la referencia de la API de StarOffice.

Propiedades de texto (objetos de dibujo)

Los servicios `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties` se pueden utilizar para dar formato a texto en objetos de dibujo. Estos servicios están relacionados con caracteres individuales y párrafos y se describen detalladamente en el capítulo 6 (***Documentos de texto:***).

En el ejemplo siguiente se inserta texto en un rectángulo y se da formato a la fuente mediante el servicio `com.sun.star.style.CharacterProperties`.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size
```

```
Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000
Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages(0)

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

Pagina.add(FormaRectangular)

FormaRectangular.String = "Esto es una prueba"
FormaRectangular.CharWeight = com.sun.star.awt.FontWeight.BOLD
FormaRectangular.CharFontName = "Arial"
```

En este fragmento de código se usa la propiedad `String` del rectángulo para insertar el texto y las propiedades `CharWeight` y `CharFontName` del servicio `com.sun.star.style.CharacterProperties` para dar formato a la fuente del texto.

El texto sólo puede insertarse una vez que se ha agregado el objeto de dibujo a la página de dibujo. También se puede utilizar el servicio `com.sun.star.drawing.Text` para situar y formatear texto en un objeto de dibujo. Algunas de las propiedades fundamentales de este servicio son:

- **`TextAutoGrowHeight (booleano)`**: adapta la altura del elemento de dibujo al texto que contiene
- **`TextAutoGrowWidth (booleano)`**: adapta el ancho del elemento de dibujo al texto que contiene
- **`TextHorizontalAdjust (Enum)`**: posición horizontal del texto dentro del elemento de dibujo (valores predeterminados de acuerdo con `com.sun.star.drawing.TextHorizontalAdjust`)
- **`TextVerticalAdjust (Enum)`**: posición vertical del texto dentro del elemento de dibujo (valores predeterminados de acuerdo con `com.sun.star.drawing.TextVerticalAdjust`)
- **`TextLeftDistance (Long)`**: distancia por la izquierda entre el elemento de dibujo y el texto en centésimas de milímetro
- **`TextRightDistance (Long)`**: distancia por la derecha entre el elemento de dibujo y el texto en centésimas de milímetro
- **`TextUpperDistance (Long)`**: distancia por arriba entre el elemento de dibujo y el texto en centésimas de milímetro
- **`TextLowerDistance (Long)`**: distancia por abajo entre el elemento de dibujo y el texto en centésimas de milímetro

En el ejemplo siguiente se muestra el uso de las propiedades mencionadas.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size
```

```
Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

Pagina.add(FormaRectangular)

FormaRectangular.String = "Esto es una prueba"  ' Sólo puede aparecer después de
Page.add!

FormaRectangular.TextVerticalAdjust = com.sun.star.drawing.TextVerticalAdjust.TOP
FormaRectangular.TextHorizontalAdjust = com.sun.star.drawing.TextHorizontalAdjust.LEFT

FormaRectangular.TextLeftDistance = 300
FormaRectangular.TextRightDistance = 300
FormaRectangular.TextUpperDistance = 300
FormaRectangular.TextLowerDistance = 300
```

En este fragmento de código se inserta un elemento de dibujo en una página y, a continuación, se agrega texto en la esquina superior izquierda del objeto de dibujo utilizando las propiedades `TextVerticalAdjust` y `TextHorizontalAdjust`. La distancia mínima entre el borde del texto y el objeto de dibujo se establece en tres milímetros.

Propiedades de la sombra

El servicio `com.sun.star.drawing.ShadowProperties` permite agregar una sombra a casi todos los objetos de dibujo. Las propiedades de este servicio son:

- **`Shadow (booleano)`**: activa la sombra
- **`ShadowColor (Long)`**: color de la sombra
- **`ShadowTransparence (Short)`**: transparencia de la sombra
- **`ShadowXDistance (Long)`**: distancia vertical desde la sombra al objeto de dibujo en centésimas de milímetro
- **`ShadowYDistance (Long)`**: distancia horizontal desde la sombra al objeto de dibujo en centésimas de milímetro

En el ejemplo siguiente se crea un rectángulo con una sombra desplazada vertical y horizontalmente del rectángulo 2 milímetros. La sombra se colorea en gris oscuro con una transparencia del 50%.

```
Dim Doc As Object
```

```
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.Shadow = True
FormaRectangular.ShadowColor = RGB(192,192,192)
FormaRectangular.ShadowTransparence = 50
FormaRectangular.ShadowXDistance = 200
FormaRectangular.ShadowYDistance = 200

Pagina.add(FormaRectangular)
```

Resumen de diversos objetos de dibujo

Formas rectangulares

Los objetos rectangulares (`com.sun.star.drawing.RectangleShape`) admiten los siguientes servicios para el formato de objetos:

- **Propiedades de relleno**: `com.sun.star.drawing.FillProperties`
- **Propiedades de línea**: `com.sun.star.drawing.LineProperties`
- **Propiedades de texto**: `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties`)
- **Propiedades de sombra**: `com.sun.star.drawing.ShadowProperties`
- **CornerRadius (Long)**: radio para el redondeo de esquinas en centésimas de milímetro

Círculos y elipses

El servicio `com.sun.star.drawing.EllipseShape` se encarga de los círculos y las elipses; admite los servicios siguientes:

- **Propiedades de relleno**: `com.sun.star.drawing.FillProperties`
- **Propiedades de línea**: `com.sun.star.drawing.LineProperties`

- **Propiedades de texto**: `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties`)
- **Propiedades de sombra**: `com.sun.star.drawing.ShadowProperties`

Además de estos servicios, los círculos y las elipses disponen también de estas propiedades:

- **`CircleKind (Enum)`**: tipo de círculo o elipse (valores predeterminados de acuerdo con `com.sun.star.drawing.CircleKind`)
- **`CircleStartAngle (Long)`**: ángulo inicial en décimas de grado (sólo para segmentos circulares o elípticos)
- **`CircleEndAngle (Long)`**: ángulo final en décimas de grado (sólo para segmentos circulares o elípticos))

La propiedad `CircleKind` determina si un objeto es un círculo completo, un sector o un segmento circulares. Están disponibles los valores siguientes:

- **`com.sun.star.drawing.CircleKind.FULL`**: círculo o elipse completos
- **`com.sun.star.drawing.CircleKind.CUT`**: segmento circular (círculo parcial cuyas interfaces están enlazadas directamente entre sí)
- **`com.sun.star.drawing.CircleKind.SECTION`**: sector circular
- **`com.sun.star.drawing.CircleKind.ARC`**: ángulo (no incluye la línea del círculo)

En el ejemplo siguiente se crea un sector circular con un ángulo de 70 grados (generado por la diferencia entre los ángulos inicial de 20 grados y final de 90 grados)

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaEliptica As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaEliptica = Doc.createInstance("com.sun.star.drawing.EllipseShape")
FormaEliptica.Size = Tamano
FormaEliptica.Position = Punto

FormaEliptica.CircleStartAngle = 2000
FormaEliptica.CircleEndAngle = 9000
FormaEliptica.CircleKind =  com.sun.star.drawing.CircleKind.SECTION

Pagina.add(FormaEliptica)
```

Líneas

Para los objetos de línea, StarOffice ofrece el servicio `com.sun.star.drawing.LineShape`. Los objetos de línea admiten todos los servicios de formato generales excepto aquellos que afectan a áreas. A continuación se enumeran todas las propiedades asociadas con el servicio `LineShape`:

- **Propiedades de línea**: `com.sun.star.drawing.LineProperties`
- **Propiedades de texto**: `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties`)
- **Propiedades de sombra**: `com.sun.star.drawing.ShadowProperties`

En el ejemplo siguiente se crea y se da formato a una línea con la ayuda de las propiedades mencionadas. El origen de la línea se especifica en la propiedad `Location`, mientras que las coordenadas correspondientes a la propiedad `Size` especifican el punto final de la línea.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaLineal As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaLineal = Doc.createInstance("com.sun.star.drawing.LineShape")
FormaLineal.Size = Tamano
FormaLineal.Position = Punto

Pagina.add(FormaLineal)
```

Formas polipoligonales

StarOffice admite también formas poligonales complejas mediante el servicio `com.sun.star.drawing.PolyPolygonShape`. Hablando con precisión, un ***PolyPolygon*** ("polipolígono") no es un polígono simple sino un polígono múltiple. Por consiguiente, se pueden especificar varias listas independientes con puntos de intersección y combinarlas para formar un objeto completo.

Como sucede con las formas rectangulares, los polipolígonos admiten todas las propiedades de formato de los objetos de dibujo:

- **Propiedades de relleno**: `com.sun.star.drawing.FillProperties`
- **Propiedades de línea**: `com.sun.star.drawing.LineProperties`

- **Propiedades de texto**: `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties`)
- **Propiedades de sombra**: `com.sun.star.drawing.ShadowProperties`

El servicio `PolyPolygonShape` dispone asimismo de una propiedad que permite definir las coordenadas de un polígono:

- **`PolyPolygon (matriz)`**: campo que contiene las coordenadas del polígono (matriz doble con puntos del tipo `com.sun.star.awt.Point`)

En el ejemplo siguiente se muestra cómo se define un triángulo con el servicio `PolyPolygonShape`.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaPolipoligonal As Object
Dim Polipoligono As Variant
Dim Coordenadas(2) As New com.sun.star.awt.Point

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

FormaPolipoligonal = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")
Page.add(FormaPolipoligonal) ' Page.add debe aparecer antes de definir las coordenadas

Coordenadas(0).x = 1000
Coordenadas(1).x = 7500
Coordenadas(2).x = 10000
Coordenadas(0).y = 1000
Coordenadas(1).y = 7500
Coordenadas(2).y = 5000

FormaPolipoligonal.PolyPolygon = Array(Coordenadas())
```

Puesto que los puntos de un polígono se definen como valores absolutos, no es necesario especificar el tamaño o la posición inicial de un polígono. En cambio, es necesario crear una matriz de puntos, empaquetarla dentro de una segunda matriz (mediante la llamada `Array (Coordenadas())` y, a continuación, asignar esta matriz al polígono. Antes de poder efectuar la llamada correspondiente, el polígono debe insertarse en el documento.

La matriz doble de la definición permite crear formas complejas mediante la combinación de varios polígonos. Por ejemplo, se puede crear un rectángulo y, a continuación, insertar otro dentro de él para crear un hueco en el rectángulo original:

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaPolipoligonal As Object
Dim Polipoligono As Variant
Dim Cuadrado1(3) As New com.sun.star.awt.Point
Dim Cuadrado2(3) As New com.sun.star.awt.Point
```

```
Dim Cuadrado3(3) As New com.sun.star.awt.Point

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

FormaPolipoligonal = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")

Page.add(FormaPolipoligonal) ' Page.add debe aparecer antes de definir las coordenadas

Cuadrado1(0).x = 5000
Cuadrado1(1).x = 10000
Cuadrado1(2).x = 10000
Cuadrado1(3).x = 5000
Cuadrado1(0).y = 5000
Cuadrado1(1).y = 5000
Cuadrado1(2).y = 10000
Cuadrado1(3).y = 10000

Cuadrado2(0).x = 6500
Cuadrado2(1).x = 8500
Cuadrado2(2).x = 8500
Cuadrado2(3).x = 6500
Cuadrado2(0).y = 6500
Cuadrado2(1).y = 6500
Cuadrado2(2).y = 8500
Cuadrado2(3).y = 8500

Cuadrado3(0).x = 6500
Cuadrado3(1).x = 8500
Cuadrado3(2).x = 8500
Cuadrado3(3).x = 6500
Cuadrado3(0).y = 9000
Cuadrado3(1).y = 9000
Cuadrado3(2).y = 9500
Cuadrado3(3).y = 9500

FormaPolipoligonal.PolyPolygon = Array(Cuadrado1(), Cuadrado2(), Cuadrado3())
```

Para saber qué áreas están rellenas y cuáles son huecos, StarOffice aplica una regla sencilla: el borde de la forma exterior es siempre el borde exterior del polipolígono. La siguiente línea hacia dentro es el borde interior de la forma y marca la transición al primer hueco. Si hay otra línea más hacia el interior, ésta marca la transición hacia un área rellena.

Imágenes

El último de los elementos de dibujo analizado son los objetos gráficos (imágenes) basados en el servicio `com.sun.star.drawing.GraphicObjectShape`. Se pueden utilizar con cualquier imagen de StarOffice cuyo aspecto pueda adaptarse mediante el uso de una amplia gama de propiedades.

Los objetos gráficos admiten dos de las propiedades de formato generales:

- **Propiedades de texto**: `com.sun.star.drawing.Text` (con `com.sun.star.style.CharacterProperties` y `com.sun.star.style.ParagraphProperties`)
- **Propiedades de sombra**: `com.sun.star.drawing.ShadowProperties`

Las propiedades adicionales admitidas por los objetos gráficos son:

- **`GraphicURL (String)`**: URL del gráfico
- **`AdjustLuminance (Short)`**: luminancia de los colores en forma de porcentaje (se permiten valores negativos)
- **`AdjustContrast (Short)`**: contraste en forma de porcentaje (se permiten valores negativos)
- **`AdjustRed (Short)`**: valor de rojo en forma de porcentaje (se permiten valores negativos)
- **`AdjustGreen (Short)`**: valor de verde en forma de porcentaje (se permiten valores negativos)
- **`AdjustBlue (Short)`**: valor de azul en forma de porcentaje (se permiten valores negativos)
- **`Gamma (Short)`**: valor de gamma de un gráfico
- **`Transparency (Short)`**: transparencia de un gráfico en forma de porcentaje

`GraphicColorMode (Enum)`: modo de color; por ejemplo, estándar, tonos de gris, blanco y negro (valor predeterminado de acuerdo con `com.sun.star.drawing.ColorMode`)

En el ejemplo siguiente se muestra cómo insertar en una página un objeto gráfico. Dim Doc As Object

```
Dim Pagina As Object
Dim FormaObjetoGrafico As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000                      ' especificaciones no significativas porque las
coordenadas indicadas en último lugar son
                                    prevalentes
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaObjetoGrafico = Doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

FormaObjetoGrafico.Size = Tamano
FormaObjetoGrafico.Position = Punto

FormaObjetoGrafico.GraphicURL = "file:///c:/prueba.jpg"
FormaObjetoGrafico.AdjustBlue = -50
FormaObjetoGrafico.AdjustGreen = 5
```

```
FormaObjetoGrafico.AdjustBlue = 10
FormaObjetoGrafico.AdjustContrast = 20
FormaObjetoGrafico.AdjustLuminance = 50
FormaObjetoGrafico.Transparency = 40
FormaObjetoGrafico.GraphicColorMode = com.sun.star.drawing.ColorMode.STANDARD

Pagina.add(FormaObjetoGrafico)
```

En este código se inserta la imagen `prueba.jpg` y se adapta su aspecto mediante las propiedades `Adjust`. En el ejemplo las imágenes se representan con un 40 por ciento de transparencia, sin otras conversiones de color (`GraphicColorMode` = `STANDARD`).

Edición de objetos de dibujo

Agrupamiento de objetos

En muchas situaciones resulta conveniente agrupar varios objetos de dibujo individuales para que se comporten como un único objeto.

En el ejemplo siguiente se combinan dos objetos de dibujo:

```
Dim Doc As Object
Dim Pagina As Object
Dim Cuadrado As Object
Dim Circulo As Object
Dim Formas As Object
Dim Grupo As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size
Dim NuevaPos As New com.sun.star.awt.Point
Dim Altura As Long
Dim Ancho As Long

Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages(0)
Punto.x = 3000
Punto.y = 3000
Tamano.Width = 3000
Tamano.Height = 3000
' crear elemento de dibujo cuadrado
Cuadrado = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Cuadrado.Size = Tamano
Cuadrado.Position = Punto
Cuadrado.FillColor = RGB(255,128,128)
Pagina.add(Cuadrado)
' crear elemento de dibujo circular
Circulo = Doc.createInstance("com.sun.star.drawing.EllipseShape")
Circulo.Size = Tamano
Circulo.Position = Punto
```

```
Circulo.FillColor = RGB(255,128,128)
Circulo.FillColor = RGB(0,255,0)
Pagina.add(Circulo)
' combinar los elementos de dibujo cuadrado y circulo
Formas = createUnoService("com.sun.star.drawing.ShapeCollection")
Formas.add(Cuadrado)
Formas.add(Circulo)
Grupo = Pagina.group(Formas)
' centrar elementos de dibujo combinados
Altura = Pagina.Height
Ancho = Pagina.Width
PosNueva.X = Ancho / 2
PosNueva.Y = Altura / 2
Altura = Grupo.Size.Height
Ancho = Grupo.Size.Width
PosNueva.X = PosNueva.X - Ancho / 2
PosNueva.Y = PosNueva.Y - Altura / 2
Grupo.Position = PosNueva
```

Este código crea un rectángulo y un círculo y los inserta en una página. A continuación crea un objeto que admita el servicio `com.sun.star.drawing.ShapeCollection` y utiliza el método `Add` para agregar el rectángulo y el círculo a dicho objeto. `ShapeCollection` se añade a la página mediante el método `Group` y devuelve el objeto `Grupo` que puede editarse como un objeto `Shape` individual.

Si desea dar formato a los objetos individuales de un grupo, aplique el formato antes de agregar los objetos al grupo. Una vez agrupados los objetos, no es posible modificarlos.

Giro y distorsión de objetos de dibujo

Todos los objetos de dibujo descritos en las secciones anteriores se pueden girar y distorsionar mediante el servicio `com.sun.star.drawing.RotationDescriptor`.

Este servicio ofrece las propiedades siguientes:

- **`RotateAngle (Long)`**: ángulo de rotación en centésimas de grado
- **`ShearAngle (Long)`**: ángulo de distorsión en centésimas de grado

En el ejemplo siguiente se crea un rectángulo y se gira 30 grados mediante el uso de la propiedad `RotateAngle`:

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000
```

```
Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages(0)

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.RotateAngle = 3000

Pagina.add(FormaRectangular)
```

En el ejemplo siguiente se crea el mismo rectángulo que en el ejemplo anterior, pero en este caso se le aplica una distorsión de 30 grados mediante la propiedad `ShearAngle`.

```
Dim Doc As Object
Dim Pagina As Object
Dim FormaRectangular As Object
Dim Punto As New com.sun.star.awt.Point
Dim Tamano As New com.sun.star.awt.Size

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

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

FormaRectangular = Doc.createInstance("com.sun.star.drawing.RectangleShape")
FormaRectangular.Size = Tamano
FormaRectangular.Position = Punto

FormaRectangular.ShearAngle = 3000

Pagina.add(FormaRectangular)
```

Búsqueda y sustitución

Como sucede en los documentos de texto, los documentos de dibujo ofrecen una función de buscar y reemplazar que es similar a la utilizada en los documentos de texto descrita en el capítulo 6, ***Documentos de texto:***. Sin embargo, en los documentos de dibujo los objetos descriptores de buscar y reemplazar no se crean directamente a través del objeto documento, sino a través del nivel de carácter asociado. En el ejemplo siguiente se esquematiza el proceso de reemplazar dentro de un dibujo:

```
Dim Doc As Object
Dim Pagina As Object
Dim DescriptorSustitucion As Object
Dim I As Integer
```

```
Doc = StarDesktop.CurrentComponent
Pagina = Doc.drawPages(0)

DescriptorSustitucion = Pagina.createReplaceDescriptor()
DescriptorSustitucion.SearchString = "es"
DescriptorSustitucion.ReplaceString = "era"

For I = 0 to Doc.drawPages.Count - 1
      Pagina = Doc.drawPages(I)
      Pagina.ReplaceAll(DescriptorSustitucion)
Next I
```

En este fragmento de código se utiliza la primera `DrawPage` del documento para crear un `DescriptorSustitucion` que, a continuación, se aplica a todas las páginas del documento de dibujo mediante un bucle.

Presentaciones

Las presentaciones de StarOffice se basan en documentos de dibujo. Cada página de la presentación es una diapositiva a la que se puede acceder como se accede a un documento de dibujo estándar, a través de la lista `DrawPages` del objeto documento. El servicio `com.sun.star.presentation.PresentationDocument`, que se encarga de los documentos de presentación, ofrece también el servicio `com.sun.star.drawing.DrawingDocument` completo.

Trabajo con presentaciones

Además de las funciones de dibujo que ofrece la propiedad `Presentation`, el documento de presentación dispone de un objeto presentación que proporciona acceso a las principales propiedades y mecanismos de control para presentaciones. Por ejemplo, este objeto ofrece un método `start` para iniciar presentaciones.

```
Dim Doc As Object
Dim Presentacion As Object

Doc = StarDesktop.CurrentComponent
Presentacion = Doc.Presentation
Presentacion.start()
```

.Este fragmento de código crea un objeto `Doc` que hace referencia al documento de presentación actual y establece el objeto presentación asociado. El método `start()` del objeto se utiliza para iniciar el ejemplo y ejecutar la presentación en pantalla.

Se ofrecen los siguientes métodos como objetos de presentación:

- **`start`**: inicia la presentación
- **`end`**: finaliza la presentación
- **`rehearseTimings`**: inicia la presentación desde el principio y establece el tiempo de ejecución

También están disponibles las propiedades siguientes:

- **`AllowAnimations (booleano)`**: ejecuta animaciones en la presentación

- **CustomShow (String)**: permite especificar el nombre de la presentación, de manera que dentro de ésta se pueda hacer referencia a aquél
- **FirstPage (String)**: nombre de la diapositiva con la que desee iniciar la presentación
- **IsAlwaysOnTop (booleano)**: muestra siempre la ventana de presentación como la primera ventana en la pantalla
- **IsAutomatic (booleano)**: ejecuta automáticamente la presentación
- **IsEndless (booleano)**: cuando la presentación se termina la reinicia desde el principio
- **IsFullScreen (booleano)**: inicia automáticamente la presentación en modo de pantalla completa
- **IsMouseVisible (booleano)**: muestra el ratón durante la presentación
- **Pause (long)**: tiempo que se muestra una pantalla en blanco al finalizar la presentación
- **StartWithNavigator (booleano)**: muestra la ventana del navegador al iniciarse la presentación
- **UsePn (booleano)**: muestra el puntero durante la presentación

CAPÍTULO 9

Diagramas

StarOffice puede mostrar datos en forma de diagramas, creando así vínculos gráficos entre los datos en forma de barras, sectores, líneas u otros elementos, que se pueden mostrar en 2D o 3D; el aspecto de los elementos se puede adaptar individualmente de forma similar a lo que ocurre con los elementos de dibujo.

Si los datos están disponibles en forma de hoja de cálculo, ésta se puede vincular con el diagrama de forma dinámica, con lo que las modificaciones efectuadas en los datos de base se podrán ver de inmediato en el diagrama asignado. Este capítulo ofrece un resumen de la interfaz de programación para los módulos de diagramas de StarOffice y se centra en el uso de diagramas dentro de documentos de hoja de cálculo.

Uso de diagramas en hojas de cálculo

En StarOffice, los diagramas no se tratan como documentos independientes, sino como objetos incrustados en un documento previo.

Mientras que, en los documentos de texto y de dibujo, los diagramas permanecen aislados del contenido del documento, en el caso de los documentos de hoja de cálculo existe un mecanismo que permite establecer un vínculo entre los datos del documento y los diagramas incrustados en él.

En el ejemplo siguiente se explica la interacción entre el documento de hoja de cálculo y el diagrama:

```
Dim Doc As Object
Dim Diagramas As Object
Dim Diagrama As Object

Dim Rectangulo As New com.sun.star.awt.Rectangle
Dim DireccionArea(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Diagramas = Doc.Sheets(0).Charts

Rectangulo.X = 8000
Rectangulo.Y = 1000
Rectangulo.Ancho = 10000
Rectangulo.Altura = 7000

DireccionArea(0).Sheet = 0
DireccionArea(0).StartColumn = 0
DireccionArea(0).StartRow = 0
DireccionArea(0).EndColumn = 2
```

```
DireccionArea(0).EndRow = 12

Diagramas.addNewByName("MiDiagrama", Rectangulo, DireccionArea(), True, True)
```

Aunque el código utilizado en el ejemplo pueda parecer complicado, los procesos esenciales se limitan a tres líneas: la primera línea esencial crea la variable de documento `Doc`, que hace referencia al documento de hoja de cálculo actual (línea `Doc = StarDesktop.CurrentComponent`). El código del ejemplo crea a continuación una lista con todos los diagramas de la primera hoja de cálculo (línea `Diagramas = Doc.Sheets(0).Charts`). Finalmente se añade un diagrama nuevo a la última línea de esta lista mediante el método `addNewByName`. Este nuevo diagrama es visible para el usuario.

En la última línea se inicializan las estructuras auxiliares `Rectangulo` y `DireccionArea` que el método `addNewByName` ofrece también como parámetro. `Rectangulo` determina la posición del diagrama dentro de la hoja de cálculo. `DireccionArea` determina el área cuyos datos se deben vincular al diagrama.

En el ejemplo anterior se crea un diagrama de barras. Si se necesita un tipo de gráfico distinto, el diagrama de barras se debe sustituir de forma explícita:

```
Diagrama = Diagramas.getByName("MiDiagrama").embeddedObject
Diagrama.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

La primera línea define el objeto diagrama correspondiente. La segunda línea sustituye el diagrama actual por uno nuevo (en este caso, uno de líneas).

En Excel se distingue entre los diagramas insertados como página independiente en un documento de Excel y aquellos que se han incrustado en una página de tabla. Por consiguiente se han definido dos métodos de acceso distinto a los diagramas. Esta distinción no existe en StarOffice Basic, porque los diagramas de StarOffice Calc se crean siempre como objetos incrustados en una página de tabla. El acceso a los diagramas se efectúa siempre mediante la lista `Charts` del objeto `Sheet` asociado.

Estructura de los diagramas

La estructura de un diagrama (y, por consiguiente, la lista de servicios e interfaces que admite) depende del tipo de diagrama. Por ejemplo, los métodos y propiedades relacionados con el eje Z sólo están disponibles en diagramas 3D, no en diagramas 2D. En los diagramas de sectores no hay interfaces para trabajar con ejes.

Elementos individuales de un diagrama

Título, subtítulo y clave

El título, el subtítulo y la clave forman parte de los elementos esenciales de cualquier diagrama. Los diagramas proporcionan sus propios objetos para cada uno de estos elementos. El objeto `Chart` (diagrama) ofrece las propiedades siguientes para la administración de estos elementos:

- **`HasMainTitle (booleano)`**: activa el título

- **`Title (objeto)`**: objeto con información detallada acerca del título del diagrama (admite el servicio `com.sun.star.chart.ChartTitle`).
- **`HasSubTitle (booleano)`**: activa el subtítulo
- **`Subtitle (objeto)`**: objeto con información detallada acerca del subtítulo del diagrama (admite el servicio `com.sun.star.chart.ChartTitle`).
- **`HasLegend (booleano)`**: activa la clave
- **Legend (objeto)**: objeto con información detallada acerca de la clave del diagrama (admite el servicio `com.sun.star.chart.ChartLegendPosition`).

En muchos sentidos, los elementos especificados corresponden a elementos de dibujo. El motivo es que tanto `com.sun.star.chart.ChartTitle` como `com.sun.star.chart.ChartLegendPosition` admiten el servicio `com.sun.star.drawing.Shape`, que constituye la base técnica de los elementos de dibujo.

Por consiguiente los usuarios tiene la oportunidad de determinar la posición y el tamaño del elemento mediante las propiedades `Size` y `Position`.

También se incorporan el resto de propiedades de relleno y de línea (servicios `com.sun.star.drawing.FillProperties` y `com.sun.star.drawing.LineStyle`), así como las de carácter (`com.sun.star.style.CharacterProperties`) para dar formato a los elementos.

`com.sun.star.chart.ChartTitle` no sólo contiene las propiedades de formato mencionadas, sino también dos propiedades adicionales:

- **`TextRotation (Long)`**: ángulo de rotación del texto en centésimas de grado
- **`String (String)`**: texto que se debe mostrar como título o subtítulo.

La clave del diagrama (servicio `com.sun.star.chart.ChartLegend`) contiene la siguiente propiedad adicional:

- **`Alignment (Enum)`**: posición en la que aparece la clave (valor predeterminado de acuerdo con `com.sun.star.chart.ChartLegendPosition`).

En el ejemplo siguiente se crea un diagrama y se le asigna el título "Prueba", el subtítulo "Prueba 2" y una clave. Está tiene el fondo de color gris, se encuentra en la parte inferior del diagrama y el tamaño de sus caracteres es de 7 puntos.

```
Dim Doc As Object
Dim Diagramas As Object
Dim Diagrama As Object

Dim Rectangulo As New com.sun.star.awt.Rectangle
Dim DireccionArea(0) As New com.sun.star.table.CellRangeAddress

Rectangulo.X = 8000
Rectangulo.Y = 1000
Rectangulo.Ancho = 10000
Rectangulo.Altura = 7000
```

```
DireccionArea(0).Sheet = 0
DireccionArea(0).StartColumn = 0
DireccionArea(0).StartRow = 0
DireccionArea(0).EndColumn = 2
DireccionArea(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Diagramas = Doc.Sheets(0).Charts
Diagramas.addNewByName("MiDiagrama", Rectangulo, DireccionArea(), True, True)
Diagrama = Diagramas.getByName("MiDiagrama").EmbeddedObject

Diagrama.HasMainTitle = True
Diagrama.Title.String = "Prueba"

Diagrama.HasSubTitle = True
Diagrama.Subtitle.String = "Prueba 2"

Diagrama.HasLegend = True
Diagrama.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Diagrama.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Diagrama.Legend.FillColor = RGB(210, 210, 210)
Diagrama.Legend.CharHeight = 7
```

Fondo

Todos los diagramas tienen un fondo. Todas las áreas tienen un objeto correspondiente al que se puede acceder a través de las propiedades siguientes del objeto de diagrama:

- **`Area (objeto)`**: área del fondo del diagrama (admite el servicio `com.sun.star.chart.ChartArea`).

El fondo de un diagrama cubre toda el área de ésta, incluida la situada bajo el título, el subtítulo y la clave del diagrama. El servicio `com.sun.star.chart.ChartArea` asociado admite propiedades de línea y de relleno y no proporciona más propiedades extensivas.

Planos laterales y base del diagrama

Aunque el fondo del diagrama cubre toda el área, el plano posterior del diagrama se limita a la situada detrás del área de datos.

En los diagramas 3D suele haber dos planos laterales: uno detrás del área de datos y otro que actúa como límite izquierdo del eje Y. Los diagramas 3D suelen tener también una base.

- **`Floor (objeto)`**: panel que forma la base del diagrama (sólo en diagramas 3D, admite el servicio `com.sun.star.chart.ChartArea`).
- **`Wall (objeto)`**: paneles laterales del diagrama (sólo en diagramas 3D, admite el servicio `com.sun.star.chart.ChartArea`).

Los objetos especificados admiten el servicio `com.sun.star.chart.ChartArea` que, a su vez, proporciona las propiedades habituales de línea y de relleno (servicios

com.sun.star.drawing.FillProperties y com.sun.star.drawing.LineStyle; consulte el capítulo 8).

Para acceder a los paneles laterales y la base del diagrama se emplea el objeto Chart que, a su vez, forma parte del objeto Chart:

```
Chart.Area.FillBitmapName = "Sky"
```

En el ejemplo siguiente se muestra cómo puede utilizarse una imagen (denominada Sky) contenida en StarOffice como fondo de un diagrama.

```
Dim Doc As Object
Dim Diagramas As Object
Dim Diagrama As Object

Dim Rectangulo As New com.sun.star.awt.Rectangle
Dim DireccionArea(0) As New com.sun.star.table.CellRangeAddress

Rectangulo.X = 8000
Rectangulo.Y = 1000
Rectangulo.Ancho = 10000
Rectangulo.Altura = 7000

DireccionArea(0).Sheet = 0
DireccionArea(0).StartColumn = 0
DireccionArea(0).StartRow = 0
DireccionArea(0).EndColumn = 2
DireccionArea(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Diagramas = Doc.Sheets(0).Charts

Diagramas.addNewByName("MiDiagrama", Rectangulo, DireccionArea(), True, True)
Diagrama = Diagramas.getByName("MiDiagrama").EmbeddedObject

Diagrama.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Diagrama.Area.FillBitmapName = "Sky"
Diagrama.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Ejes

StarOffice reconoce cinco ejes distintos que se pueden utilizar en un diagrama. En el escenario más sencillo, los ejes son el X y el Y. Cuando se trabaja con diagramas 3D, a veces se proporciona también un eje Z. Para los diagramas en los que los valores de las distintas filas de datos se desvían entre sí de forma significativa, StarOffice ofrece un segundo par de ejes X e Y para operaciones en otra escala.

Primeros ejes X, Y y Z

Además del eje en sí, cada uno de los primeros ejes X, Y y Z pueden tener asociados un título, una descripción, una cuadrícula y una cuadrícula auxiliar. Existe la opción de mostrar u ocultar cada uno de estos elementos. El objeto diagrama ofrece las propiedades siguientes para la administración de

las características mencionadas (se toma como ejemplo el eje X; las propiedades de los ejes Y y Z se estructuran de la misma forma):

- **HasXAxis (booleano)**: activa el eje X
- **XAxis (objeto)**: objeto con información detallada acerca del eje X (admite el servicio `com.sun.star.chart.ChartAxis`).
- **HasXAxisDescription (booleano)**: activa la descripción del eje X
- **HasXAxisGrid (booleano)**: activa la cuadrícula principal del eje X
- **XMainGrid (objeto)**: objeto con información detallada acerca de la cuadrícula principal del eje X (admite el servicio `com.sun.star.chart.ChartGrid`).
- **HasXAxisHelpGrid (booleano)**: activa la cuadrícula auxiliar del eje X
- **XHelpGrid (objeto)**: objeto con información detallada acerca de la cuadrícula auxiliar del eje X (admite el servicio `com.sun.star.chart.ChartGrid`).
- **HasXAxisTitle (booleano)**: activa el título del eje X
- **XAxisTitle (objeto)**: objeto con información detallada acerca del título del eje X (admite el servicio `com.sun.star.chart.ChartTitle`).

Segundos ejes X e Y

Los segundos ejes X e Y disponen de las propiedades siguientes (tomando como ejemplo el segundo eje X):

- **HasSecondaryXAxis (booleano)**: activa el segundo eje X
- **SecondaryXAxis (objeto)**: objeto con información detallada acerca del segundo eje X (admite el servicio `com.sun.star.chart.ChartAxis`).
- **HasSecondaryXAxisDescription (booleano)**: activa la descripción del eje X

Propiedades de los ejes

Los objetos eje de un diagrama de StarOffice admiten el servicio `com.sun.star.chart.ChartAxis`. Además de las propiedades de caracteres (servicio `com.sun.star.style.CharacterProperties`, consulte el capítulo 6) y líneas (servicio `com.sun.star.drawing.LineStyle`, consulte el capítulo 8), proporciona las propiedades siguientes:

- **Max (Double)**: valor máximo del eje.
- **Min (Double)**: valor mínimo del eje.
- **Origin (Double)**: punto de intersección de ejes que se cruzan.
- **StepMain (Double)**: distancia entre dos líneas de división principales del eje.
- **StepHelp (Double)**: distancia entre dos líneas de división secundarias del eje.
- **AutoMax (booleano)**: determina automáticamente el valor máximo para el eje.

- **AutoMin (booleano)**: determina automáticamente el valor mínimo para el eje.
- **AutoOrigin (booleano)**: determina automáticamente el punto de intersección de ejes que se cruzan.
- **AutoStepMain (booleano)**: determina automáticamente la distancia entre las líneas de división principales del eje.
- **AutoStepHelp (booleano)**: determina automáticamente la distancia entre las líneas de división secundarias del eje.
- **Logarithmic (booleano)**: aplica escala logarítmica (en lugar de lineal) a los ejes.
- **DisplayLabels (booleano)**: activa la etiqueta de texto de los ejes.
- **TextRotation (Long)**: ángulo de rotación de la etiqueta de texto de los ejes en centésimas de grado
- **Marks (Const)**: constante que especifica si las líneas de división principales del eje deben estar dentro o fuera del área del diagrama (valores predeterminados de acuerdo con `com.sun.star.chart.ChartAxisMarks`)
- **HelpMarks (Const)**: constante que especifica si las líneas de división secundarias del eje deben estar dentro o fuera del área del diagrama (valores predeterminados de acuerdo con `com.sun.star.chart.ChartAxisMarks`)
- **Overlap (Long)**: porcentaje que especifica la cuantía de la superoposición de las barras correspondientes a distintos grupos de datos (un valor del 100% significa que las barras se muestran totalmente superpuestas; un valor del -100% indica que hay una distancia de una barra entre ellas).
- **GapWidth (long)**: porcentaje que especifica la distancia que puede haber entre los distintos grupos de barras en un diagrama (un valor del 100% significa que la distancia es del ancho de una barra).
- **ArrangeOrder (enum)**: detalles de la posición de la inscripción; aparte de situar una línea, existe la opción de dividir la etiqueta en dos líneas (valor predeterminado de acuerdo con `com.sun.star.chart.ChartAxisArrangeOrderType`)
- **TextBreak (booleano)**: permite los saltos de línea.
- **TextCanOverlap (booleano)**: permite la superposición de texto.
- **NumberFormat (Long)**: formato numérico (consulte el capítulo 7, sección ***Formatos de número, fecha y texto***)

Propiedades de la cuadrícula de ejes

El objeto correspondiente a la cuadrícula de los ejes se basa en el servicio `com.sun.star.chart.ChartGrid` que, a su vez, admite las propiedades de línea del servicio `com.sun.star.drawing.LineStyle` (consulte el capítulo 8).

Propiedades del título del eje

Los objetos para dar formato al título del eje se basan en el servicio `com.sun.star.chart.ChartTitle` que se utiliza también en los títulos de diagramas.

Ejemplo

En el ejemplo siguiente se crea un diagrama de líneas. El color del panel posterior del diagrama se establece en blanco. Ambos ejes, X e Y, disponen de una cuadrícula auxiliar gris para una mejor orientación visual. El valor mínimo del eje Y se establece en 0 y el valor máximo se fija en 100, de modo que la resolución del diagrama se conserva aunque los valores se modifiquen.

```
Dim Doc As Object
Dim Diagramas As Object
Dim Diagrama As Object

Dim Rectangulo As New com.sun.star.awt.Rectangle
Dim DireccionArea(0) As New com.sun.star.table.CellRangeAddress

Doc = StarDesktop.CurrentComponent
Diagramas = Doc.Sheets(0).Charts

Rectangulo.X = 8000
Rectangulo.Y = 1000
Rectangulo.Ancho = 10000
Rectangulo.Altura = 7000

DireccionArea(0).Sheet = 0
DireccionArea(0).StartColumn = 0
DireccionArea(0).StartRow = 0
DireccionArea(0).EndColumn = 2
DireccionArea(0).EndRow = 12

Diagramas.addNewByName("MiDiagrama", Rectangulo, DireccionArea(), True, True)

Diagrama = Diagramas.getByName("MiDiagrama").embeddedObject
Diagrama.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")

Diagrama.Diagram.Wall.FillColor = RGB(255, 255, 255)

Diagrama.Diagram.HasXAxisGrid = True
Diagrama.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)

Diagrama.Diagram.HasYAxisGrid = True
Diagrama.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)
Diagrama.Diagram.YAxis.Min = 0
Diagrama.Diagram.YAxis.Max = 100
```

Diagramas 3D

La mayoría de los diagramas de StarOffice se pueden mostrar también con gráficos 3D. Todos los tipos de diagramas que permiten esta opción admiten el servicio `com.sun.star.chart.Dim3DDiagram`. Este servicio ofrece una única propiedad:

- **`Dim3D (booleano)`**: activa la visualización 3D.

Diagramas apilados

Los diagramas apilados están dispuestos de modo que varios valores se coloquen uno sobre el otro para producir un valor total. En esta vista no sólo se muestran los valores individuales, sino también un resumen de todos los valores.

Diversos tipos de diagramas en StarOffice se pueden mostrar en forma de diagramas apilados. Todos ellos admiten el servicio `com.sun.star.chart.StackableDiagram` que, a su vez, proporciona las propiedades siguientes:

- **`Stacked (booleano)`**: activa el modo de visualización apilada.
- **`Percent (booleano)`**: en lugar de valores absolutos se muestra su distribución porcentual.

Tipos de diagramas

Diagramas de líneas

Los diagramas de líneas (servicio `com.sun.star.chart.LineDiagram`) admiten un eje X, dos ejes Y y un eje Z. Se pueden mostrar en 2D o en 3D (servicio `com.sun.star.chart.Dim3Ddiagram`). Las líneas pueden apilarse (`com.sun.star.chart.StackableDiagram`).

Los diagramas de líneas ofrecen las propiedades siguientes:

- **`SymbolType (const)`**: símbolo para mostrar los puntos de datos (constante de acuerdo con `com.sun.star.chart.ChartSymbolType`).
- **`SymbolSize (Long)`**: tamaño del símbolo para mostrar los puntos de datos en centésimas de milímetro.
- **`SymbolBitmapURL (String)`**: nombre de archivo de la imagen empleada para mostrar los puntos de datos.
- **`Lines (booleano)`**: enlaza los puntos de datos mediante líneas.
- **`SplineType (Long)`**: función spline para suavizar las líneas (0: sin función de spline, 1: splines cúbicas , 2: splines B).
- **`SplineOrder (Long)`**: grado polinómico de las splines (sólo para splines B).
- **`SplineResolution (Long)`**: número de puntos de apoyo para el cálculo de splines.

Diagramas de área

Los diagramas de área (servicio `com.sun.star.chart.AreaDiagram`) admiten un eje X, dos ejes Y y un eje Z. Se pueden mostrar en 2D o en 3D (servicio `com.sun.star.chart.Dim3Ddiagram`). Las áreas pueden apilarse (`com.sun.star.chart.StackableDiagram`).

Diagramas de barras

Los diagramas de barras (servicio `com.sun.star.chart.BarDiagram`) admiten un eje X, dos ejes Y y un eje Z. Se pueden mostrar en 2D o en 3D (servicio `com.sun.star.chart.Dim3Ddiagram`). Las barras pueden apilarse (`com.sun.star.chart.StackableDiagram`).

Ofrecen las propiedades siguientes:

- **`Vertical (booleano)`:** muestra las barras verticalmente; en caso contrario se muestran de forma horizontal.
- **`Deep (booleano)`**: en modo de visualización 3D, sitúa las barras una detrás de otra en lugar de una junto a la otra.
- **`StackedBarsConnected (booleano)`**: enlaza mediante líneas las barras asociadas en un diagrama apilado (sólo está disponible en diagramas horizontales).
- **NumberOfLines (Long)**: número de líneas que se debe mostrar en un diagrama apilado como líneas en lugar de como barras.

Diagramas de sectores

Los diagramas de sectores (servicio `com.sun.star.chart.PieDiagram`) no contienen ejes y no pueden apilarse. Se pueden mostrar en 2D o en 3D (servicio `com.sun.star.chart.Dim3Ddiagram`).

CAPÍTULO **10**

Acceso a bases de datos

StarOffice tiene integrada una interfaz de base de datos (independiente de cualquier sistema) denominada Star Database Connectivity (SDBC). Al desarrollar esta interfaz se ha intentado proporcionar acceso al máximo número posible de fuentes de datos distintas.

Para ello, se accede a las fuentes de datos mediante controladores. Las fuentes de las cuales éstos toman los datos son irrelevantes para el usuario de SDBC. Algunos controladores acceden a bases de datos basadas en archivos y toman los datos directamente de estos. Otros emplean interfaces estándar como JDBC u ODBC. No obstante, hay algunos controladores especiales que acceden a la libreta de direcciones MAPI, a los directorios LDAP o a hojas de cálculo de StarOffice como fuentes de datos.

Los controladores se basan en componentes UNO, por lo que es posible desarrollar otros controladores y así incorporar nuevas fuentes de datos. StarOffice Developer's Guide ofrece más detalles al respecto.

Desde un punto de vista conceptual, SDBC es comparable a las bibliotecas ADO y DAO disponibles en VBA. Permite acceder a alto nivel a bases de datos, independientemente de los mecanismos de base de datos subyacentes.

La interfaz de base de datos de StarOffice ha crecido con la aparición de la versión StarOffice 6. Aunque anteriormente el acceso a las bases de datos se efectuaba principalmente mediante diversos métodos del objeto `Application`, la interfaz de StarOffice 6 se subdivide en varios objetos. `DatabaseContext` se utiliza como objeto raíz para las funciones de base de datos.

SQL: un lenguaje de consultas

El lenguaje de consultas incluido para usuarios de SDBC es SQL. Para comparar las diferencias entre los distintos dialectos de SQL, los componentes SDBC de StarOffice tienen su propio analizador sintáctico SQL que utiliza la ventana de consulta para verificar las órdenes SQL y corrige los errores de sintaxis simples, como los relativos a mayúsculas y minúsculas.

Si un controlador permite acceder a una fuente de datos que no admita SQL, se deberán convertir independientemente las órdenes SQL transferidas al acceso nativo requerido.

La implementación SQL de SDBC está orientada hacia el estándar SQL-ANSI. Las extensiones específicas de Microsoft, como la construcción `INNER JOIN`, no se admiten y se deben sustituir por órdenes estándar (por ejemplo, `INNER JOIN` se debe sustituir por la cláusula `WHERE` correspondiente).

Tipos de acceso a bases de datos

La interfaz de base de datos de StarOffice está disponible en las aplicaciones StarOffice Writer y StarOffice Calc, así como en los formularios de base de datos.

En StarOffice Writer se pueden crear cartas estándar a partir de fuentes de datos SDBC e imprimirlas. También se ofrece la opción de trasladar datos de la ventana de base de datos a los documentos de texto mediante la función de arrastrar y colocar.

Si el usuario mueve una tabla de base de datos a una hoja de cálculo, StarOffice crea un área de tabla que se puede actualizar con una pulsación del ratón si los datos originales se han modificado. De igual forma, se pueden mover datos de una hoja de cálculo a una tabla de base de datos y efectuar una importación de base de datos.

Finalmente, StarOffice ofrece un mecanismo para crear formularios fundamentados en bases de datos. Para ello el usuario debe crear en primer lugar un formulario estándar de StarOffice Writer o de StarOffice Calc y luego vincular los campos a una base de datos.

Todas las opciones especificadas aquí se basan en la interfaz de usuario de StarOffice. No es necesario ningún conocimiento de programación para utilizar las funciones correspondientes.

Sin embargo, este capítulo no ofrece mucha información acerca de las funciones especificadas, sino que se centra en la interfaz de programación de SDBC, que permite automatizar las consultas a las bases de datos y ofrece, por tanto, una mayor variedad de aplicaciones.

Sin embargo, es necesario un conocimiento básico del funcionamiento de las bases de datos y del lenguaje de consultas SQL para entender por completo las secciones siguientes.

Fuentes de datos

Para incorporar una base de datos a StarOffice se debe crear lo que se suele denominar una a ***fuente de datos***. La interfaz de usuario proporciona una opción para crear fuentes de datos en el menú **Opciones**. En todo caso, también se pueden crear fuentes de datos y trabajar con ellas mediante el propio StarOffice Basic.

Un objeto de contexto de base de datos creado mediante la función `createUnoService` actúa como punto de partida para acceder a una fuente de datos. Este objeto se basa en el servicio `com.sun.star.sdb.DatabaseContext` y se trata del objeto raíz para todas las operaciones de base de datos.

En el ejemplo siguiente se muestra cómo se puede crear un contexto de base de datos y utilizarlo para determinar los nombres de todas las fuentes de datos disponibles. Muestra los nombres en un cuadro de mensaje.

```
Dim ContextoBaseDatos As Object
Dim Nombres
Dim I As Integer

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")

Nombres = ContextoBaseDatos.getElementNames()
```

```
For I = 0 To UBound(Nombres())
      MsgBox Nombres(I)
Next I
```

Las fuentes de datos individuales se basan en el servicio `com.sun.star.sdb.DataSource` y se pueden determinar a partir del contexto de base de datos mediante el método `getByName`:

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")
```

En el ejemplo se crea un objeto `FuenteDatos` para una fuente de datos denominada ***Clientes***.

Las fuentes de datos ofrecen diversas propiedades que, a su vez, proporcionan información general acerca del origen de los datos y sobre los métodos de acceso. Estas propiedades son:

- **`Name (String)`**: nombre de la fuente de datos.
- **`URL (String)`**: URL de la fuente de datos con el formato ***jdbc: subprotocolo : subnombre*** o ***sdbc: subprotocolo : subnombre***.
- **`Info (Array)`**: matriz que contiene pares `PropertyValue` con parámetros de conexión (al menos nombre de usuario y contraseña, en general).
- **`User (String)`**: nombre del usuario.
- **`Password (String)`**: contraseña del usuario (no se guarda).
- **`IsPasswordRequired (booleano)`**: se necesita la contraseña y se solicita al usuario de forma interactiva.
- **`IsReadOnly (booleano)`**: permite acceso de sólo lectura a la base de datos.
- **`NumberFormatsSupplier (Object)`**: objeto que contiene los formatos numéricos disponibles para la base de datos (admite la interfaz `com.sun.star.util.XNumberFormatsSupplier`; consulte el capítulo 7, sección ***Formatos de número, fecha y texto***).
- **`TableFilter (Array)`**: lista de los nombres de tabla que se deben mostrar.
- **`TableTypeFilter (Array)`**: lista de los tipos de tabla que se deben mostrar. Los valores disponibles son `TABLE`, `VIEW` y `SYSTEM TABLE`.
- **`SuppressVersionColumns (Booleano)`**: suprime la visualización de las columnas que se utilizan para la administración de versiones.

Las fuentes de datos de StarOffice no son totalmente equivalentes con las fuentes de datos de ODBC. Mientras que éstas cubren únicamente información acerca del origen de los datos, aquéllas proporcionan también información acerca de cómo se muestran los datos dentro de las ventanas de base de datos de StarOffice.

Consultas

Se pueden asignar a una fuente de datos consultas predefinidas. StarOffice toma nota de las órdenes SQL de las consultas para que estén siempre disponibles. Las consultas se utilizan para

simplificar el trabajo con bases de datos porque se pueden abrir con una simple pulsación del ratón y ofrecen a los usuarios sin conocimientos de SQL la opción de emitir órdenes SQL.

Detrás de un consulta se oculta un objeto que admite el servicio `com.sun.star.sdb.QueryDefinition`. A las consultas se accede mediante el método `QueryDefinitions` de la fuente de datos.

En el ejemplo siguiente se listan en un cuadro de mensaje los nombres de las consultas de la fuente de datos que se pueden establecer.

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object
Dim DefinicionesConsultas As Object
Dim DefinicionConsulta As Object
Dim I As Integer

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")
DefinicionesConsultas = FuenteDatos.getQueryDefinitions()

For I = 0 To DefinicionesConsultas.Count() - 1
      DefinicionConsulta = DefinicionesConsultas(I)
      MsgBox DefinicionConsulta.Name
Next I
```

Además de la propiedad `Name` que se utiliza en el ejemplo, `com.sun.star.sdb.QueryDefinition` ofrece una amplia gama de otras propiedades. Éstas son:

- **`Name (String)`**: nombre de la consulta.
- **`Command (String)`**: orden SQL (generalmente una orden `SELECT`).
- **`UpdateTableName (String)`**: en consultas basadas en varias tablas, nombre de la tabla en la que son posibles modificaciones de valor.
- **`UpdateCatalogName (String)`**: nombre de los catálogos de actualización de tablas.
- **`UpdateSchemaName (String)`**: nombre de los diagramas de actualización de tablas.

En el ejemplo siguiente se muestra cómo crear un objeto de consulta con control de programa y cómo asignarlo a una fuente de datos.

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object
Dim DefinicionesConsultas As Object
Dim DefinicionConsulta As Object
Dim I As Integer

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")
DefinicionesConsultas = FuenteDatos.getQueryDefinitions()

DefinicionConsulta = createUnoService("com.sun.star.sdb.QueryDefinition")
```

```
DefinicionConsulta.Command = "SELECT * FROM Cliente"

DefinicionesConsultas.insertByName("NuevaConsulta", DefinicionConsulta)
```

El objeto de consulta se crea en primer lugar mediante la llamada `createUnoService` y, a continuación, se inserta en el objeto `DefinicionesConsultas` mediante `insertByName`.

Vínculos con formularios de base de datos

Para simplificar el trabajo con las fuentes de datos, StarOffice ofrece una opción para vincular éstas con formularios de base de datos. Los vínculos están disponibles a través del método `getBookmarks()` que devuelve un contenedor con nombre (`com.sun.star.sdb.DefinitionContainer`) que contiene todos los vínculos de la fuente de datos. Se puede acceder a los marcadores a través de `Name` o de `Index`.

En el ejemplo siguiente se determina la URL del marcador ***MiMarcador***.

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object
Dim Marcadores As Object
Dim URL As String
Dim I As Integer

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")
Marcadores = FuenteDatos.Bookmarks()

URL = Marcadores.getByName("MiMarcador")
MsgBox URL
```

Acceso a bases de datos

Para acceder a una base de datos es necesaria una conexión de base de datos, un canal de transferencia que permita una comunicación directa con la base de datos. A diferencia de las fuentes de datos presentadas en la sección anterior, la conexión de base de datos debe volver a establecerse cada vez que se reinicie el programa.

StarOffice ofrece varias formas de establecer conexiones de base de datos. Ésta es una explicación del método basado en una fuente de datos ya existente.

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object
Dim Conexion As Object
Dim ManejadorInteraccion as Object

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")

If Not FuenteDatos.IsPasswordRequired Then
      Conexion = FuenteDatos.GetConnection("","")
```

```
Else
      ManejadorInteraccion = createUnoService("com.sun.star.sdb.InteractionHandler")
      Conexion = FuenteDatos.ConnectWithCompletion(ManejadorInteraccion)
End If
```

El código de este ejemplo comprueba, en primer lugar, si la base de datos está protegida por una contraseña. En caso contrario crea la conexión de base de datos requerida con la llamada `GetConnection`. Las dos cadenas vacías de la línea de órdenes representan el nombre de usuario y la contraseña.

Si la base de datos está protegida por una contraseña, el ejemplo crea un `ManejadorInteraccion` y abre la conexión de base de datos mediante el método `ConnectWithCompletion`. El ManejadorInteraccion garantiza que StarOffice solicite al usuario los datos de inicio de sesión requeridos.

Iteración de tablas

Para acceder a una tabla en StarOffice se suele emplear el objeto `ResultSet` (conjunto de resultados). Un `ResultSet` es un tipo de marcador que indica un conjunto de datos actual dentro de un volumen de resultados obtenido a partir de una orden `SELECT`.

En el ejemplo se muestra de qué forma se puede utilizar `ResultSet` para consultar valores de una tabla de base de datos.

```
Dim ContextoBaseDatos As Object
Dim FuenteDatos As Object
Dim Conexion As Object
Dim ManejadorInteraccion as Object
Dim Expresion As Object
Dim ResultSet As Object

ContextoBaseDatos = createUnoService("com.sun.star.sdb.DatabaseContext")
FuenteDatos = ContextoBaseDatos.getByName("Clientes")

If Not FuenteDatos.IsPasswordRequired Then
      Conexion = FuenteDatos.GetConnection("","")
Else
      ManejadorInteraccion = createUnoService("com.sun.star.sdb.InteractionHandler")
      Conexion = FuenteDatos.ConnectWithCompletion(ManejadorInteraccion)
End If

Expresion = Conexion.createStatement()
ResultSet = Expresion.executeQuery("SELECT NumeroCliente FROM Cliente")

If Not IsNull(ResultSet) Then
      While ResultSet.next
            MsgBox ResultSet.getString(1)
      Wend
End If
```

Una vez establecida la conexión de base de datos, el código utilizado en el ejemplo, en primer lugar, utiliza la llamada `Connection.createObject` para crear un objeto `Expresion` El cual, a continuación, utiliza la llamada `executeQuery` para devolver el propio `ResultSet`. El programa comprueba si `ResultSet` existe realmente y recorre los registros de datos mediante un bucle. Los

valores requeridos (en el ejemplo, los del campo `NumeroCliente`) devuelven `ResultSet` mediante el método `getString`, mientras que el parámetro 1 determina que la llamada es relativa a los valores de la primera columna.

El objeto `ResultSet` de SDBC es similar al objeto `Recordset` de DAO y ADO, ya que también ofrece acceso iterativo a una base de datos.

En StarOffice 6.x el acceso a la base de datos se efectúa realmente a través de un objeto `ResultSet` que refleja el contenido de una tabla o el resultado de una orden SELECT de SQL. Antiguamente, el objeto `ResultSet` ofrecía los métodos residentes del objeto `Application` para desplazarse en los datos (por ejemplo, `DataNextRecord`).

Métodos específicos de cada tipo para recuperar valores

Como se puede ver en el ejemplo de la sección anterior, StarOffice proporciona un método `getString` para acceder al contenido de las tablas que proporciona el resultado en forma de cadena de caracteres. Están disponibles los siguientes métodos de `get`:

- `getByte()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getShort()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getInt()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getLong()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getFloat()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getDouble()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getBoolean()`: admite los tipos de datos SQL para números, caracteres y cadenas.
- `getString()`: admite todos los tipos de datos SQL.
- `getBytes()`: admite los tipos de datos SQL para valores binarios.
- `getDate()`: admite los tipos de datos SQL para números, cadenas y marcas de fecha y hora.
- `getTime()`: admite los tipos de datos SQL para números, cadenas y marcas de fecha y hora.
- `getTimestamp()`: admite los tipos de datos SQL para números, cadenas y marcas de fecha y hora.
- `getCharacterStream()`: admite los tipos de datos SQL para números, cadenas y valores binarios.
- `getUnicodeStream()`: admite los tipos de datos SQL para números, cadenas y valores binarios.
- `getBinaryStream()`: valores binarios.
- `getObject()`: admite todos los tipos de datos SQL.

En todos los casos, el número de columnas debe incluirse como parámetro cuyos valores deben consultarse.

Las variantes ResultSet

La velocidad de acceso a las bases de datos suele ser un problema esencial. StarOffice ofrece varios métodos para optimizar `ResultSets` y controlar así la velocidad de acceso. Cuantas más funciones ofrezca `ResultSet`, más compleja será su implementación y, por tanto, más lentas serán sus funciones.

Un `ResultSet` sencillo, como el que se muestra en la sección "Iteración de tablas", ofrece la cantidad mínima de funciones. Sólo permite aplicar iteración hacia delante y para los valores que se van a interrogar. No se incluye ninguna opción de navegación más amplia, como la posibilidad de modificar valores.

El objeto `Expresion` que se utiliza para crear `ResultSet` ofrece algunas propiedades que permiten influir sobre las funciones de `ResultSet`:

- **`ResultSetConcurrency (const)`**: especifica, por ejemplo, si los datos pueden modificarse o no (especificaciones de acuerdo con `com.sun.star.sdbc.ResultSetConcurrency`).
- **`ResultSetType (const)`**: especifica el tipo de `ResultSets` (especificaciones de acuerdo con `com.sun.star.sdbc.ResultSetType`).

Los valores definidos en `com.sun.star.sdbc.ResultSetConcurrency` son:

- **`UPDATABLE`**: `ResultSet` permite modificar los valores.
- **`READ_ONLY`**: `ResultSet` no permite modificaciones.

El grupo de constantes `com.sun.star.sdbc.ResultSetConcurrency` ofrece las especificaciones siguientes:

- **`FORWARD_ONLY`**: `ResultSet` permite únicamente navegar hacia delante.
- **`SCROLL_INSENSITIVE`**: `ResultSet` permite cualquier tipo de navegación; sin embargo, no se consignan los cambios en los datos originales.
- **`SCROLL_SENSITIVE`**: `ResultSet` permite cualquier tipo de navegación; los cambios en los datos originales influyen en `ResultSet`.

> `ResultSet` con las propiedades `READ_ONLY` y `SCROLL_INSENSITIVE` corresponde a un conjunto de registros de tipo `Snapshot` en ADO y DAO.
>
> Si se utilizan las propiedades de `ResultSet` `UPDATEABLE` y `SCROLL_SENSITIVE`, el ámbito de la función de `ResultSet` es comparable a `Recordset` de tipo `Dynaset` de ADO y DAO.

Métodos de navegación en ResultSet

Si `ResultSet` es del tipo `SCROLL_INSENSITIVE` o `SCROLL_SENSITIVE`, admite una amplia gama de métodos para navegar dentro de los datos. Los principales son:

- **`next()`**: desplazamiento al siguiente registro de datos.

- **`previous()`**: desplazamiento al registro de datos anterior.
- **`first()`**: desplazamiento al primer registro de datos.
- **`last()`**: desplazamiento al último registro de datos.
- **`beforeFirst()`**: desplazamiento al punto anterior al primer registro de datos.
- **`afterLast()`**: desplazamiento al punto posterior al último registro de datos.

Todos los métodos devuelven un parámetro booleano que especifica si la navegación se ha llevado a cabo satisfactoriamente o no.

Para determinar la posición actual del cursor se ofrecen los siguientes métodos de prueba, todos los cuales devuelven un valor booleano:

- **`isBeforeFirst()`**: `ResultSet` se encuentra en el punto anterior al primer registro de datos.
- **`isAfterLast()`**: `ResultSet` se encuentra en el punto posterior al último registro de datos.
- **`isFirst()`**: `ResultSet` es el primer registro de datos.
- **`isLast()`**: `ResultSet` es el último registro de datos.

Modificación de los registros de datos

Si se ha creado `ResultSet` con el valor `ResultSetConcurrency = UPDATEABLE`, es posible editar su contenido. Esto sólo sucede mientras la orden SQL permita rescribir los datos en la base de datos (depende del principio). Por ejemplo, esto no es posible en el caso de órdenes SQL complejas con columnas vinculadas o valores acumulados.

El objeto `ResultSet` ofrece métodos `Update` para modificar valores, estructurados de la misma forma que los métodos `get` para recuperar valores. El método `updateString`, por ejemplo, permite escribir una cadena.

Después de modificarlos, los valores deben transferirse a la base de datos mediante el método `updateRow()`. La llamada debe hacerse antes de la orden de navegación siguiente o los valores se perderán.

Si hay algún error durante las modificaciones, se puede deshacer mediante el método `cancelRowUpdates()`. Esta llamada sólo está disponible si los datos no se han reescrito en la base de datos mediante `updateRow()`.

CAPÍTULO **11**

Diálogos

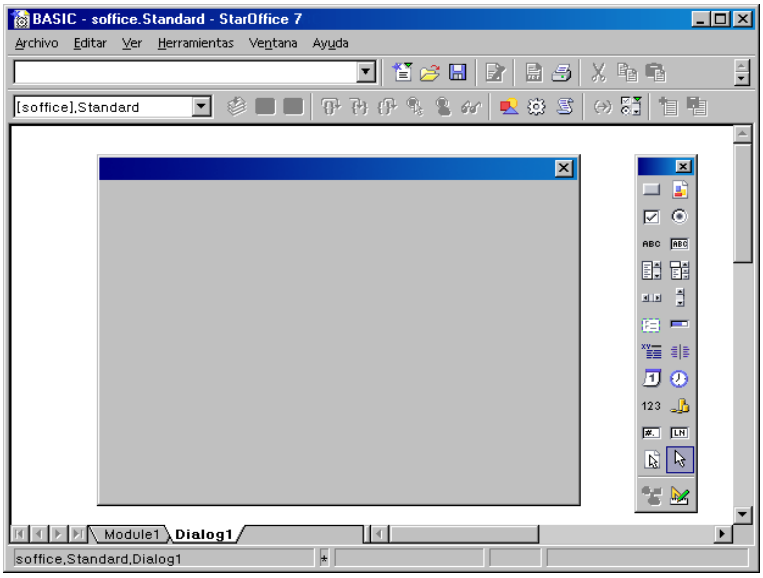
En los documentos de StarOffice se pueden añadir ventanas de diálogo y formularios que pueden vincularse con macros de StarOffice Basic para ampliar significativamente la utilidad de esta aplicación. Por ejemplo, los diálogos pueden mostrar información de la base de datos o guiar, en forma de Piloto automático, a los usuarios en el proceso de crear un nuevo documento paso a paso.

Trabajo con diálogos

Los diálogos de StarOffice Basic constan de una ventana de diálogo que puede contener campos de texto, listados, campos de opción y otros elementos de control.

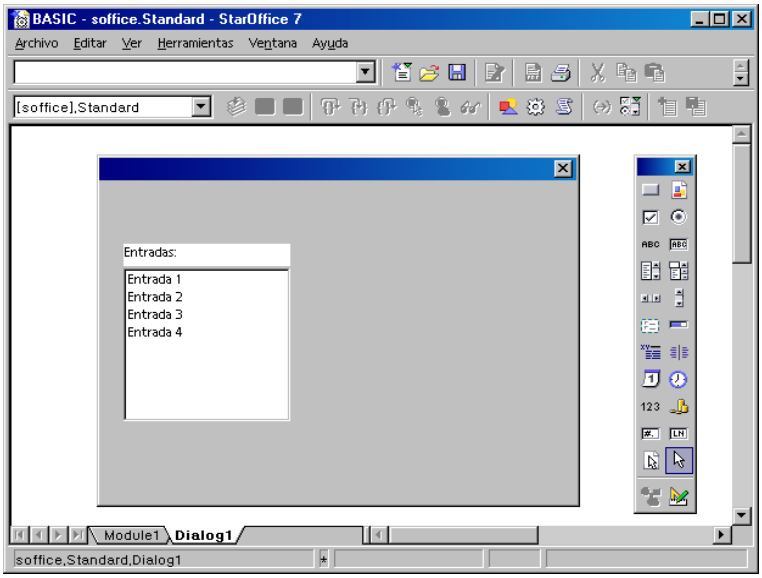
Creación de diálogos

Para crear y estructurar diálogos se puede utilizar el editor de diálogos de StarOffice que se usa de forma similar a StarOffice Draw:



En esencia, se trata de arrastrar los elementos de control deseados desde la paleta de diseño (situada a la derecha) al área del diálogo, en la que se puede definir su posición y tamaño.

En el ejemplo se muestra un diálogo que contiene una etiqueta y un listado.



Para abrir un diálogo se puede utilizar el código siguiente:

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` crea un objeto denominado `Dlg` que hace referencia al diálogo asociado. Antes de poder crear el diálogo, deberá asegurarse de que la biblioteca que utiliza (en este ejemplo, la biblioteca `Standard`) esté cargada. En caso contrario, se debe utilizar el método `LoadLibrary` para hacerlo.

Una vez inicializado el objeto `Dlg`, se puede utilizar el método `Execute` para mostrarlo. Este tipo de diálogos se denominan modales, ya que no permiten efectuar ninguna otra acción en el programa hasta que se cierran. Mientras este diálogo esté abierto, el programa permanecerá dentro de la llamada `Execute`.

El método `dispose` situado al final del código aprueba los recursos utilizados por el diálogo una vez finalizado el programa.

Cierre de diálogos

Cierre con Aceptar o Cancelar

Si un diálogo contiene los botones **Aceptar** o **Cancelar**, se cierra automáticamente al pulsar cualquiera de ellos. Encontrará más información acerca de estos botones en la sección Elementos de control de diálogos al detalle, en este mismo capítulo.

Al cerrar un diálogo mediante el botón **Aceptar** , el método `Execute` devuelve un valor de `1`; en caso contrario se devuelve el valor 0.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
      MsgBox "Se ha pulsado Aceptar"
Case 0
      MsgBox "Se ha pulsado Cancelar"
End Select
```

Cierre con el botón Cerrar de la barra de título

Si lo desea puede cerrar un diálogo pulsando el botón Cerrar de la barra de título de la ventana de diálogo. En este caso, el método `Execute` del diálogo devuelve el valor 0, igual que cuando se pulsa el botón Cancelar.

Cierre con una llamada explícita en el programa

También se puede cerrar un diálogo abierto mediante el método `endExecute`:

```
Dlg.endExecute()
```

Acceso a elementos de control individuales

Un diálogo puede contener un número cualquiera de elementos de control accesibles mediante el método `getControl` que devuelve el nombre del elemento de control.

```
Dim Control As Object

Control = Dlg.getControl("MiBoton")
Control.Label = "Etiqueta nueva"
```

Este fragmento de código determina el objeto correspondiente al elemento de control `MiBoton` y, a continuación, inicializa la variable de objeto `Control` con una referencia al elemento. Finalmente el código define para la propiedad `Label` del elemento de control el valor `Etiqueta nueva`.

Tenga en cuenta que StarOffice Basic distingue entre mayúsculas y minúsculas en los nombres de los elementos de control.

Trabajo con el modelo de diálogos y elementos de control

La división entre elementos de programa visibles (***Ver***) y los datos o documentos subyacentes (***Modelo***) tiene lugar con frecuencia en la API de StarOffice. Además de los métodos y propiedades de los elementos de control, tanto los objetos de diálogo como los de elementos de control tienen un objeto del `modelo` subordinado. Este objeto permite acceder directamente al contenido de un diálogo o elemento de control.

En los diálogos, la distinción entre los datos y su representación visual no es siempre tan clara como en otras áreas de la API de StarOffice. Los elementos de la API están disponibles a través de la visualización (View) y del modelo (Model).

La propiedad `Model` proporciona un acceso controlado por el programa al modelo del diálogo y a los objetos de elementos de control.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

En este ejemplo se desactiva el botón `cmdNext` del diálogo `Dlg` con la ayuda del objeto de modelo de `cmdNtext`.

Propiedades

Nombre y título

Cada elemento de control dispone de su propio nombre que se puede consultar utilizando la siguiente propiedad de modelo:

- **`Model.Name (String)`**: nombre del elemento de control

Se puede especificar el título que aparece en la barra de título de un diálogo mediante la siguiente propiedad de modelo:

- **`Model.Title (String)`**: título del diálogo (sólo aplicable a diálogos).

Posición y tamaño

Se puede consultar la posición y el tamaño de un elemento de control mediante las siguientes propiedades del objeto modelo:

- **`Model.Height (long)`**: altura del elemento de control (en unidades ma)
- **`Model.Width (long)`**: ancho del elemento de control (en unidades ma)
- **`Model.PositionX (long)`**: posición X del elemento de control, desde el borde interior izquierdo del diálogo (en unidades ma)
- **`Model.PositionY (long)`**: posición Y del elemento de control, desde el borde interior superior del diálogo (en unidades ma)

Para garantizar que el aspecto de los diálogos sea independiente de la plataforma, StarOffice utiliza la unidad interna ***Map AppFont (ma)*** para especificar la posición y el tamaño dentro de los diálogos. Una unidad ma se define como un octavo de la altura promedio de un carácter de la fuente del sistema definida en el sistema operativo y un cuarto de su ancho. Con el uso de las unidades ma, StarOffice garantiza que el aspecto de un diálogo sea el mismo en distintos sistemas con configuraciones diferentes.

Si desea cambiar el tamaño o la posición de los elementos de control para el tiempo de ejecución, determine el tamaño total del diálogo y ajuste los valores para los elementos de control según las proporciones correspondientes.

Map AppFont (ma) sustituye a la unidad Twips para mejorar la independencia de la plataforma.

Foco y orden de tabulación

En cualquier diálogo se puede navegar dentro de los elementos de control pulsando la tecla Tab. En este contexto están disponibles las siguientes propiedades en el modelo de elementos de control:

- **`Model.Enabled (Booleano)`**: activa el elemento de control
- **`Model.Tabstop (booleano)`**: permite llegar al elemento de control mediante la tecla Tab
- **`Model.TabIndex (Long)`**: posición del elemento de control en el orden de activación

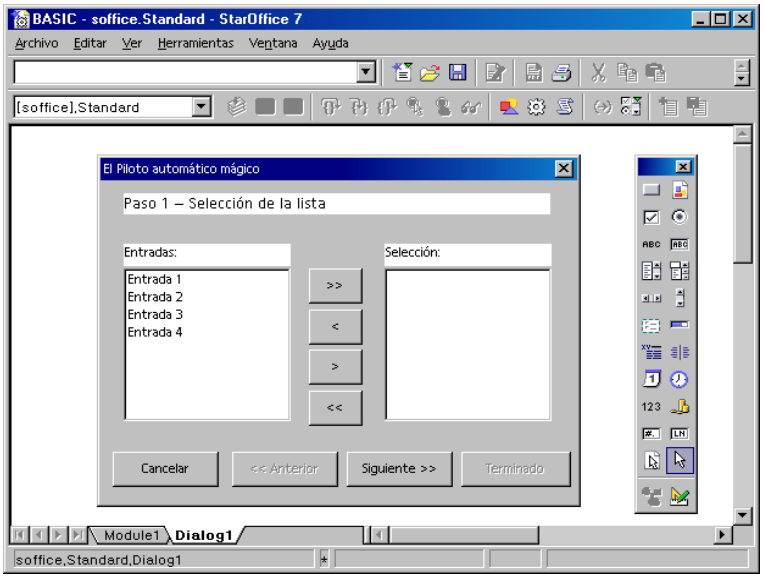
Finalmente, el elemento de control proporciona un método `getFocus` que garantiza que el elemento de control subyacente recibirá el foco:

- **`getFocus`**: el elemento de control recibe el foco (sólo para diálogos)

Diálogos de varias páginas

En StarOffice, un diálogo puede tener más de una ficha. La propiedad `Step` de un diálogo define la ficha actual del mismo, mientras que la propiedad `Step` de un elemento de control especifica la ficha en la que se debe mostrar dicho elemento de control.

Un valor de `Step` de `0` constituye un caso especial. Si define este valor como cero en un diálogo, todos los elementos de control serán visibles independientemente de su valor de `Step`. De forma similar, si define este valor como cero en un elemento de control, dicho elemento se mostrará en todas las fichas de un diálogo.



En este ejemplo se puede asignar un valor de `Step` de `0` a la línea de división, así como a los botones `Cancelar`, `Anterior`, `Siguiente` y `Terminado` para que éstos se muestren en todas las fichas. También se pueden asignar los elementos a una ficha individual (por ejemplo, la ficha `1`).

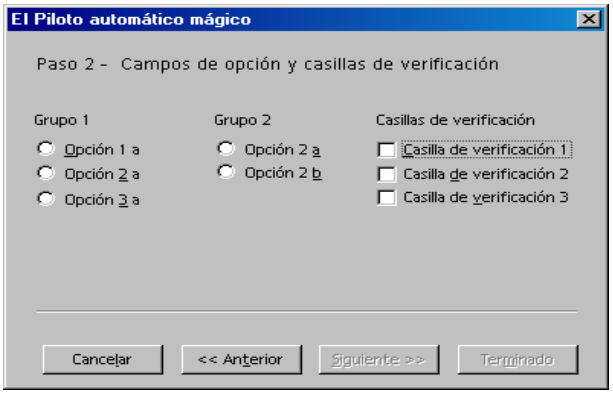
En el siguiente código de programa se muestra de qué forma se puede aumentar el valor de `Step` en los manejadores de acontecimientos de los botones `Siguiente` y `Anterior` para cambiar el estado de éstos.

```
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
```

Se debe incluir una variable `Dlg` global que haga referencia a un diálogo abierto para que este ejemplo sea posible. El aspecto del diálogo cambia de la siguiente forma:

Ficha 1:

Ficha 2:



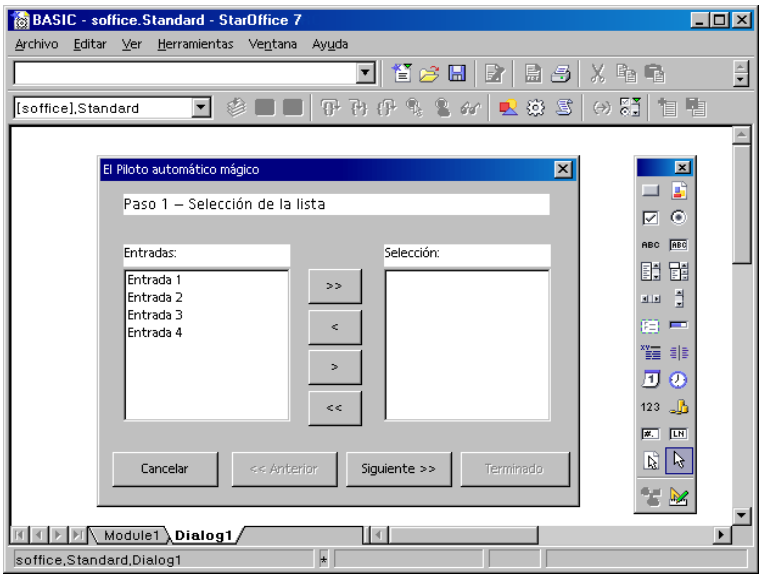
Acciones

Los diálogos y formularios de StarOffice se basan en un modelo de programación orientado a acciones, en el que se pueden asignar ***manejadores de acción*** a los elementos de control. Un manejador de acción ejecuta un procedimiento predefinido cuando tiene lugar una acción específica, incluso si ésta es otra acción. Mediante manejo de acciones se pueden también editar documentos o abrir bases de datos, así como acceder a otros elementos de control.

Los elementos de control de StarOffice reconocen distintos tipos de acciones que se pueden activar en diferentes situaciones. Dichos tipos de acciones se pueden dividir en cuatro grupos:

- **Control del ratón**: acciones que corresponden a actuaciones del ratón (por ejemplo, simples movimientos del ratón o pulsaciones en ubicaciones específicas de la pantalla)
- **Control del teclado**: acciones activadas por pulsaciones de teclas
- **Modificación de foco**: acciones efectuadas por StarOffice al activar o desactivar elementos de control
- **Acciones específicas de elementos de control**: acciones que sólo ocurren en relación con ciertos elementos de control

Al trabajar con acciones, compruebe que ha creado el diálogo asociado en el entorno de desarrollo de StarOffice y que contiene los elementos de control o documentos necesarios (si las acciones corresponden a un formulario).



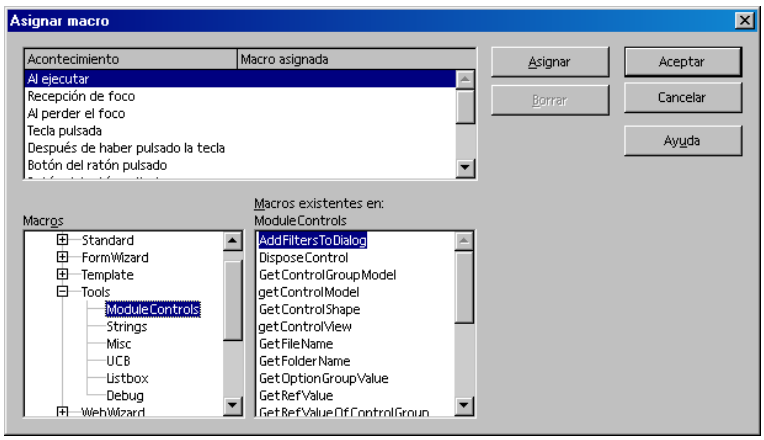
En la figura anterior se muestra el entorno de desarrollo de StarOffice Basic con una ventana de diálogo que contiene dos listados. Los datos se pueden mover de un listado al otro mediante los botones situados entre ambos.

Si desea mostrar el diseño en pantalla deberá crear los procedimientos asociados de StarOffice Basic para que los manejadores de acciones puedan llamarlos. Aunque estos procedimientos pueden utilizarse en cualquier módulo, lo mejor es limitar su uso a dos módulos. Para facilitar la lectura del código, es conveniente asignar a estos procedimientos nombres significativos. El hecho de saltar directamente a un procedimiento de programa general desde una macro puede hacer que el código sea confuso. Para simplificar el mantenimiento del código y la solución de problemas, es conveniente crear otro procedimiento que actúe como punto de entrada para la gestión de acciones, aunque conste únicamente de una llamada al procedimiento de destino.

El código del siguiente ejemplo mueve una entrada del listado izquierdo al derecho de un diálogo.

```
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
```

Si este procedimiento se ha creado en StarOffice Basic, se puede asignar a una acción requerida mediante la ventana de propiedades del editor de diálogos.



En el diálogo de asignación se listan todos los procedimientos de StarOffice Basic. Para asignar un procedimiento a una acción, seleccione el procedimiento y pulse **Asignar**.

Parámetros

La aparición de una acción específica no siempre basta para que la respuesta sea apropiada. Es posible que se necesite información adicional. Por ejemplo, para procesar una pulsación de ratón puede ser necesaria la posición de la pantalla en la que se ha pulsado.

En StarOffice Basic se pueden utilizar parámetros de objeto para proporcionar información adicional acerca de una acción o procedimiento; por ejemplo:

```
Sub ProcessEvent(Event As Object)

End Sub
```

La precisión con la que se haya estructurado el objeto `Event` y las propiedades de éste dependen del tipo de acción activada por la llamada de procedimiento. En las secciones siguientes se describen en detalle distintos tipos de acciones.

Independientemente del tipo de acción, todos los objetos ofrecen acceso al elemento de control relevante y su modelo. Se puede acceder al elemento de control mediante

```
Event.Source
```

y a su modelo mediante

```
Event.Source.Model
```

Estas propiedades se pueden emplear para activar una acción dentro de un manejador de acciones.

Acciones del ratón

StarOffice Basic reconoce las siguientes acciones del ratón:

- **`Ratón movido`**: el usuario ha movido el ratón
- **`Ratón movido con la tecla pulsada`**: el usuario arrastra el ratón al tiempo que mantiene pulsada una tecla

- **Botón del ratón pulsado**: el usuario pulsa un botón del ratón
- **Botón del ratón soltado**: el usuario suelta un botón del ratón
- **Ratón fuera**: el usuario mueve el ratón fuera de la ventana actual

La estructura de los objetos de acción asociados se define en la estructura `com.sun.star.awt.MouseEvent` que proporciona la información siguiente:

- **Buttons (short)**: botón pulsado (una o más constantes de acuerdo con `com.sun.star.awt.MouseButton`).
- **X (long)**: coordenada X del ratón, medida en píxeles a partir de la esquina superior izquierda del elemento de control
- **Y (long)**: coordenada Y del ratón, medida en píxeles a partir de la esquina superior izquierda del elemento de control
- **ClickCount (long)**: número de pulsaciones asociadas con el acción de ratón (si StarOffice puede responder con la suficiente velocidad, ClickCount es también 1 para una pulsación simple, porque sólo se inicia un ación individual).

Las constantes definidas en `com.sun.star.awt.MouseButton` para los botones del ratón son:

- **LEFT**: botón izquierdo del ratón
- **RIGHT**: botón derecho del ratón
- **MIDDLE**: botón central del ratón

En el ejemplo siguiente se muestra en pantalla la posición del ratón, así como el botón pulsado:

```
Sub MouseUp(Event As Object)
      Dim Mensaje As String
      Mensaje = "Botones: "
      If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
            Mensaje = Mensaje & "IZQUIERDO "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
            Mensaje = Mensaje & "DERECHO "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
            Mensaje = Mensaje & "CENTRAL "
      End If
      Mensaje = Mensaje & Chr(13) & "Posición: "
      Mensaje = Mensaje & Event.X & "/" & Event.Y
      MsgBox Mensaje
End Sub
```

Las acciones de VBA `Click` y `Doubleclick` no están disponibles en StarOffice Basic. En su lugar utilice la acción de StarOffice Basic `MouseUp` para la acción `Click` e imite la acción `Doubleclick` cambiando la lógica de la aplicación.

Acciones de teclado

En StarOffice Basic dispone de las siguientes acciones de teclado:

- **`Tecla pulsada`**: el usuario pulsa una tecla
- **`Después de haber pulsado la tecla`**: el usuario suelta una tecla

Ambas acciones están relacionados con acciones de teclas ***lógicas*** y no con acciones ***físicas***. Si el usuario pulsa varias teclas para mostrar un único carácter (por ejemplo, para agregar un acento a un carácter), StarOffice Basic crea una única acción.

Una única acción de tecla en una tecla de modificación, como Mayús o Alt, no crea una acción independiente.

La información acerca de teclas pulsadas la proporciona el objeto de acción que StarOffice Basic proporciona al procedimiento para la gestión de acciones. Contiene las propiedades siguientes:

- **`KeyCode (short)`**: código de la tecla pulsada (valores predeterminados de acuerdo con `com.sun.star.awt.Key`)
- **`KeyChar (String)`**: carácter que se ha introducido (teniendo en cuenta las teclas de modificación)

En el ejemplo siguiente se utiliza la propiedad `KeyCode` para establecer si se ha pulsado la tecla Intro, la tecla Tab o alguna de las otras teclas de control. Si se ha pulsado una de estas teclas se devolverá el nombre de ésta; en caso contrario se devolverá el carácter introducido:

```
Sub TeclaPulsada(Event As Object)
      Dim Mensaje As String
      Select Case Event.KeyCode
      Case com.sun.star.awt.Key.RETURN
            Mensaje = "Se ha pulsado Intro"
      Case com.sun.star.awt.Key.TAB
            Mensaje = "Se ha pulsado Tab"
      Case com.sun.star.awt.Key.DELETE
            Mensaje = "Se ha pulsado Supr"
      Case com.sun.star.awt.Key.ESCAPE
            Mensaje = "Se ha pulsado Escape"
      Case com.sun.star.awt.Key.DOWN
            Mensaje = "Se ha pulsado Flecha abajo"
      Case com.sun.star.awt.Key.UP
            Mensaje = "Se ha pulsado Flecha arriba"
      Case com.sun.star.awt.Key.LEFT
            Mensaje = "Se ha pulsado Flecha izquierda"
      Case com.sun.star.awt.Key.RIGHT
            Mensaje = "Se ha pulsado flecha derecha"
      Case Else
            Mensaje = "Se ha introducido el carácter " & Event.KeyChar
      End Select
      MsgBox Mensaje
End Sub
```

Encontrará información acerca de otras constantes de teclado en la referencia de la API, en el grupo de constantes `com.sun.star.awt.Key`.

Acciones de foco

Las acciones de foco indican si un elemento de control recibe o pierde el foco; se pueden utilizar, por ejemplo, para determinar si un usuario ha acabado de procesar un elemento de control y así poder actualizar otros elementos de un diálogo. Las acciones disponibles de foco son:

- **`Recepción de foco`**: el elemento recibe el foco
- **`Pérdida de foco`**: el elemento pierde el foco

Los objetos `Event` para las acciones de foco están estructurados de la siguiente forma:

- **`FocusFlags (short)`**: causa del cambio de foco (valor predeterminado de acuerdo con `com.sun.star.awt.FocusChangeReason`).
- **`NextFocus (Object)`**: objeto que recibe el foco (sólo para la acción `Pérdida de foco`)
- **`Temporary (booleano)`**: el foco se ha perdido temporalmente

Acciones específicas de elementos de control

Aparte de las acciones indicadas anteriormente, que admiten todos los elementos de control, también existen acciones específicas de ciertos elementos de control, que sólo están definidas para éstos. Las acciones más importantes dentro de este grupo son:

- **`Elemento modificado`**: el valor de un elemento de control ha cambiado
- **`Estado modificado`**: el estado de un elemento de control ha cambiado
- **`Texto modificado`**: el texto de un elemento de control ha cambiado
- **`Al ejecutar`**: una acción que se puede efectuar al activar el elemento de control (por ejemplo, al pulsar un botón)

Al trabajar con acciones tenga en cuenta que algunas, como `Al ejecutar`, se pueden ejecutar cada vez que se pulsa el ratón sobre ciertos elementos de control (por ejemplo, campos de opción). No se efectúa ninguna acción para comprobar si el estado del elemento de control ha cambiado realmente. Para evitar estas •gacciones ciegas•h guarde el valor del elemento de control en una variable global y luego compruebe si el valor ha cambiado cuando se esté ejecutando una acción.

Las propiedades de la acción `Estado modificado` son:

- **`Selected (long)`**: entrada actualmente seleccionada
- **`Highlighted (long)`**: entrada actualmente resaltada
- **`ItemId (long)`**: ID de la entrada

Elementos de control de diálogos

StarOffice Basic reconoce diversos elementos de control que pueden dividirse en los grupos siguientes:

Campos de entrada:

- Campos de texto
- Campos de fecha
- Campos de hora
- Campos numéricos
- Campos de moneda
- Campos que adoptan cualquier formato

Botones:

- Botones estándar
- Casillas de verificación
- Botones de opción

Listas de selección:

- Listados
- Cuadros combinados

Otros elementos de control:

- Barras de desplazamiento (horizontales y verticales)
- Campos de grupos
- Barras de progresión
- Líneas de división (horizontales y verticales)
- Imágenes
- Campos de selección de archivos

A continuación se presentan los elementos de control más importantes de entre los citados.

Botones

Un botón efectúa una acción al pulsarlo.

El caso más sencillo es que el botón active una acción `Al ejecutar` cuando un usuario lo pulsa. También se puede vincular otra acción al botón para que abra un diálogo empleando la propiedad `PushButtonType`. Al pulsar un botón en el que esta propiedad tiene el valor 0, el diálogo no queda afectado. Al pulsar un botón en el que esta propiedad tiene el valor 1, el diálogo se cierra y el método `Execute` del diálogo devuelve el valor 1 (la secuencia de diálogos ha finalizado correctamente). Si `PushButtonType` tiene el valor 2, el diálogo se cierra y el método `Execute` del diálogo devuelve el valor 0 (diálogo cerrado).

A continuación se enumeran las propiedades disponibles a través del modelo de botón:

- **`Model.BackgroundColor (long)`**: color de fondo
- **`Model.DefaultButton (booleano)`**: el botón se utiliza como valor predeterminado y responde a la tecla Intro si no tiene el foco.

- **Model.FontDescriptor (struct)**: estructura que especifica los detalles de la fuente que se debe utilizar (de acuerdo con la estructura `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)**: etiqueta que se muestra en el botón
- **Model.Printable (booleano)**: el elemento de control se puede imprimir
- **Model.TextColor (Long)**: color de texto del elemento de control
- **Model.HelpText (String)**: texto de ayuda que se muestra al situar el cursor sobre el elemento de control
- **Model.HelpURL (String)**: URL de la ayuda para el elemento de control correspondiente
- **PushButtonType (short)**: acción vinculada al botón (0: sin acción, 1: Aceptar, 2: Cancelar)

Campos de opción

Estos campos de opción se suelen utilizar en grupos y permiten seleccionar una de entre varias opciones. Al seleccionar una de las opciones, el resto de opciones del grupo quedan desactivadas. De esta forma se garantiza que, en un momento determinado, sólo una opción estará activa.

El elemento de control campo de opción ofrece dos propiedades:

- **State (booleano)**: activa el botón
- **Label (String)**: etiqueta que se muestra en el botón

También se pueden utilizar las siguientes propiedades del modelo de los campos de opción:

- **Model.FontDescriptor (struct)**: estructura con detalles de la fuente que se debe utilizar (de acuerdo con `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)**: etiqueta que se muestra en el elemento de control
- **Model.Printable (Booleano)**: el elemento de control se puede imprimir
- **Model.State (Short)**: si el valor de esta propiedad es 1, la opción está activa; en caso contrario estará inactiva
- **Model.TextColor (Long)**: color del texto del elemento de control
- **Model.HelpText (String)**: texto de ayuda que se muestra al situar el cursor sobre el elemento de control
- **Model.HelpURL (String)**: URL de la ayuda en línea para el elemento de control correspondiente

Para combinar varios campos de opción en un grupo se deben situar uno junto al otro en el orden de activación, sin espacios (propiedad `Model.TabIndex`, descrita como Orden en el editor de diálogos). Si el orden de activación queda interrumpido por otro elemento de control StarOffice inicia automáticamente un nuevo grupo de elementos de control que se puede activar de forma independiente del primer grupo.

A diferencia de VBA, no es posible insertar campos de opción en un grupo de elementos de control en StarOffice Basic. La agrupación de elementos de control en StarOffice Basic se utiliza únicamente para garantizar una división visual mediante un marco situado alrededor de los elementos de control.

Casillas de verificación

Las casillas de verificación se utilizan para indicar un valor de Sí o de No y, según el modo, pueden adoptar dos o tres estados. Además de los estados Sí y No, una casilla de verificación puede adoptar un ***estado intermedio*** si el estado Sí o No correspondiente tiene más de un significado o no está del todo claro.

Las casillas de verificación ofrecen las propiedades siguientes:

- **`State (Short)`**: estado de la casilla de verificación (0: no, 1: sí, 2: estado intermedio)
- **`Label (String)`**: etiqueta del elemento de control
- **`enableTriState (booleano)`**: aparte de los estados activado y desactivado, se puede utilizar también el estado intermedio

El objeto modelo de una casilla de verificación ofrece las propiedades siguientes:

- **`Model.FontDescriptor (struct)`**: estructura con detalles de la fuente que se debe utilizar (de acuerdo con `com.sun.star.awt.FontDescriptor`)
- **`Model.Label (String)`**: etiqueta del elemento de control
- **`Model.Printable (booleano)`**: el elemento de control se puede imprimir
- **`Model.State (Short)`**: estado de la casilla de verificación (0: no, 1: sí, 2: estado intermedio)
- **`Model.Tabstop (booleano)`**: permite llegar al elemento de control mediante la tecla Tab
- **`Model.TextColor (Long)`**: color del texto del elemento de control
- **`Model.HelpText (String)`**: texto de ayuda que se muestra al situar el cursor sobre el elemento de control
- **`Model.HelpURL (String)`**: URL de la ayuda en línea para el elemento de control correspondiente

Campos de texto

Los campos de texto permiten a los usuarios introducir números y texto. El servicio `com.sun.star.awt.UnoControlEdit` constituye la base de los campos de texto.

Un campo de texto puede contener una o más líneas y puede ser editable o estar bloqueado para entradas del usuario. Los campos de texto se pueden también utilizar como campos especiales de moneda o numéricos, así como campos de pantalla para tareas especiales. Estos elementos de control se basan en el servicio de Uno `UnoControlEdit`, por lo que su manejo en programas es similar.

Los campos de texto ofrecen las propiedades siguientes:

- **`Text (String)`**: texto actual
- **`SelectedText (String)`**: texto actualmente resaltado

- **Selection (Struct)**: resaltado de sólo lectura de detalles (estructura de acuerdo con com.sun.star.awt.Selection, con las propiedades Min y Max para especificar el inicio y el final del resaltado actual)
- **MaxTextLen (short)**: número máximo de caracteres que se pueden introducir en el campo
- **Editable (booleano)**: True activa la opción de introducir texto, False bloquea la opción de entrada de texto (la propiedad no se puede llamar directamente, sino sólo a través de IsEditable)
- **IsEditable (booleano)**: el contenido del elemento de control se puede modificar, sólo lectura.

Adicionalmente, a través del objeto modelo asociado se puede acceder a estas propiedades:

- **Model.Align (short)**: orientación del texto (0: alineado a la izquierda, 1: centrado, 2: alineado a la derecha)
- **Model.BackgroundColor (long)**: color de fondo del elemento de control
- **Model.Border (short)**: tipo de borde (0: sin borde, 1: borde 3D, 2: borde simple)
- **Model.EchoChar (String)**: carácter mostrado en pantalla en campos de contraseña
- **Model.FontDescriptor (struct)**: estructura que contiene detalles de la fuente utilizada (de acuerdo con la estructura com.sun.star.awt.FontDescriptor)
- **Model.HardLineBreaks (Booleano)**: se insertan saltos de línea automáticos de forma permanente en el texto del elemento de control
- **Model.HScroll (booleano)**:
- **Model.MaxTextLen (Short)**: longitud máxima del texto; 0 significa sin límite de longitud
- **Model.MultiLine (booleano)**: permite que la entrada abarque varias líneas
- **Model.Printable (booleano)**: el elemento de control se puede imprimir
- **Model.ReadOnly (booleano)**: el contenido del elemento de control es de sólo lectura.
- **Model.Tabstop (booleano)**: permite llegar al elemento de control mediante la tecla Tab
- **Model.Text (String)**: texto asociado con el elemento de control
- **Model.TextColor (Long)**: color del texto del elemento de control
- **Model.VScroll (booleano)**: el texto tiene asociada una barra de desplazamiento vertical
- **Model.HelpText (String)**: texto de ayuda que se muestra al situar el cursor del ratón sobre el elemento de control
- **Model.HelpURL (String)**: URL de la ayuda en línea para el elemento de control

Listados

Los listados (servicio com.sun.star.awt.UnoControlListBox) admiten las propiedades siguientes:

- **ItemCount (Short)**: número de elementos, sólo lectura
- **SelectedItem (String)**: texto de la entrada resaltada, sólo lectura
- **SelectedItems (matriz de Strings)**: campo de datos con entradas resaltadas, sólo lectura
- **SelectedItemPos (Short)**: número de la entrada actualmente resaltada, sólo lectura
- **SelectedItemsPos (matriz de Short)**: campo de datos con el número de entradas resaltadas (para listas que admiten selección múltiple), sólo lectura
- **MultipleMode (booleano)**: `True` activa la opción de seleccionar varias entradas, `False` bloquea la opción de selección múltiple (la propiedad no se puede llamar directamente, sino únicamente a través de `IsMultipleMode`)
- **IsMultipleMode (booleano)**: permite la selección múltiple dentro de listas, sólo lectura

Los listados ofrecen los métodos siguientes:

- **addItem (Item, Pos)**: inserta la cadena especificada en el parámetro `Item` en la posición `Pos` de la lista
- **addItems (ItemArray, Pos)**: inserta las cadenas especificadas en el campo de datos `ItemArray` en la posición `Pos` de la lista
- **removeItems (Pos, Count)**: suprime `Count` entradas a partir de la posición `Pos`
- **selectItem (Item, SelectMode)**: activa o desactiva el resaltado del elemento especificado en la cadena `Item` en función de la variable booleana `SelectMode`
- **makeVisible (Pos)**: se desplaza en el listado hasta que la entrada especificada en `Pos` sea visible

El objeto modelo de los listados ofrece las propiedades siguientes:

- **Model.BackgroundColor (long)**: color de fondo del elemento de control
- **Model.Border (short)**: tipo de borde (0: sin borde, 1: borde 3D, 2: borde simple)
- **Model.FontDescriptor (struct)**: estructura que contiene detalles de la fuente utilizada (de acuerdo con la estructura `com.sun.star.awt.FontDescriptor`)
- **Model.LineCount (Short)**: número de líneas del elemento de control
- **Model.MultiSelection (booleano)**: permite seleccionar varias entradas
- **Model.SelectedItems (matriz de Strings)**: lista de entradas resaltadas
- **Model.StringItemList (matriz de Strings)**: lista de todas las entradas
- **Model.Printable (booleano)**: el elemento de control se puede imprimir
- **Model.ReadOnly (booleano)**: el contenido del elemento de control es de sólo lectura.
- **Model.Tabstop (booleano)**: permite llegar al elemento de control mediante la tecla Tab.
- **Model.TextColor (Long)**: color del texto del elemento de control

- **`Model.HelpText (String)`**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control
- **`Model.HelpURL (String)`**: URL de la ayuda en línea para el elemento de control correspondiente

La opción de VBA para emitir entradas de listado con un valor numérico adicional (`ItemData`) no existe en StarOffice Basic. Si desea administrar un valor numérico (por ejemplo, un ID de base de datos) aparte del texto de lenguaje natural, deberá crear un campo de datos auxiliar que se administre de forma paralela al listado.

CAPÍTULO **12**

Formularios

En muchos sentidos, la estructura de los formularios de StarOffice se corresponde con los diálogos comentados en el capítulo anterior. Sin embargo, hay algunas diferencias fundamentales:

- Los diálogos aparecen como una única ventana de diálogo que se muestra sobre el documento y que no permite efectuar ninguna acción que no esté relacionada con el proceso del propio diálogo hasta finalizar éste. En cambio, los formularios se muestran directamente en el documento como si fuesen elementos de dibujo.
- El entorno de desarrollo de StarOffice Basic contiene un editor de diálogos para la creación de éstos. Los formularios se crean directamente en el documento mediante la barra de herramientas **Funciones del formulario**.
- Mientras que las funciones de diálogos están disponibles en todos los documentos de StarOffice, la totalidad de las funciones de formulario sólo está disponible en documentos de texto y hojas de cálculo.
- Los elementos de control de un formulario se pueden vincular con una tabla de base de datos externa. Esta función no está disponible en diálogos.
- Los elementos de control de los diálogos y de los formularios difieren en diversos aspectos.

Los usuarios que deseen incorporar en sus formularios sus propios métodos de gestión de acciones deberán consultar el capítulo 11 (***Diálogos***). Los mecanismos que se explican en él son idénticos a los correspondientes a formularios.

Trabajo con formularios

Los formularios de StarOffice pueden contener campos de texto, listados, campos de opción y muchos otros elementos de control que se insertan directamente en documentos de texto o en hojas de cálculo. La barra de herramientas **Funciones del formulario** se utiliza para la edición de formularios.

Los formularios de StarOffice pueden adoptar uno de estos modos: el modo de borrador y el modo de visualización. En modo de borrador, la posición de los elementos de control se puede modificar y sus propiedades se pueden editar mediante una ventana de propiedades.

La **barra de herramientas Funciones del formulario** se utiliza también para cambiar de un modo a otro.

Determinar objetos de formulario

StarOffice sitúa los elementos de control de un formulario en el nivel de objetos de dibujo. Se puede acceder al formulario objeto en sí a través de la lista `Formularios` en el nivel de dibujo. En los documentos de texto se accede a los objetos de la siguiente forma:

```
Dim Doc As Object
Dim PaginaDibujo As Object
Dim Formulario As Object

Doc = StarDesktop.CurrentComponent
PaginaDibujo = Doc.DrawPage
Formulario = PaginaDibujo.Forms.GetByIndex(0)
```

Elmétodo `GetByIndex` devuelve el formulario con el número de índice 0.

Al trabajar con hojas de cálculo se debe pasar por la etapa intermedia de la lista `Hojas`, ya que los niveles de dibujo no se encuentran directamente en el documento, sino en las hojas individuales:

```
Dim Doc As Object
Dim Hoja As Object
Dim PaginaDibujo As Object
Dim Formulario As Object

Doc = StarDesktop.CurrentComponent
Hoja = Doc.Sheets.GetByIndex(0)
PaginaDibujo = Hoja.DrawPage
Formulario = PaginaDibujo.Forms.GetByIndex(0)
```

Como el propio nombre del método `GetByIndex` sugiere, un documento puede contener varios formularios. Esta característica es útil, por ejemplo, si en un solo documento se muestra el contenido de distintas bases de datos o si en un formulario se muestra una relación de base de datos 1:n. La opción de crear subformularios es, asimismo, útil en este sentido.

Aspectos de un elemento de control de un formulario

Un elemento de control de un formulario tiene tres aspectos:

- En primer lugar está el ***Modelo*** del elemento de control. Se trata del objeto clave para el programador de StarOffice Basic en el trabajo con elementos de control en formularios.
- La contrapartida de éste es la ***Visualización*** del elemento de control, que administra la información de presentación.
- Puesto que los elementos de control de los documentos se administran como elementos de dibujo especiales, también existe un objeto ***Shape*** (forma) que refleja las propiedades específicas de elemento de dibujo del elemento de control (en particular su posición y su tamaño).

Acceso al modelo de los elementos de control de formulario

Los modeles de los elementos de control de un formulario están disponibles mediante el método `GetByName` del `objeto formulario`:

```
Dim Doc As Object
Dim Formulario As Object
Dim Control As Object

Doc = StarDesktop.CurrentComponent
Formulario = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Formulario.getByName("MiListado")
```

En el ejemplo se determina el modelo del elemento de control `MiListado`, que se encuentra en el primer formulario del documento de texto actualmente abierto.

Si no está seguro del formulario en el que se encuentra el elemento de control requerido puede optar por buscarlo en todos los formularios:

```
Dim Doc As Object
Dim Formularios As Object
Dim Formulario As Object
Dim Control As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Formularios = Doc.Drawpage.Forms

For I = 0 To Formularios.Count - 1
      Formulario = Formularios.GetbyIndex(I)
      If Formulario.HasByName("MiListado") Then
            Ctl = Formulario.GetbyName("MiListado")
            Exit Function
      End If
Next I
```

En el ejemplo se utiliza el método `HasByName` para comprobar todos los formularios de un documento de texto y determinar si contiene un modelo de elemento de control denominado `MiListado`. Si se encuentra un modelo que se corresponda con éste se guarda una referencia al mismo en la variable `Ctl` y finaliza la búsqueda.

Acceso a la visualización de los elementos de control de formulario

Para acceder a la visualización de un elemento de control de un formulario es necesario en primer lugar el modelo asociado. La visualización del elemento de control se puede entonces determinar con la ayuda del modelo y utilizando el controlador de documento.

```
Dim Doc As Object
Dim DocCrl As Object
```

```
Dim Formularios As Object
Dim Formulario As Object
Dim Control As Object
Dim VisCtl As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
DocCrl = Doc.getCurrentControler()
Formularios = Doc.Drawpage.Forms

For I = 0 To Formularios.Count - 1
      Formulario = Formularios.GetbyIndex(I)
      If Formulario.HasByName("MiListado") Then
            Ctl = Formulario.GetbyName("MiListado")
            VisCtl = DocCrl.GetControl(Ctl)
            Exit Function
      End If
Next I
```

El código del ejemplo es muy similar al del ejemplo anterior en el que se determinaba el modelo de un elemento de control. No sólo utiliza el objeto documento `Doc`, sino también el objeto controlador de documento `DocCrl` que hace referencia a la ventana de documento actual. Con la ayuda de este objeto controlador y del modelo del elemento de control, utiliza el método `GetControl` para determinar la visualización (variable `CtlView`) del elemento de control del formulario.

Acceso al objeto Shape de los elementos de control de formulario

El método para acceder al objeto Shape de un elemento de control utiliza también el nivel de dibujo correspondiente del documento. Para determinar un elemento de control especial se debe buscar en todos los elementos de dibujo del nivel de dibujo.

```
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 = "MiListado" Then
                        Exit Function
            End If
      End If
Next
```

En el ejemplo se comprueban todos los elementos de dibujo para determinar si admiten la interfaz `com.sun.star.drawing.XControlShape` necesaria para los formularios de elementos de control. Si es así, la propiedad `Control.Name` comprueba si el nombre del elemento de control es `MiListado`. En tal caso, la función finaliza la búsqueda.

Determinar el tamaño y la posición de los elementos de control

Como ya se ha mencionado, el tamaño y la posición de los elementos de control se puede determinar mediante el objeto `Shape` asociado. El objeto Shape del elemento de control, igual que el resto de objetos `Shape` , contiene las propiedades `Size` y `Position` del mismo:

- **`Size (struct)`**: tamaño del elemento de control (estructura de datos `com.sun.star.awt.Size`).
- **`Position (struct)`**: posición del elemento de control (`estructura de datos` com.sun.star.awt.Point).

En el ejemplo siguiente se muestra de qué forma se pueden establecer la posición y el tamaño de un elemento de control mediante el objeto `Shape` asociado:

```
Dim Shape as Object

Punto.x = 1000
Punto.y = 1000
Tamano.Width = 10000
Tamano.Height = 10000

Shape.Size = Tamano
Shape.Position = Punto
```

El objeto `Shape` del elemento de control debe ser conocido para que el código funcione. Si no es así se deberá determinar mediante el código indicado más arriba.

Elementos de control de formularios en detalle

Los elementos de control disponibles para formularios son similares a los utilizados en los diálogos. La variedad abarca desde campos de texto sencillos a listados y cuadros combinados pasando por diversos tipos de botones.

A continuación se muestra una lista de las propiedades más importantes de los elementos de control de formularios. Todas estas propiedades pertenecen a los objetos modelo asociados.

Además de los elementos de control estándar, en los formularios también está disponible un elemento de control de tabla que permite incorporar tablas de base de datos enteras. Este elemento se describe en la sección ***Formularios de base de datos*** del capítulo 11.

Botones

El objeto modelo de un botón de formulario ofrece las propiedades siguientes:

- **`BackgroundColor (long)`**: color del fondo.

- **DefaultButton (booleano)**: el botón actúa como valor predeterminado. En este caso responde también al botón de entrada si no tiene el foco.
- **Enabled (booleano)**: el elemento de control se puede activar
- **Tabstop (booleano)**: se puede llegar al elemento de control mediante la tecla de tabulador.
- **TabIndex (Long)**: posición del elemento de control en el orden de activación
- **FontName (cadena)**: nombre del tipo de fuente.
- **FontHeight (Single)**: altura del carácter en puntos (pt).
- **Tag (String)**: cadena que contiene información adicional que se puede guardar en el botón para acceso controlado por programa.
- **TargetURL (String)**: URL de destino para los botones del tipo URL.
- **TargetFrame (String)**: nombre de la ventana (o marco) en el que se debe abrir `TargetURL` al activar el botón (para botones de tipo `URL`).
- **Label (String)**: etiqueta de botón.
- **TextColor (Long)**: color del texto del elemento de control
- **HelpText (String)**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control
- **HelpURL (String)**: URL de la ayuda en línea para el elemento de control correspondiente.
- **ButtonType (Enum)**: acción vinculada con el botón (valor predeterminado en `com.sun.star.form.FormButtonType`).

La propiedad `ButtonType` ofrece la oportunidad de definir una acción que se ejecuta automáticamente al pulsar el botón. El grupo de constantes `com.sun.star.form.FormButtonType` asociado ofrece los valores siguientes:

- **PUSH**: botón estándar.
- **SUBMIT**: final de entrada de formulario (especialmente importante en formularios HTML).
- **RESET**: restablece los valores originales del formulario.
- **URL**: llamada de la URL definida en `TargetURL` (se abre dentro de la ventana especificada mediante TargetFrame).

Los tipos de botón **Aceptar** y **Cancelar** de los diálogos no se admiten en formularios.

Campos de opción

El objeto modelo ***de un campo de opción ofrece las siguientes propiedades***:

- **Enabled (booleano)**: el elemento de control se puede activar
- **Tabstop (booleano)**: se puede llegar al elemento de control mediante la tecla de tabulador.
- **TabIndex (Long)**: posición del elemento de control en el orden de activación

- **FontName (cadena)**: nombre del tipo de fuente.
- **FontHeight (Single)**: altura del carácter en puntos (pt).
- **Tag (String)**: cadena que contiene información adicional que se puede guardar en el botón para acceso controlado por programa.
- **Label (String)**: etiqueta del botón.
- **Printable (booleano)**: el elemento de control se puede imprimir.
- **State (Short)**: si es 1, se activa la opción; en caso contrario, se desactiva.
- **RefValue (String)**: cadena para guardar información adicional (por ejemplo, para administrar ID de registros de datos).
- **TextColor (Long)**: color del texto del elemento de control
- **HelpText (String)**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control.
- **HelpURL (String)**: URL de la ayuda en línea para el elemento de control correspondiente.

El mecanismo para agrupar campos de opción distingue entre los elementos de control de diálogos y formularios. Mientras que los elementos de control que aparecen uno a continuación del otro en los diálogos se combinan automáticamente para formar un grupo, en los formularios la acción de agrupar se efectúa por nombres. Para ello, todos los campos de opción de un grupo deben contener el mismo nombre. StarOffice combina los elementos de control agrupados en un matriz de forma que se pueda acceder a los botones individuales de un programa de StarOffice Basic de la misma forma que antes.

En el ejemplo siguiente se muestra cómo determinar el modelo de un grupo de elementos de control.

```
Dim Doc As Object
Dim Formularios As Object
Dim Formulario As Object
Dim Control As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent
Formularios = Doc.Drawpage.Forms

For I = 0 To Formularios.Count - 1
      Formulario = Formularios.GetbyIndex(I)
      If Formulario.HasByName("MisOpciones") Then
            Ctl = Formulario. GetGroupbyName("MisOpciones")
            Exit Function
      End If
Next I
```

El código corresponde al ejemplo anterior para determinar un modelo de un elemento de control simple. Busca en todos los formularios del documento de texto actual mediante un bucle y utiliza el método `HasByName` para comprobar si el formulario correspondiente contiene un elemento con

el nombre buscado, `MisOpciones`. En tal caso se accede a la matriz de modelos mediante el método `GetGroupByName` (en lugar del método `GetByName`, para determinar modelos simples).

Casillas de verificación

El objeto modelo ***de una casilla de verificación ofrece las propiedades siguientes:***

- **`Enabled (booleano)`**: el elemento de control se puede activar
- **`Tabstop (booleano)`**: se puede llegar al elemento de control mediante la tecla de tabulador.
- **`TabIndex (Long)`**: posición del elemento de control en el orden de activación
- **`FontName (cadena)`**: nombre del tipo de fuente.
- **`FontHeight (Single)`**: altura del carácter en puntos (pt).
- **`Tag (String)`**: cadena que contiene información adicional que se puede guardar en el botón para acceso controlado por programa.
- **`Label (String)`**: etiqueta de botón.
- **`Printable (booleano)`**: el elemento de control se puede imprimir.
- **`State (Short)`**: si es 1, se activa la opción; en caso contrario, se desactiva.
- **`RefValue (String)`**: cadena para guardar información adicional (por ejemplo, para administrar ID de registros de datos).
- **`TextColor (Long)`**: color del texto del elemento de control
- **`HelpText (String)`**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control.
- **`HelpURL (String)`**: URL de la ayuda en línea para el elemento de control correspondiente.

Campos de texto

El objeto modelo de un campo de texto de formulario ofrece las propiedades siguientes:

- **`Align (short)`**: orientación del texto (0: alineado a la izquierda, 1: centrado, 2: alineado a la derecha).
- **`BackgroundColor (long)`**: color de fondo del elemento de control.
- **`Border (short)`**: tipo de borde (0: sin borde, 1: borde 3D, 2: borde simple).
- **`EchoChar (String)`**: carácter mostrado en pantalla en campos de contraseña.
- **`FontName (cadena)`**: nombre del tipo de fuente.
- **`FontHeight (Single)`**: altura del carácter en puntos (pt).
- **`HardLineBreaks (Booleano)`**: se insertan saltos de línea automáticos de forma permanente en el texto del elemento de control.
- **`HScroll (booleano)`**: el texto tiene asociada una barra de desplazamiento horizontal.

- **MaxTextLen (Short)**: longitud máxima del texto; 0 significa sin límite de longitud.
- **MultiLine (booleano)**: permite entradas de varias líneas.
- **Printable (booleano)**: el elemento de control se puede imprimir.
- **ReadOnly (booleano)**: el contenido del elemento de control es de sólo lectura.
- **Enabled (booleano)**: el elemento de control se puede activar
- **Tabstop (booleano)**: se puede llegar al elemento de control mediante la tecla de tabulador.
- **TabIndex (Long)**: posición del elemento de control en el orden de activación.
- **FontName (cadena)**: nombre del tipo de fuente.
- **FontHeight (Single)**: altura del carácter en puntos (pt).
- **Text (String)**: texto del elemento de control.
- **TextColor (Long)**: color del texto del elemento de control
- **VScroll (booleano):** el texto tiene asociada una barra de desplazamiento vertical.
- **HelpText (String)**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control.
- **HelpURL (String):** URL de la ayuda en línea para el elemento de control correspondiente.

Listados

El objeto modelo de un listado de formulario ofrece las propiedades siguientes:

- **BackgroundColor (long)**: color de fondo del elemento de control.
- **Border (short)**: tipo de borde (0: sin borde, 1: borde 3D, 2: borde simple).
- **FontDescriptor (struct)**: estructura que contiene detalles de la fuente utilizada (de acuerdo con la estructura `com.sun.star.awt.FontDescriptor`).
- **LineCount (Short)**: número de líneas del elemento de control.
- **MultiSelection (booleano)**: permite seleccionar varias entradas.
- **SelectedItems (matriz de Strings)**: lista de entradas resaltadas.
- **StringItemList (matriz de Strings)**: lista de todas las entradas.
- **ValueItemList (matriz de Variant)**: lista que contiene información adicional para cada entrada (por ejemplo, para administrar ID de registros de datos).
- **Printable (booleano)**: el elemento de control se puede imprimir.
- **ReadOnly (booleano)**: el contenido del elemento de control es de sólo lectura.
- **Enabled (booleano)**: el elemento de control se puede activar
- **Tabstop (booleano)**: se puede llegar al elemento de control mediante la tecla de tabulador.
- **TabIndex (Long)**: posición del elemento de control en el orden de activación
- **FontName (cadena)**: nombre del tipo de fuente.

- **`FontHeight (Single)`**: altura del carácter en puntos (pt).
- **`Tag (String)`**: cadena que contiene información adicional que se puede guardar en el botón para acceso controlado por programa.
- **`TextColor (Long)`**: color del texto del elemento de control
- **`HelpText (String)`**: texto de ayuda que se muestra automáticamente al situar el cursor del ratón sobre el elemento de control.
- **`HelpURL (String)`:** URL de la ayuda en línea para el elemento de control correspondiente.

Mediante la propiedad `ValueItemList`, los listados de formulario ofrecen una contrapartida de la propiedad `ItemData` de VBA para administrar información adicional para entradas individuales del listado.

Además, el objeto de visualización del listado ofrece los métodos siguientes:

- **`addItem (Item, Pos)`**: inserta la cadena especificada en el parámetro `Item` en la posición `Pos` de la lista
- **`addItems (ItemArray, Pos)`**: inserta las cadenas especificadas en el campo de datos ItemArray en la posición Pos de la lista
- **`removeItems (Pos, Count)`**: borra Count entradas a partir de la posición Pos.
- **`selectItem (Item, SelectMode)`**: activa o desactiva el resaltado del elemento especificado en la cadena `Item` en función de la variable SelectMode.
- **`makeVisible (Pos)`**: se desplaza en el listado hasta que la entrada especificada en Pos sea visible.

Formularios de base de datos

Los formularios de StarOffice se pueden vincular directamente a una base de datos. Los formularios creados así ofrecen todas las funciones de una interfaz de base de datos completa sin precisar de un trabajo de programación independiente.

El usuario tiene la opción de desplazarse y buscar en las tablas y consultas seleccionadas, así como de modificar registros de datos e insertar otros nuevos. StarOffice se asegura automáticamente que se recuperen los datos relevantes de la base de datos y de que los cambios efectuados se escriban en ella.

Un formulario de base de datos corresponde básicamente a un formulario estándar de StarOffice. Además de las propiedades estándar se deben configurar en el formulario las siguientes propiedades específicas de bases de datos:

- **`DataSourceName (String)`**: nombre de la fuente de datos (consulte el capítulo 10, ***Acceso a bases de datos***; Acceso a bases de datos; la fuente de datos se debe crear globalmente en StarOffice).
- **`Command (String)`**: nombre de la tabla, consulta u orden Select de SQL con la que se debe crear un vínculo.

- **`CommandType (Const)`**: especifica si `Command` es una tabla, una consulta o una orden SQL (valor procedente de la enumeración `com.sun.star.sdb.CommandType`).

La enumeración `com.sun.star.sdb.CommandType` abarca los siguientes valores:

- **`TABLE`**: tabla
- **`QUERY`**: consulta
- **`COMMAND`**: orden SQL

Los campos de la base de datos se asignan a los elementos de control individuales a través de esta propiedad:

- **`DataField (String)`**: nombre del campo vinculado de la base de datos.

Tablas

El elemento de control tabla es un elemento específico que se ofrece para trabajar con bases de datos. Representa el contenido completo de una tabla o consulta de base de datos. En el caso más sencillo, un elemento de control de tabla se vincula a una base de datos mediante el formulario del Piloto automático, que vincula todas las columnas con los campos pertinentes de la base de datos según las especificaciones del usuario. La API asociada es relativamente compleja, por lo que no vamos a ofrecer aquí una descripción completa de la misma.

Apéndice

Consejos de migración para VBA

Consejos de migración para StarOffice 5.x

Índice

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y