Guide de programmation de StarOffice 8 pour BASIC

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

Référence : 819–1328–10
Juin 2005

Copyright 2005 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, CA 95054 U.S.A. Tous droits réservés.

050323@11223

Table des matières

CHAPITRE 1

Introduction

Ce guide est une introduction à la programmation avec StarOffice™ 8 Basic et décrit les applications fournies par StarOffice Basic dans StarOffice. Pour tirer le meilleur parti de cet ouvrage, vous devez connaître d'autres langages de programmation.

Des exemples détaillés vous sont proposés pour vous aider à développer rapidement vos propres programmes StarOffice Basic.

Organisation du manuel

Les trois premiers chapitres présentent StarOffice Basic :

- Chapitre 2, Langage de StarOffice Basic
- Chapitre 3, Bibliothèque d'exécution de StarOffice Basic
- Chapitre 4, Introduction à l'API StarOffice

Ces chapitres proposent un aperçu de StarOffice Basic et leur lecture est conseillée à toute personne souhaitant écrire des programmes StarOffice Basic.

Les autres chapitres décrivent plus en détail les différents composants de l'API StarOffice et peuvent être lus séparément selon vos besoins :

- Chapitre 5, Utilisation de documents StarOffice
- Chapitre 6, Documents texte
- Chapitre 7, Classeurs
- Chapitre 8, Dessins et présentations
- Chapitre 9, Diagrammes
- Chapitre 10, Accès aux bases de données
- Chapitre 11, Boîtes de dialogue
- Chapitre 12, Formulaires

À propos de StarOffice Basic

Le langage de programmation StarOffice Basic a été développé spécialement pour StarOffice et est étroitement intégré au package Office.

Comme son nom l'indique, StarOffice Basic est un langage de programmation appartenant à la famille Basic. Les utilisateurs ayant déjà travaillé avec d'autres langages Basic, en particulier Visual Basic ou Visual Basic pour Applications (VBA) de Microsoft, se familiariseront rapidement avec StarOffice Basic. Une grande partie des structures de base de StarOffice Basic sont compatibles avec Visual Basic.

Le langage de programmation StarOffice Basic peut être divisé en quatre composants :

- **Le langage de StarOffice Basic**, qui définit les structures linguistiques élémentaires, par exemple, pour les déclarations de variables, les boucles et les fonctions.
- **La bibliothèque d'exécution**, qui fournit des fonctions standard n'ayant pas de référence directe avec StarOffice, par exemple, des fonctions d'édition de nombres, de chaînes, de dates et de fichiers.
- **L'API (Application Programming Interface) StarOffice**, qui permet d'accéder aux documents StarOffice afin de les créer, de les enregistrer, de les modifier et de les imprimer.
- **L'éditeur de boîte de dialogue**, qui permet de créer des boîtes de dialogue personnelles et d'ajouter des éléments de contrôle ainsi que des gestionnaires d'événements.

Remarque – La compatibilité entre StarOffice Basic et VBA concerne le langage StarOffice Basic, ainsi que la bibliothèque d'exécution. L'API et l'éditeur de boîte de dialogue StarOffice ne sont *pas* compatibles avec VBA : la standardisation de ces interfaces aurait rendu impossibles de nombreux aspects de StarOffice.

Utilisateurs cibles de StarOffice Basic

Le champ d'application de StarOffice Basic commence là où s'arrêtent les fonctions standard de StarOffice. StarOffice Basic permet en effet d'automatiser les tâches récurrentes, d'établir des liens vers d'autres programmes (vers un serveur de base de données, par exemple) et d'exécuter les activités complexes par simple clic de bouton grâce à des scripts prédéfinis.

StarOffice Basic offre un accès complet à toutes les fonctions StarOffice, qu'il prend en charge, modifie les types de document et fournit des options de création de boîtes de dialogue personnelles.

Utilisation de StarOffice Basic

Tous les utilisateurs de StarOffice peuvent utiliser StarOffice Basic sans aucun autre programme ni aucune autre aide. L'installation standard de StarOffice Basic comprend tous les composants nécessaires à la création de macros Basic, à savoir :

- **L'environnement de développement intégré** (IDE, Integrated Development Environment) dispose d'un éditeur pour la création et le test des macros.
- **L'interpréteur** sert à exécuter les macros de StarOffice Basic.
- **Les interfaces** vers diverses applications StarOffice permettent d'accéder directement aux documents Office.

Informations complémentaires

Le choix des composants de l'API StarOffice qui sont traités dans ce manuel a été effectué en fonction des avantages pratiques qu'ils apportent au programmeur StarOffice Basic. En général, les interfaces ne sont que partiellement traitées. Pour une description plus détaillée, reportez-vous à la référence de l'API disponible sur Internet à l'adresse :

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

Le Developer's Guide (Guide du développeur) décrit l'API StarOffice de manière plus détaillée que ce manuel, mais est avant tout destiné aux programmeurs Java et C++. Les personnes déjà familiarisées avec la programmation StarOffice Basic pourront trouver dans le Developer's Guide des informations complémentaires sur StarOffice Basic et la programmation dans StarOffice. Vous pouvez télécharger le Developer's Guide sur Internet à l'adresse :

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

Nous conseillons aux programmeurs qui souhaitent utiliser Java or C++ plutôt que StarOffice Basic de consulter le Developer's Guide de StarOffice. La programmation de StarOffice avec Java ou C++ est une tâche beaucoup plus complexe que dans StarOffice Basic.

CHAPITRE 2

Langage de StarOffice Basic

StarOffice Basic fait partie de la famille des langages de type Basic. De nombreuses parties de StarOffice Basic sont identiques à Microsoft Visual Basic pour Applications (VBA) et à Microsoft Visual Basic. Les utilisateurs ayant déjà travaillé avec ces langages se familiariseront rapidement avec StarOffice Basic.

De même, les programmeurs utilisant d'autres langages (tels que Java, C++ ou Delphi) se familiariseront aisément avec StarOffice Basic. C'est un langage de programmation procédural abouti qui n'utilise plus de structures de contrôle rudimentaires, comme `GoTo` et `GoSub`.

Vous pouvez également profiter des avantages de la programmation orientée objet puisque StarOffice Basic possède une interface permettant d'utiliser des bibliothèques d'objets externes. L'intégralité de l'API StarOffice est fondée sur ces interfaces, qui sont décrites en détail plus loin.

Ce chapitre propose un aperçu général des éléments clés et des structures du langage StarOffice Basicainsi que du cadre dans lequel les applications et les bibliothèques sont orientées vers StarOffice Basic.

Présentation du programme StarOffice Basic

StarOffice Basic est un langage interprété. Contrairement à C++ ou à Turbo Pascal, le compilateur de StarOffice ne crée pas de fichiers exécutables ou auto-extractibles, capables de s'exécuter de façon autonome. En revanche, vous pouvez exécuter un programme StarOffice Basic simplement en cliquant sur un bouton. Le code est d'abord vérifié pour rechercher les erreurs patentes, puis exécuté ligne par ligne.

Lignes de programme

L'interpréteur Basic fonctionne ligne par ligne, ce qui le différencie des autres langages de programmation. Alors que l'emplacement des retours à la ligne dans le code source des programmes Java, C++ ou Delphi est indifférent, chaque ligne d'un programme Basic constitue une unité indépendante. Les appels de fonction, les expressions mathématiques et les autres éléments du langage, comme les en-têtes de fonction ou de boucle, doivent commencer et finir à l'intérieur d'une même ligne.

Si l'espace est insuffisant ou si cela donne des lignes trop longues, il est possible de lier plusieurs lignes ensemble par l'ajout de caractères de soulignage (_). L'exemple suivant montre comment lier les quatre lignes d'une expression mathématique :

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

Remarque – Le caractère de soulignage doit toujours être le dernier de la ligne à lier et ne doit être suivi d'aucun espace ou tabulation, sans quoi le code provoque une erreur.

Parallèlement à la liaison de plusieurs lignes, StarOffice Basic permet d'employer le signe de ponctuation ':' pour diviser une ligne en différentes sections afin de pouvoir y placer plusieurs expressions. Les assignations telles que :

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

peuvent être écrites de la façon suivante :

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

Commentaires

Outre le code de programme à exécuter, un programme StarOffice Basic peut contenir des commentaires qui expliquent les différentes parties du programme et fournissent des informations importantes qui peuvent s'avérer utiles par la suite.

StarOffice Basic propose deux méthodes pour insérer des commentaires dans le code :

- Tous les caractères suivant une apostrophe sont considérés comme des commentaires :

```
Dim A    ' Ceci est un commentaire pour la variable A
```

- Le mot-clé `Rem`, suivi du commentaire :

```
Rem Ce commentaire est inséré grâce au mot-clé Rem.
```

Un commentaire comprend généralement tous les caractères jusqu'à la fin de la ligne. StarOffice Basic interprète la ligne suivante de nouveau comme une instruction normale. Si les commentaires s'étendent sur plusieurs lignes, chaque ligne doit être signalée comme étant un commentaire :

```
Dim B     ' Ce commentaire pour la variable B étant assez long
                          ' il s'étend sur plusieurs lignes. Le
                          ' signe de commentaire doit donc être répété
                          ' à chaque ligne.
```

Marqueurs

Un programme StarOffice Basic peut contenir des dizaines, des centaines ou même des milliers de *marqueurs* : noms de variables, de constantes, de fonctions, etc. Lorsque vous sélectionnez un nom de marqueur, les règles suivantes s'appliquent :

- Les marqueurs ne peuvent contenir que des lettres latines, des chiffres et des caractères de soulignage (_).
- Le premier caractère d'un marqueur doit être une lettre ou un caractère de soulignage.
- Les marqueurs ne peuvent pas contenir de caractères spéciaux comme é, î, à, ç et œ.
- La longueur maximum d'un marqueur est de 255 caractères.
- Il n'y a pas de distinction entre les lettres majuscules et minuscules. Ainsi, le marqueur `UneVariableTest` désigne la même variable que `uneVariabletext` et que `UNEVARIABLETEST`.

  Il existe cependant une exception à cette règle : les majuscules sont distinguées des minuscules pour les constantes UNO-API. Vous trouverez de plus amples informations sur UNO dans le Chapitre 4.

Remarque – Les règles de construction des marqueurs dans StarOffice Basic ne sont pas les mêmes que dans VBA. Par exemple, avec Option Compatible, StarOffice Basic n'autorise que les caractères spéciaux dans les marqueurs, puisqu'ils peuvent causer des problèmes dans les projets d'envergure internationale.

Voici quelques exemples de marqueurs corrects et incorrects :

```
Nom      ' Correct
Surname5     ' Correct (le chiffre 5 n'est pas au début du marqueur)
Nom de famille   ' Incorrect (les espaces ne sont pas autorisés)
DéjàVu       ' Incorrect (les lettres telles que é, à
                          'ne sont pas autorisées)
5Surnames    ' Incorrect (le premier caractère ne peut pas être un chiffre)
Nom.de,famille   ' Incorrect (les virgules et les points
                 ' ne sont pas autorisés)
```

Utilisation des variables

Déclaration implicite de variables

Les langages Basic sont conçus pour être conviviaux. Ainsi, StarOffice Basic permet de créer une variable simplement en l'utilisant et sans avoir à la déclarer explicitement. En d'autres termes, une variable existe dès que vous l'intégrez dans le code. En fonction des variables déjà présentes, l'extrait de code suivant peut déclarer jusqu'à trois nouvelles variables :

```
a = b + c
```

La déclaration implicite de variables n'est pas une bonne habitude de programmation, car cela peut amener à introduire involontairement de nouvelles variables, par exemple en faisant une faute de frappe. Au lieu de générer un message d'erreur, l'interpréteur se contente d'initialiser une nouvelle variable correspondant à la faute de frappe avec une valeur de 0. Les erreurs de ce type peuvent être assez difficiles à repérer dans le code.

Déclaration explicite de variables

Pour éviter de générer des erreurs en déclarant des variables implicitement, StarOffice Basic propose une option appelée :

```
Option Explicit
```

Elle doit être indiquée dans la première ligne de code de chaque module. Ainsi, un message d'erreur sera émis chaque fois qu'une variable sera utilisée sans avoir été déclarée au préalable. L'option `Option Explicit` doit se trouver dans tous les modules Basic.

La forme la plus simple pour la déclaration explicite d'une variable est la suivante :

```
    Dim MyVar
```

Dans cet exemple, une variable portant le nom `MyVar` et de type variant est déclarée. Une variable de type variant est une variable universelle pouvant contenir tout type de valeur, comme des chaînes de caractères, des nombres entiers, des nombres à virgule flottante et des valeurs booléennes. Voici quelques exemples de variables de typevariant :

```
MyVar = 'Hello World'      ' Assignation d'une chaîne de caractères
MyVar = 1                   ' Assignation d'un nombre entier
MyVar = 1.0                 ' Assignation d'un nombre à virgule flottante
```

```
MyVar = True                ' Assignation d'une valeur booléenne
```

Les variables déclarées dans l'exemple ci-dessus peuvent même être employées pour contenir différents types de variables à l'intérieur d'un même programme. Même si ce dernier point procure une grande souplesse, mieux vaut restreindre une variable à un type unique. Lorsque StarOffice Basic rencontre un type de variable défini de façon incorrecte pour un contexte particulier, il génère un message d'erreur.

Utilisez la syntaxe suivante pour déclarer une variable associée à un type particulier :

```
Dim MyVar As Integer   ' Déclaration d'une variable de type entier
```

La variable est déclarée en tant qu'entier et peut contenir des valeurs numériques entières. Vous avez également la possibilité d'utiliser la syntaxe suivante pour déclarer une variable de type entier :

```
Dim MyVar%          ' Déclaration d'une variable du type entier
```

L'instruction Dim peut être utilisée pour la déclaration de plusieurs variables :

```
Dim MyVar1, MyVar2
```

Pour assigner aux variables un type permanent, vous devez procéder séparément pour chacune :

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

Si vous ne spécifiez pas de type pour une variable, StarOffice Basic considère qu'elle est de type variant. Par exemple, dans la déclaration de variable suivante, `MyVar1` est de type variant et `MyVar2` de type entier :

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

Les sections suivantes dressent la liste des types de variables disponibles dans StarOffice Basic et indiquent la façon de les utiliser et de les déclarer.

Chaînes de caractères

Les chaînes de caractères constituent, avec les nombres, les types de base les plus importants de StarOffice Basic. Une chaîne est constituée d'une suite de caractères consécutifs. L'ordinateur stocke les chaînes en interne sous forme d'une suite de nombres, où chacun correspond à un caractère particulier.

D’un ensemble de caractères ASCII à Unicode

Les ensembles de caractères font la correspondance entre les caractères d’une chaîne et les codes correspondants (chiffres et lettres) dans un tableau qui décrit comment l’ordinateur affiche la chaîne.

Le jeu de caractères ASCII

Le jeu de caractères ASCII est un ensemble de codes représentant les chiffres, les caractères et les symboles spéciaux sur un octet. Les codes ASCII de 0 à 127 représentent l’alphabet et des symboles courants (comme les points, les virgules et les parenthèses) ainsi que certains caractères de contrôle spéciaux pour l’affichage à l’écran ou l’impression. Le jeu de caractères ASCII est couramment utilisé comme format standard pour l’échange de données texte entre ordinateurs.

Cependant, il manque notamment à ce jeu toute une plage de caractères spéciaux employés en Europe, comme â, ä et î, ainsi que d’autres formats de caractères, comme l’alphabet cyrillique.

Le jeu de caractères ANSI

Microsoft a utilisé pour son produit Windows le jeu de caractères ANSI (American National Standards Institute) qui a été peu à peu étendu pour inclure les caractères absents du jeu de caractères ASCII.

Pages de code

Les ensembles de caractères ISO 8859 constituent une norme internationale. Les 128 premiers caractères du jeu ISO correspondent au jeu de caractères ASCII. Le standard ISO apporte de nouveaux jeux de caractères (*pages de code*) afin de pouvoir afficher correctement un plus grand nombre de langues. L’inconvénient est cependant qu’une même valeur peut représenter différents caractères dans différentes langues.

Unicode

Unicode augmente la longueur d’un caractère à quatre octets et combine différents jeux de caractères afin de créer un standard permettant de représenter autant de langues que possible. La version 2.0 d’Unicode est à présent prise en charge par de nombreux programmes, dont StarOffice et StarOffice Basic.

Variables de chaîne de caractères

StarOffice Basic enregistre les chaînes de caractères comme des variables de chaîne de caractères en Unicode. Une variable de chaîne de caractères peut contenir jusqu'à 65 535 caractères. En interne, StarOffice Basic enregistre la valeur Unicode associée à chaque caractère. La mémoire de travail nécessaire pour une variable de chaîne de caractères dépend de la longueur de celle-ci.

Exemple de déclaration d'une variable de chaîne de caractères :

```
Dim Variable As String
```

Vous pouvez également écrire cette déclaration sous la forme :

```
Dim Variable$
```

Remarque – Lorsque vous portez des applications VBA, assurez-vous que la longueur maximale autorisée pour les chaînes de caractères dans StarOffice Basic est bien respectée (65 535 caractères).

Spécification de chaînes explicites

Pour assigner une chaîne explicite à une variable de chaîne de caractères, placez la chaîne entre guillemets (').

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

Pour répartir une chaîne sur deux lignes, ajoutez un signe plus à la fin de la première :

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

Pour insérer un guillemet (') dans une chaîne, saisissez-en deux consécutifs à l'endroit voulu :

```
Dim MyString As String
MyString = 'a ''-quotation mark.'    ' donne un ' (guillemet)
```

Nombres

StarOffice Basic prend en charge cinq types de base pour traiter les nombres :

- Integer ;
- Long ;
- Float ;
- Double ;
- Currency.

Variables entières (Integer)

Les variables de type entier (Integer) peuvent stocker tout nombre entier compris entre -32 768 et 32 767. Un entier peut occuper jusqu'à deux octets de mémoire. Pour déclarer une variable de type entier, on utilise le signe %. Les calculs utilisant des variables entières sont très rapides et particulièrement utiles pour les compteurs de boucles. Si vous assignez un nombre à virgule flottante à une variable de type entier, il est arrondi au nombre entier le plus proche.

Exemples de déclaration pour des variables entières :

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

Variables entières longues (Long)

Les variables de type entier long (Long) peuvent stocker tout nombre entier compris entre -2 147 483 648 et 2 147 483 647 et peuvent occuper jusqu'à quatre octets de mémoire. Pour déclarer une variable de type entier long, on utilise le signe &. Les calculs utilisant des variables entières longues sont très rapides et particulièrement utiles pour les compteurs de boucles. Si vous affectez un nombre à virgule flottante à une variable entière, il est arrondi au nombre entier le plus proche.

Exemples de déclaration pour des variables de type entier long :

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

Variables simples (Single)

Les variables simples (Single) peuvent stocker tout nombre à virgule flottante positif ou négatif compris entre 3.402823×10^{38} et 1.401298×10^{-45}. Une variable de type simple peut occuper jusqu'à quatre octets de mémoire. Pour déclarer une variable de type simple, on utilise le symbole '!'.

À l'origine, les variables de type simple servaient à réduire le temps de calcul exigé par les variables doubles, plus précises. Cependant, ces considérations de rapidité ne sont plus vraiment pertinentes, ce qui réduit l'intérêt des variables de type simple.

Exemples de déclaration pour des variables simples :

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

Variables doubles (Double)

Les variables de type double peuvent stocker tout nombre à virgule flottante positif ou négatif compris entre $1.79769313486232 \times 10^{308}$ et $4.94065645841247 \times 10^{-324}$. Une variable de type double peut occuper jusqu'à huit octets de mémoire. Les variables doubles peuvent être utilisées pour des calculs précis. Pour déclarer une variable de type double, on utilise le signe #.

Exemples de déclarations de variables doubles :

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

Variables monétaires (Currency)

Les variables monétaires (Currency) diffèrent des autres types de variables par la manière dont elles gèrent les valeurs. Le signe décimal est fixe et suivi de quatre décimales. La variable peut compter jusqu'à 15 chiffres pour sa partie entière. Une variable monétaire peut stocker toute valeur comprise entre -922 337 203 685 477,5808 et +922 337 203 685 477,5807 et peut occuper jusqu'à huit octets de mémoire. Pour déclarer une variable monétaire, on utilise le signe @.

Les variables monétaires sont surtout destinées aux calculs financiers qui génèrent des erreurs d'arrondi imprévisibles dues à l'emploi de nombres à virgule flottante.

Exemples de déclaration de variables monétaires :

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

Spécification de nombres explicites

Les nombres peuvent être représentés de différentes manières, par exemple au format décimal ou en notation scientifique, voire dans une base différente du système décimal. Les règles suivantes s'appliquent aux caractères numériques dans StarOffice Basic :

Nombres entiers

La méthode la plus simple consiste à utiliser des nombres entiers. Ils sont représentés dans le texte source, sans espace pour séparer le chiffre des milliers :

```
Dim A As Integer
Dim B As Float

A = 1210
B = 2438
```

Les nombres peuvent être précédés d'un signe plus (+) ou moins (-) (avec ou sans espace après le signe) :

```
Dim A As Integer
Dim B As Float

A = + 121
B = - 243
```

Nombres décimaux

Lorsque vous saisissez un nombre décimal, séparez la partie entière de la partie décimale à l'aide d'un point (.). Ainsi, les textes sources peuvent être transférés d'un pays à un autre sans conversion.

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

A = 1223.53      ' est arrondi
B = - 23446.46   ' est arrondi
C = + 3532.76323
```

Vous pouvez utiliser les signes '+' et '-' comme préfixes aux nombres décimaux (ici aussi, avec ou sans espace).

Si vous assignez un nombre décimal à une variable entière, StarOffice Basic l'arrondit par excès ou par défaut.

Notation exponentielle

StarOffice Basic permet d'écrire les nombres en notation exponentielle ; vous pouvez par exemple écrire 1.5e-10 pour représenter le nombre $1{,}5 \times 10^{-10}$ (0.00000000015). La lettre 'e' peut être majuscule ou minuscule et précédée ou non d'un signe plus (+).

Voici quelques exemples corrects et incorrects de nombres au format exponentiel :

```
Dim A As Double

A = 1.43E2        ' Correct
```

```
A = + 1.43E2      ' Correct (espace entre le signe + et le nombre de base)
A = - 1.43E2      ' Correct (espace entre le signe - et le nombre de base)
A = 1.43E-2       ' Correct (exposant négatif)

A = 1.43E -2      ' Incorrect (les espaces sont interdits à
                  ' l'intérieur des nombres)
A = 1,43E-2       ' Incorrect (les virgules sont interdites
                  ' pour séparer la partie décimale)
A = 1.43E2.2      ' Incorrect (l'exposant doit être un nombre entier)
```

Notez que les premier et troisième exemples incorrects ne génèrent aucun message d'erreur, bien qu'ils renvoient des valeurs incorrectes. L'expression :

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

est interprétée comme 1,43 moins 2, soit -0,57. C'est pourtant la valeur $1.43 * 10^2$ (correspondant à 0,0143) qui était attendue. Avec la valeur :

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

StarOffice Basic ignore la partie de l'exposant après le point et considère l'expression comme étant :

```
A = 1.43E2
```

Valeurs hexadécimales

Dans le système hexadécimal (base 16), un nombre à deux chiffres correspond exactement à un octet. Ceci permet de gérer les nombres d'une manière reflétant plus fidèlement l'architecture de la machine. Dans le système hexadécimal, les nombres sont représentés au moyen des chiffres de 0 à 9 et des lettres de A à F. La lettre A correspond au nombre 10 en décimal et la lettre F au nombre 15. Pour utiliser des valeurs hexadécimales pour les nombres entiers dans StarOffice Basic, il suffit de les faire précéder par &H.

```
Dim A As Long

A = &HFF   ' Valeur hexadécimale FF, correspondant à la valeur décimale 255
A = &H10   ' Valeur hexadécimale 10, correspondant à la valeur décimale 16
```

Valeurs octales

StarOffice Basic comprend également le système octal (base 8), qui utilise les chiffres 0 à 7. Les valeurs dans cette base doivent être précédées de &O.

```
Dim A As Long

A = &O77   ' Valeur octale 77, correspondant à la valeur décimale 63
A = &O10   ' Valeur octale 10, correspondant à la valeur décimale 8
```

True (vrai) et False (faux)

Variables booléennes

Les variables booléennes ne peuvent prendre que deux valeurs : `True (vrai)` et `False (faux)`. Elles sont adaptées aux spécifications binaires qui ne peuvent avoir que deux états. Une valeur booléenne est enregistrée en interne sous forme d'entier sur deux octets, 0 correspondant à `False` et toute autre valeur à `True`. Il n'existe pas de symbole de déclaration de type pour les variables booléennes. La déclaration ne peut se faire qu'en utilisant la mention supplémentaire *As Boolean*.

Exemple de déclaration d'une variable booléenne :

```
Dim Variable As Boolean
```

Date et heure

Variables de date

Les variables de date peuvent contenir des valeurs de date et d'heure. Lors de l'enregistrement de valeurs de date, StarOffice Basic emploie un format interne permettant d'effectuer des comparaisons et des opérations mathématiques sur les heures et les dates. Il n'existe pas de symbole de déclaration de type pour les variables de date. La déclaration ne peut se faire qu'en utilisant la mention supplémentaire *As Date*.

Exemple de déclaration d'une variable de date :

```
Dim Variable As Date
```

Champ de données

Outre les variables simples (*scalaires*), StarOffice Basic prend en charge des champs de données (*matrices*). Un champ de données contient plusieurs variables adressées au moyen d'un index.

Matrices simples

Une déclaration de matrice est similaire à celle d'une variable simple. Cependant, contrairement aux variables, le nom de la matrice est suivi de parenthèses dont le contenu spécifie le nombre d'éléments. L'expression :

```
Dim MyArray(3)
```

déclare une matrice de quatre variables de type variant, à savoir `MyArray(0)`, `MyArray(1)`, `MyArray(2)` et `MyArray(3)`.

Vous pouvez également spécifier le type de variables contenues dans une matrice. Par exemple, la ligne suivante déclare une matrice de quatre variables de type entier :

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

Dans les exemples précédents, l'index de la matrice commence toujours par la valeur initiale standard de zéro. Il est également possible de spécifier une plage de validité (valeurs initiale et finale) lors de la déclaration du champ de données. L'exemple suivant déclare un champ de données comprenant six valeurs entières, pouvant être adressées par des valeurs d'index de 5 à 10 :

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

Les valeurs d'index ne sont pas nécessairement positives. L'exemple suivant montre également une déclaration correcte, mais avec des limites négatives pour le champ de données.

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

Il déclare un champ de données d'entiers comprenant 6 valeurs, pouvant être adressées par des valeurs d'index allant de -10 à -5.

Il y a trois limites à respecter lorsque vous définissez des valeurs d'index pour un champ de données :

- La valeur d'index minimale est de -32 768.
- La valeur d'index maximale est de 32 767.
- Le nombre maximum d'éléments (pour une dimension de champ de données) est de 16 368.

Remarque – D'autres limites peuvent s'appliquer aux valeurs d'index pour les champs de données dans VBA. La même considération s'applique également au nombre maximum d'éléments possibles par dimension. Vous trouverez les valeurs qui s'appliquent ici dans la documentation VBA appropriée.

Valeur spécifique pour l'index de début

Généralement, l'index de début d'un champ de données est 0, mais vous pouvez changer cette valeur par défaut en 1 pour toutes les déclarations de champ de données grâce à l'appel :

```
Option Base 1
```

Cet appel doit être inclus dans l'en-tête d'un module si vous voulez qu'il s'applique à l'ensemble des déclarations de matrice du module. Cependant, cet appel n'affecte pas les séquences UNO définies par l'API StarOffice dont l'index commence *toujours* à 0. Pour plus de clarté, évitez d'utiliser Option Base 1.

Option Base 1 se contente de modifier la valeur d'index de début et n'affecte pas le nombre d'éléments d'une matrice. La déclaration :

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

crée 4 variables entières qui peuvent être désignées par les expressions `MyInteger(1)`, `MyInteger(2)`, `MyInteger(3)` et `MyInteger(4)`.

Remarque – Dans StarOffice Basic, l'expression Option Base 1 n'affecte pas le nombre d'éléments d'une matrice, contrairement à ce qui se produit dans VBA. Dans StarOffice Basic, seul l'index de début est modifié. La déclaration `MyInteger(3)` crée trois valeurs entières dans VBA avec des valeurs d'index de 1 à 3, tandis que la même déclaration dans StarOffice Basic crée quatre valeurs entières avec des valeurs d'index de 1 à 4. Avec Option Compatible, StarOffice Basic se comporte comme VBA.

Champs de données multidimensionnels

Outre les champs de données à une dimension, vous pouvez créer des champs de données multidimensionnels dans StarOffice Basic. Les différentes dimensions sont séparées les unes des autres par des virgules. L'exemple :

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

définit une matrice d'entiers à deux dimensions, comportant chacune six index (pouvant être adressées par les valeurs d'index de 0 à 5). La matrice peut stocker un total de 6 × 6 = 36 valeurs entières.

Théoriquement, vous pouvez définir dans StarOffice Basic des matrices ayant des centaines de dimensions. Cependant, en pratique, ce nombre est limité par la quantité de mémoire dont vous disposez.

Modifications dynamiques des dimensions des champs de données

Dans les exemples précédents, les dimensions des champs de données étaient spécifiées. Il est également possible de définir des matrices dont la dimension des champs de données change de façon dynamique. Vous pouvez par exemple définir une matrice pour contenir tous les mots d'un texte commençant par la lettre A. Comme le nombre de ces mots n'est pas connu au départ, vous devez pouvoir modifier les limites du champ par la suite. Pour effectuer cette opération dans StarOffice Basic, utilisez l'appel de fonction suivant :

```
ReDim MyArray(10)
```

Remarque – Contrairement à VBA, où seules les matrices dynamiques peuvent être dimensionnées avec `Dim MyArray()`, StarOffice Basic permet de modifier les matrices statiques et dynamiques grâce à `ReDim`.

L'exemple suivant modifie la dimension de la matrice initiale pour qu'elle puisse stocker 11 ou 21 valeurs :

```
Dim MyArray(4) As Integer         ' Déclaration avec cinq éléments

' ...

ReDim MyArray(10) As Integer      ' Passe à 11 éléments

' ...

ReDim MyArray(20) As Integer      ' Passe à 21 éléments
```

Lorsque vous redéfinissez les dimensions d'une matrice, vous pouvez utiliser chacune des options présentées dans les sections précédentes. Cela comprend notamment les déclarations de champs de données multidimensionnels et les spécifications explicites de valeurs d'index de début et de fin. Lors de la modification des dimensions d'un champ de données, tout son contenu est perdu. Pour conserver les valeurs d'origine, utilisez la commande `Preserve` :

```
Dim MyArray(10) As Integer            ' Définition des dimensions
                                      ' initiales

' ...

ReDim Preserve MyArray(20)   As Integer      ' Augmente le champ de
```

```
                                        ' données sans affecter
                                        ' le contenu
```

Lorsque vous utilisez `Preserve`, assurez-vous que le nombre de dimensions et le type de variable restent les mêmes.

Remarque – Contrairement à VBA, où seule la limite supérieure de la dernière dimension peut être modifiée lorsque la commande `Preserve` est utilisée, StarOffice Basic permet de modifier également les autres dimensions.

Si vous utilisez `ReDim` avec `Preserve`, vous devez employer le même type de données que celui qui a été indiqué lors de la déclaration initiale du champ de données.

Portée et durée de vie des variables

Une variable dans StarOffice Basic est créée avec une durée de vie limitée, ainsi qu'une portée limitée qui détermine à partir de quel point du programme elle peut être lue et utilisée. La durée pendant laquelle une variable est conservée ainsi que les endroits depuis lesquels il est possible d'y accéder dépendent de son emplacement et de son type.

Variables locales

Les variables déclarées dans une fonction ou une procédure sont appelées variables locales :

```
Sub Test
   Dim MyInteger As Integer

   ' ...
End Sub
```

Les variables locales ne restent valides que durant l'exécution de la fonction ou de la procédure et sont ensuite réinitialisées à zéro. À chaque nouvel appel de la fonction, les valeurs générées précédemment ne sont plus disponibles.

Pour conserver les valeurs précédentes, il faut définir la variable comme `statique` :

```
Sub Test
   Static MyInteger As Integer

   ' ...
```

```
End Sub
```

Remarque – Contrairement à VBA, StarOffice Basic s'assure que le nom d'une variable locale n'est pas utilisé simultanément comme nom de variable globale et privée dans l'en-tête du module. Lorsqu'une application VBA est portée vers StarOffice Basic, il faut modifier tous les noms de variable dupliqués.

Variables du domaine public

Les variables du domaine public sont définies dans la section d'en-tête d'un module par le mot-clé Dim. Ces variables sont accessibles à tous les modules de leur bibliothèque :

Module A :

```
Dim A As Integer

Sub Test
   Flip
   Flop
End Sub

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

Module B :

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

La valeur de la variable `A` n'est pas modifiée par la fonction `Test`, mais elle est augmentée de un dans la fonction `Flip` et diminuée de un dans la fonction `Flop`. Ces deux modifications sont globales.

Vous pouvez également employer le mot-clé `Public` à la place de `Dim` pour déclarer une variable du domaine public :

```
    Public A As Integer
```

Une variable du domaine public n'est accessible que durant l'exécution de la macro associée et est ensuite réinitialisée.

Variables globales

Du point de vue de leur fonction, les variables globales sont semblables aux variables du domaine public, si ce n'est que leurs valeurs sont conservées même après l'exécution de la macro associée. Les variables globales sont déclarées dans la section d'en-tête d'un module avec le mot-clé `Global` :

```
Global A As Integer
```

Variables privées

Les variables `privées` ne sont accessibles que dans le module à l'intérieur duquel elles ont été définies. Pour définir une variable privée, utilisez le mot-clé `Private` :

```
Private MyInteger As Integer
```

Si plusieurs modules contiennent une variable `privée` portant le même nom, StarOffice Basic crée une variable différente pour chaque occurrence. Dans l'exemple suivant, chacun des modules `A` et `B` possède une variable `privée` `C`. Dans la fonction `Test`, la variable `privée` est définie une première fois dans le module`A`, puis définie à nouveau dans le module `B`.

Module A :

```
Private C As Integer

Sub Test
   SetModuleA      ' Définit la variable C du module A
   SetModuleB      ' Définit la variable C du module B

   ShowVarA        ' Affiche la variable C du module A (= 10)
   ShowVarB        ' Affiche la variable C du module B (= 20)
End Sub

Sub SetmoduleA
   A = 10
End Sub

Sub ShowVarA
   MsgBox C        ' Affiche la variable C du module A.
End Sub
```

Module B :

```
Private C As Integer

Sub SetModuleB
   A = 20
End Sub

Sub ShowVarB
```

```
   MsgBox C        ' Affiche la variable C du module B. End Sub
End Sub
```

Constantes

Dans StarOffice Basic, le mot-clé `Const` permet de déclarer une constante.

```
Const A = 10
```

Vous pouvez aussi préciser le type de la constante dans la déclaration :

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

Opérateurs

StarOffice Basic comprend les opérateurs mathématiques, logiques et de comparaison courants.

Opérateurs mathématiques

Les opérateurs mathématiques s'appliquent à tous les types de nombres. L'opérateur + peut également servir à relier des chaînes de caractères.

+	Addition de nombres et de valeurs de date, liaison de chaînes
-	Soustraction de nombres et de valeurs de date
*	Multiplication de nombres
/	Division de nombres
\	Division de nombres avec un résultat entier (arrondi)
^	Élévation de nombres à une puissance
MOD	Modulo (calcule le reste d'une division)

Opérateurs logiques

Les opérateurs logiques permettent de lier des éléments selon les règles de l'algèbre de Boole. Si les opérateurs sont appliqués à des valeurs booléennes, la liaison fournit directement le résultat voulu. S'ils sont utilisés en combinaison avec des entiers ou des entiers longs, la liaison est réalisée au niveau des bits.

AND	Et logique
OR	Ou logique
XOR	Ou exclusif logique
NOT	Négation
EQV	Équivalence (les deux éléments liés ont la valeur `True` ou ils ont tous les deux la valeur `False`)
IMP	Implication (si la première expression est vraie, alors la seconde l'est également)

Opérateurs de comparaison

Les opérateurs de comparaison peuvent s'appliquer à tous les types de variables élémentaires (nombres, dates, chaînes et valeurs booléennes).

=	Égalité de nombres, de dates ou de chaînes
<>	Inégalité de nombres, de dates ou de chaînes
>	Relation 'Supérieur à' pour des nombres, des dates ou des chaînes
>=	Relation 'Supérieur ou égal à' pour des nombres, des dates ou des chaînes
<	Relation 'Inférieur à' pour des nombres, des dates ou des chaînes
<=	Relation 'Inférieur ou égal à' pour des nombres, des dates ou des chaînes

---

Remarque – StarOffice Basic ne prend pas en charge l'opérateur de comparaison VBA `Like`.

Instructions conditionnelles

Les instructions conditionnelles permettent de n'exécuter un bloc de code que lorsqu'une condition particulière est remplie.

If...Then...Else

L'instruction conditionnelle la plus courante est l'instruction `If`, comme dans l'exemple suivant :

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

L'assignation `B = 2` ne se produit que si la valeur de la variable `A` est supérieure à trois. La clause `If/Else` est une variante de l'instruction `If` :

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

Dans cet exemple, la variable `B` reçoit la valeur 2 si la variable `A` est supérieure à 3. Sinon `B` reçoit la valeur 0.

Pour les situations plus complexes, vous pouvez imbriquer plusieurs instructions `If`, comme dans l'exemple suivant :

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

Si la valeur de la variable `A` est zéro, alors `B` reçoit la valeur 0. Si `A` est inférieure à 3 (mais non égale à zéro), alors `B` reçoit la valeur 1. Dans tous les autres cas (c'est-à-dire si `A` est supérieure ou égale à 3), `B` reçoit la valeur 2.

Select...Case

L'instruction `Select...Case` constitue une alternative à l'imbrication des instructions `If`. Elle permet d'employer une même variable pour plusieurs conditions :

```
Select Case DayOfWeek

Case 1:
   NameOfWeekday = "Sunday"
Case 2:
   NameOfWeekday = "Monday"
Case 3:
   NameOfWeekday = "Tuesday"
```

```
Case 4:
   NameOfWeekday = "Wednesday"
Case 5:
   NameOfWeekday = "Thursday"
Case 6:
   NameOfWeekday = "Friday"
Case 7:
   NameOfWeekday = "Saturday"
End Select
```

Dans cet exemple, chaque jour de la semaine correspond à un nombre, de telle manière que la variable `DayOfWeek` a la valeur 1 pour `Sunday`, la valeur 2 pour `Monday`, etc.

La commande `Select` n'est pas limitée aux correspondances directes : vous pouvez également utiliser des opérateurs de comparaison ou des listes d'expressions dans une instruction `Case`. Les exemples suivants illustrent les variantes syntaxiques majeures de la commande :

```
Select Case Var

Case 1 To 5

   ' ... La variable Var est comprise entre 1 et 5

Case 6, 7, 8

   ' ... Var est égale à 6, 7 ou 8

Case Var > 8 And Var < 11

   ' ... Var est supérieur à 8 et inférieur à 11

Case Else

   ' ... Tous les autres cas


End Select
```

Boucles

Une boucle répète l'exécution d'un bloc de code un nombre de fois donné. Certaines boucles peuvent également se répéter indéfiniment.

For...Next

La boucle `For...Next` effectue un nombre de passages déterminé. Le compteur de la boucle définit le nombre de fois qu'elle sera répétée. Dans l'exemple suivant :

```
Dim I
For I = 1 To 10 ' ...  Corps de la boucle
Next I
```

la variable `I` est le compteur de la boucle, avec une valeur initiale de 1. Le compteur est incrémenté de 1 à la fin de chaque passage. Lorsque la variable I vaut 10, la boucle s'arrête.

Pour que le compteur de la boucle soit incrémenté d'une valeur supérieure à 1 à chaque passage, utilisez la fonction `Step` :

```
Dim I

For I = 1 To 10 Step 0.5

   ' ... Corps de la boucle

Next I
```

Dans l'exemple précédent, le compteur est incrémenté de 0,5 à la fin de chaque passage, et la boucle est exécutée 19 fois.

Vous pouvez également utiliser des valeurs de pas négatives :

```
Dim I

For I = 10 To 1 Step -1

   ' ... Corps de la boucle

Next I
```

Dans cet exemple, le compteur commence à 10 et décroît de 1 à chaque passage jusqu'à atteindre la valeur 1.

L'instruction `Exit For` permet de sortir prématurément d'une boucle `For`. Dans l'exemple suivant, la boucle se termine au cinquième passage :

```
Dim I

For I = 1 To 10

   If I = 5 Then Exit For End If

   ' ... Corps de la boucle

Next I
```

Remarque – La variante `For Each...Next` de VBA n'est pas prise en charge dans StarOffice Basic.

Do...Loop

L'instruction `Do...Loop` n'est pas liée à un nombre de passages fixe. En revanche, la boucle `Do...Loop` est répétée jusqu'à ce qu'une certaine condition soit remplie. Il existe quatre variantes de la boucle `Do...Loop` (les exemples suivants utilisent la condition `A > 10`) :

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

   ```
   Do While A > 10
      ' ... Corps de la boucle
   Loop
   ```

   vérifie si la condition est toujours remplie avant chaque passage et n'exécute la boucle que si c'est effectivement le cas.

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

   ```
   Do Until A > 10
      ' ... Corps de la boucle
   Loop
   ```

   exécute la boucle jusqu'à ce que la condition *ne soit plus* remplie.

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

   ```
   Do
      ' ... Corps de la boucle
   Loop While A > 10
   ```

   ne vérifie la condition qu'après le premier passage dans la boucle et s'arrête si elle est *remplie*.

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

   ```
   Do
      ' ... Corps de la boucle
   Loop Until A > 10
   ```

   vérifie aussi la condition après le premier passage, mais exécute la boucle jusqu'à ce que la condition *ne soit plus* remplie.

Tout comme pour la boucle `For...Next`, il est possible de sortir de la boucle `Do...Loop` avant la fin. La commande `Exit Do` permet d'interrompre une boucle à n'importe quel endroit.

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

   ' ... Corps de la boucle
While A > 10
```

Exemple de programme : tri à l'aide de boucles imbriquées

Les boucles ont diverses fonctions, notamment la recherche dans des listes, le renvoi de valeurs ou l'exécution de tâches mathématiques complexes. L'exemple suivant constitue un algorithme utilisant des boucles pour trier une liste par nom.

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

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

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

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

Les valeurs sont interverties deux par deux plusieurs fois, jusqu'à ce qu'elles se trouvent classées par ordre croissant. Comme des bulles, les variables remontent peu à peu vers la bonne position. C'est pourquoi cet algorithme est connu sous le nom de *tri à bulles*.

Procédures et fonctions

Les procédures et les fonctions sont les éléments centraux de la structure d'un programme. Elles forment le cadre permettant de diviser un problème complexe en différentes sous-tâches.

Procédures

Une *procédure* exécute une action sans fournir de valeur explicite. Sa syntaxe est la suivante :

```
Sub Test

   ' ... Code de la procédure à proprement parler

End Sub
```

Dans l'exemple suivant, la procédure `Test` définie contient du code accessible depuis n'importe quel endroit du programme. L'appel se fait en insérant le nom de la procédure à l'endroit adapté du programme :

```
Test
```

Fonctions

Une *function*, tout comme une procédure, regroupe un bloc d'instructions à exécuter en une unité logique. Cependant, contrairement à une procédure, une fonction renvoie une valeur de retour.

```
Function Test

   ' ... Code de la fonction à proprement parler

   Test = 123
End Function
```

La valeur de retour est assignée par une simple affectation. L'assignation ne doit pas forcément être placée à la fin de la fonction, mais peut au contraire se trouver n'importe où à l'intérieur de celle-ci.

La fonction précédente peut être appelée à l'intérieur du programme de la façon suivante :

```
Dim A

A = Test
```

Ce code définit une variable `A` et lui assigne le résultat de la fonction `Test`.

La valeur de retour peut être écrasée plusieurs fois à l'intérieur de la fonction. Comme dans le cas d'une assignation de variable classique, la fonction renvoie la valeur qui lui a été assignée en dernier.

```
Function Test

   Test = 12

   ' ...

   Test = 123

End Function
```

Dans cet exemple, la valeur de retour de la fonction est 123.

Si une affectation est interrompue, la fonction renvoie la valeur `zero` (le nombre 0 pour les valeurs numériques et une chaîne vide pour les chaînes de caractères).

La valeur de retour d'une fonction peut être de tout type. Il se déclare de la même manière que pour une variable :

```
Function Test As Integer

   ' ... Code de la fonction à proprement parler

End Function
```

Si l'indication d'une valeur explicite est interrompue, la valeur de retour sera de type variant.

Interruption prématurée d'une procédure ou d'une fonction

Dans StarOffice Basic, vous pouvez utiliser les commandes `Exit Sub` et `Exit Function` pour mettre fin à une procédure ou à une fonction prématurément, par exemple pour traiter les erreurs. Une telle commande interrompt la procédure ou la fonction et reviennent au programme principal, à l'endroit d'où elle avait été appelée.

L'exemple suivant montre une procédure s'interrompant lorsque la variable `ErrorOccured` prend la valeur `True`.

```
Sub Test
   Dim ErrorOccured As Boolean

   ' ...

   If ErrorOccured Then
      Exit Sub
   End If

   ' ...

End Sub
```

Passage de paramètres

Les fonctions et les procédures peuvent recevoir un ou plusieurs paramètres. Les paramètres essentiels doivent être indiqués entre parenthèses après le nom de la fonction ou de la procédure. L'exemple :

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

définit une procédure qui attend une valeur entière `A` et une chaîne `B` comme paramètres.

Les paramètres sont normalement passés par *référence* dans StarOffice Basic. Les modifications apportées aux variables sont conservées lorsque la procédure ou la fonction se termine :

```
Sub Test
   Dim A As Integer
   A = 10
   ChangeValue(A)
   ' Le paramètre A a maintenant la valeur 20
End Sub
Sub ChangeValue(TheValue As Integer)
    TheValue = 20
End Sub
```

Dans cet exemple, la variable A définie dans la fonction Test est passée comme paramètre à la fonction ChangeValue. Dans la fonction, A prend la valeur 20 qui est passée à la variable TheValue, elle-même conservée lorsque la fonction se termine.

Vous pouvez également passer un paramètre par *valeur* si vous ne souhaitez pas que les modifications effectuées par la suite sur le paramètre affectent sa valeur d'origine. Pour qu'un paramètre soit passé par valeur, le mot-clé ByVal doit précéder la déclaration de la variable dans l'en-tête de la fonction.

Dans l'exemple précédent, si la fonction ChangeValue est remplacée par :

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

alors la variable de niveau supérieur A n'est pas affectée par cette modification. Après l'appel de la fonction ChangeValue, la variable A conserve la valeur 10.

Remarque – La méthode pour passer les paramètres aux procédures et aux fonctions de StarOffice Basic est identique à celle de VBA. Par défaut, les paramètres sont passés par référence. Pour passer des paramètres par valeur, il faut utiliser le mot-clé ByVal. Dans VBA, le mot-clé ByRef peut également être utilisé pour forcer un paramètre à être passé par référence. Dans StarOffice Basic, par défaut les paramètres sont passés par référence, donc ce mot-clé n'est pas pris en charge.

Paramètres facultatifs

Normalement, tous les paramètres d'une fonction ou d'une procédure doivent être passés lors de l'appel à cette fonction ou procédure.

Cependant, dans StarOffice Basic, vous pouvez indiquer que certains paramètres sont *facultatifs*. Dans ce cas, si les valeurs correspondantes ne sont pas incluses dans l'appel, StarOffice Basic passe un paramètre vide. Dans l'exemple :

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

End Sub
```

le paramètre A est obligatoire, alors que le paramètre B est facultatif.

La fonction IsMissing permet de vérifier si un paramètre a bien été passé.

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

   ' Vérifie si le paramètre B est présent
   If Not IsMissing (B) Then
```

```
      B_Local = B      ' Le paramètre B est présent
   Else
      B_Local = 0      ' Le paramètre B n'est pas défini ->
                       ' la valeur par défaut est 0
   End If

   ' ... Début de la fonction elle-même

End Sub
```

L'exemple commence par tester si le paramètre `B` a bien été passé, et, le cas échéant, passe le même paramètre à la variable interne `B_Local`. Si le paramètre correspondant n'est pas défini, la variable `B_ Local` reçoit à la place la valeur par défaut (en l'occurrence la valeur 0).

Remarque – Le mot-clé `ParamArray` proposé par VBA n'est pas pris en charge dans StarOffice Basic.

Récursivité

La récursivité est maintenant possible dans StarOffice Basic. Une procédure ou une fonction récursive a la possibilité de s'appeler elle-même jusqu'à ce qu'une condition de base soit remplie. Lorsque la fonction est appelée avec cette condition de base, elle renvoie un résultat.

L'exemple suivant utilise une fonction récursive pour calculer la factorielle des nombres `42`, `-42` et `3,14`:

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

Function CalculateFactorial( Number )
    If Number  0 Or Number  Int( Number ) Then
        CalculateFactorial = 'Invalid number for factorial!'
    ElseIf Number = 0 Then
        CalculateFactorial = 1
    Else
       ' Voici l'appel récursif :
        CalculateFactorial = Number * CalculateFactorial( Number - 1 )
    Endif
End Function
```

L'exemple renvoie la factorielle du nombre `42` en appelant de façon récursive la fonction `CalculateFactorial` jusqu'à atteindre la condition `0! = 1`.

Remarque – Les niveaux de récursivité sont définis à différents niveaux selon la plate-forme utilisée. Sous Windows, le niveau de récursivité est 5800. Sous Solaris et Linux, le niveau de récursivité est calculé en fonction d'une évaluation de la taille de la pile.

Traitement des erreurs

La correction des erreurs est l'une des parties les plus lourdes de la programmation. StarOffice Basic propose de nombreux outils pour simplifier le traitement des erreurs.

Instruction On Error

L'instruction `On Error` est essentielle pour le traitement des erreurs :

```
Sub Test
   On Error Goto ErrorHandler

   ' ... Tâche au cours de laquelle une
   ' erreur est susceptible de se produire

Exit Sub

ErrorHandler:

   ' ... Code particulier pour le traitement de l'erreur

End Sub
```

La ligne `On Error Goto ErrorHandler` définit la façon dont StarOffice Basic réagit en cas d'erreur. L'instruction `Goto ErrorHandler` indique à StarOffice Basic de quitter la ligne de programme active et d'exécuter le code `ErrorHandler:` .

Commande Resume

La commande `Resume Next` reprend le déroulement du programme après l'exécution du code de traitement d'erreur, à partir de la ligne suivant celle où l'erreur s'est produite :

```
ErrorHandler:

   ' ... Code particulier pour le traitement de l'erreur

   Resume Next
```

La commande `Resume Proceed` permet d'indiquer un point particulier où reprendre le programme après le traitement de l'erreur :

```
ErrorHandler:

   ' ... Code particulier pour le traitement de l'erreur

   Resume Proceed

Proceed:

   ' ... Le programme reprend ici après l'erreur
```

Pour reprendre un programme sans afficher de message d'erreur lorsqu'une erreur se produit, utilisez la syntaxe suivante :

```
Sub Test
   On Error Resume Next

   ' ... Tâche au cours de laquelle une
   ' erreur est susceptible de se produire

End Sub
```

La commande `On Error Resume Next` doit être utilisée avec précaution, car elle a un effet global. Pour plus d'informations, reportez-vous à la section "Astuces pour le traitement d'erreur structuré" à la page 47.

Requêtes portant sur les erreurs

Pour traiter une erreur, il est précieux de disposer d'une description de celle-ci et de savoir où et pourquoi elle est survenue :

- La variable `Err` contient le nombre d'erreurs survenues.
- La variable `Error$` contient une description de l'erreur.
- La variable `Erl` contient le numéro de la ligne où l'erreur s'est produite.

L'appel :

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

indique comment afficher les informations concernant une erreur dans une fenêtre de message.

Remarque – VBA regroupe les messages d'erreur dans un objet statistique nommé `Err`, tandis que StarOffice Basic propose les trois variables : `Err`, `Error$` et `Erl`.

Les informations de statut restent valables jusqu'à ce que le programme rencontre une commande `Resume` ou `On Error`, qui les réinitialise.

Remarque – Dans VBA, la méthode `Err.Clear` de l'objet `Err` réinitialise le statut d'erreur après qu'une erreur s'est produite. Dans StarOffice Basic, les commandes utilisées à cet effet sont `On Error` et `Resume`.

Astuces pour le traitement d'erreur structuré

La commande de définition `On Error` et la commande de retour `Resume` sont toutes deux des variantes de la construction `Goto`.

Pour structurer proprement votre code afin de ne pas générer d'erreurs lorsque vous utilisez cette construction, évitez d'utiliser des commandes de saut sans les surveiller.

La plus grande prudence est recommandée lors de l'utilisation de la commande `On Error Resume Next`, car elle supprime tous les messages d'erreur ouverts.

La meilleure solution consiste à n'utiliser qu'une seule approche pour le traitement des erreurs à l'intérieur d'un programme : séparez le traitement des erreurs du code du programme lui-même et ne revenez pas dans le code initial après l'apparition d'une erreur.

Voici un exemple de procédure de traitement d'erreur :

```
Sub Example

   ' Définit un programme de traitement des erreurs au début de la fonction
   On Error Goto ErrorHandler

   ' ... Code du programme en lui-même

   ' Désactive la fonction gestion d'erreur
   On Error Goto 0
```

```
   ' Fin de l'implémentation du programme normal
Exit Sub

' Début du traitement des erreurs
ErrorHandler:

   ' Vérifie s'il s'agit d'une erreur attendue
   If Err = ExpectedErrorNo Then
      ' ... Gestion de l'erreur
   Else
      ' ... Avertissement d'erreur imprévue
   End If

   On Error Goto 0            ' Désactive le traitement de l'erreur
End Sub
```

Cette procédure commence par la définition d'un programme de traitement des erreurs, suivi du code du programme en lui-même. À la fin du code du programme, le traitement d'erreur est désactivé par l'appel `On Error Goto 0` et l'implémentation de la procédure se termine par la commande `Exit Sub` (à ne pas confondre avec `End Sub`).

L'exemple commence par vérifier si le numéro de l'erreur correspond à celui attendu (stocké dans la constante imaginaire `ExpectedErrorNo`), puis traite l'erreur en conséquence. Si une erreur différente se produit, le système émet un avertissement. Il est important de contrôler le numéro de l'erreur, de manière à pouvoir détecter les erreurs inattendues.

L'appel `On Error Goto 0` à la fin du code réinitialise les informations sur le statut de l'erreur (le code d'erreur dans les variables système `Err`) pour permettre l'identification d'une erreur ultérieure.

CHAPITRE 3

Bibliothèque d'exécution de StarOffice Basic

Les sections suivantes présentent les fonctions centrales de la bibliothèque d'exécution.

Fonctions de conversion

Il arrive souvent qu'une variable d'un certain type doive être convertie dans un autre type.

Conversions de type implicites et explicites

Le moyen le plus simple de convertir le type d'une variable est d'utiliser une assignation :

```
Dim A As String
Dim B As Integer

B = 101 A = B
```

Dans cet exemple, la variable `A` est une chaîne, tandis que la variable `B` un nombre entier. StarOffice Basic s'assure que la variable `B` est convertie en chaîne pendant l'assignation à la variable `A`. Cette conversion est bien plus complexe qu'il n'y paraît : l'entier `B` est stocké en mémoire de travail sous la forme d'un nombre sur deux octets. `A`, en revanche, est une chaîne de caractères, et l'ordinateur enregistre chaque caractère (chaque chiffre) dans une valeur sur un ou deux octets. Pour cela, avant de copier le contenu de `B` dans `A`, `B` doit être converti au format interne de `A`.

Contrairement à la plupart des autres langages de programmation, Basic effectue ce type de conversion de manière automatique. Cependant, ceci peut avoir des conséquences désastreuses. En y regardant de près, le code suivant

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

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

qui, à première vue, semble évident, se révèle en fait assez trompeur. L'interpréteur Basic commence par calculer le résultat de l'addition, puis convertit celui-ci en chaîne, ce qui donne la chaîne 2.

Si, en revanche, l'interpréteur Basic commence par convertir les valeurs de départ `B` et `C` en chaînes, puis applique l'opérateur plus au résultat, cela produit la chaîne `11`.

Le même principe s'applique lorsque vous utilisez des variables de type Variant :

```
Dim A
Dim B
Dim C

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

Comme les variables de type Variant peuvent contenir des nombres et des chaînes, il est difficile de prévoir si la variable `A` recevra le nombre 2 ou la chaîne 11.

Vous pouvez éviter les erreurs liées aux conversions de type implicites en vous astreignant à une certaine rigueur dans la programmation, par exemple en évitant d'utiliser le type de données Variant.

Pour éviter d'autres erreurs dues aux conversions de type implicites, StarOffice Basic propose un éventail de fonctions de conversion, que vous pouvez utiliser pour définir quand convertir le type des données d'une opération :

- **CStr(Var)** : convertit tout type de données en chaîne de caractères.
- **CInt(Var)** : convertit tout type de données en valeur entière.
- **CLng(Var)** : convertit tout type de données en valeur longue.
- **CSng(Var)** : convertit tout type de données en valeur simple.
- **CDbl(Var)** : convertit tout type de données en valeur double.
- **CBool(Var)** : convertit tout type de données en valeur booléenne.
- **CDate(Var)** : convertit tout type de données en date.

Vous pouvez vous servir de ces fonctions de conversion pour définir la façon dont StarOffice Basic doit effectuer ces opérations :

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

```
B = 1 C = 1

A = CStr(B + C)       ' B et C sont tout d'abord additionnés, puis
                      ' convertis (on obtient le chiffre 2)
A = CStr(B) + CStr(C)   ' B et C sont convertis en chaîne, puis
                        ' combinés (on obtient la chaîne '11')
```

Au cours du premier exemple d'addition, StarOffice Basic commence par ajouter les variables entières, puis convertit le résultat en chaîne de caractères. La chaîne `2` est assignée à `A`. Dans le second exemple, les variables entières sont d'abord converties en deux chaînes de caractères distinctes, qui sont concaténées au cours de l'assignation. `A` se voit donc assigner la chaîne `11`.

Les fonctions de conversion numériques `CSng` et `CDbl` acceptent également les nombres décimaux. Vous devez utiliser comme séparateur de décimales le symbole défini dans les paramètres régionaux correspondants. Inversement, les méthodes `CStr` utilisent les paramètres régionaux sélectionnés pour les détails du formatage des nombres, des dates et de l'heure.

La fonction `Val` est différente des méthodes `Csng, Cdbl` et `Cstr`. Elle convertit une chaîne en nombre, mais attend toujours un point comme séparateur de décimales.

```
Dim A As String
Dim B As Double

A = "2.22" B = Val(A)       ' est converti correctement, en dépit des
                            ' paramètres régionaux spécifiques
```

Vérification du contenu des variables

Dans certains cas, la date ne peut pas être convertie :

```
Dim A As String
Dim B As Date

A = 'test'
B = A             ' Crée un message d'erreur
```

Dans cet exemple, l'assignation de la chaîne `test` à une variable de date n'a aucun sens, ce qui provoque une erreur de l'interpréteur. Le même principe s'applique lorsque vous tentez d'assigner une chaîne à une variable logique :

```
Dim A As String
Dim B As Boolean

A = 'test'
B = A             ' Crée un message d'erreur
```

Là encore, l'interpréteur Basic indique une erreur.

Il est possible d'éviter ces messages d'erreur en vérifiant le programme avant une assignation, afin de déterminer si le contenu de la variable à assigner correspond au type de la variable cible. Les fonctions de test suivantes de StarOffice Basic ont été conçues à cet effet :

- **IsNumeric(Value)** : vérifie si une valeur est un nombre.
- **IsDate(Value)** : vérifie si une valeur est une date.
- **IsArray(Value)** : vérifie si une valeur est une matrice.

Ces fonctions sont particulièrement utiles lorsque vous demandez une entrée utilisateur. Par exemple, vous pouvez vérifier qu'un utilisateur a bien saisi une date ou un nombre correct.

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

Dans l'exemple ci-dessus, si la variable `UserInput` contient une valeur numérique correcte, elle est assignée à la variable `ValidInput`. Si `UserInput` ne contient pas de nombre correct, `ValidInput` se voit assigner la valeur `0` et le programme émet un message d'erreur.

Alors qu'il existe dans StarOffice Basic des fonctions de test pour vérifier les nombres, les dates et les matrices, il n'existe aucune fonction permettant de vérifier les valeurs booléennes. Il est cependant possible d'imiter cette fonctionnalité en utilisant la fonction `IsBoolean` :

```
Function IsBoolean(Value As Variant) As Boolean
   On Error Goto ErrorIsBoolean:
   Dim Dummy As Boolean

   Dummy = Value

   IsBoolean = True
   On Error Goto 0
Exit Sub

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

La fonction `IsBoolean` définit une variable auxiliaire interne `Dummy` de type Boolean et tente de l'assigner à la valeur transférée. Si l'assignation fonctionne, la fonction renvoie `True`. En cas d'échec, le programme génère une erreur d'exécution, qui évite que la fonction de test ne renvoie une erreur.

Remarque – Si, dans StarOffice Basic, une chaîne contenant une valeur qui n'est pas numérique est assignée à un nombre, StarOffice Basic ne génère pas d'erreur, mais transmet la valeur 0 à la variable. Cette procédure n'est pas la même que dans VBA. Avec ce logiciel, une erreur est générée et l'implémentation du programme est interrompue si une assignation correspondante est exécutée.

Chaînes de caractères

Utilisation de jeux de caractères

Lorsqu'il traite les jeux de caractères, StarOffice Basic utilise le jeu de caractères Unicode. Les fonctions `Asc` et `Chr` permettent d'obtenir la valeur Unicode d'un caractère ou le caractère correspondant à une valeur Unicode donnée. Les expressions suivantes assignent les différentes valeurs Unicode à la variable code :

```
Code = Asc("A")         ' Lettre latine A (valeur Unicode 65)
Code = Asc("—")         ' Caractère Euro (valeur Unicode 8364)
Code = Asc("??")         ' Lettre cyrillique ?? (valeur Unicode 1083)
```

Inversement, l'expression

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

permet d'initialiser la chaîne `MyString` avec la valeur correspondant au nombre `13`, qui correspond à un retour à la ligne forcé.

La commande `Chr` est fréquemment utilisée dans les langages Basic pour insérer des caractères de contrôle dans une chaîne. L'assignation

```
MyString = Chr(9) + 'This is a test' + Chr(13)
```

fait précéder le texte d'un caractère de tabulation (valeur Unicode 9) et le fait suivre d'un retour à la ligne forcé (valeur Unicode `13`).

Accès à une partie de la chaîne

StarOffice Basic propose quatre fonctions renvoyant des parties de chaîne :

- **Left(MyString, Length)** : renvoie les premiers caractères Length de `MyString`.
- **Right(MyString, Length)** : renvoie les derniers caractères Length de `MyString`.

- **Mid(MyString, Start, Length)** : renvoie les premiers caractères Length de `MyString` à partir de la position `Start`.
- **Len(MyString)** : renvoie le nombre de caractères dans `MyString`.

Voici quelques exemples de ces fonctions :

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

MyString = "This is a small test"

MyResult = Left(MyString,5)      ' Renvoie la chaîne "This "
MyResult = Right(MyString, 5)    ' Renvoie la chaîne " test"
MyResult = Mid(MyString, 8, 5)   ' Renvoie la chaîne " a sm"
MyLen = Len(MyString)            ' Renvoie la valeur 21
```

Recherche et remplacement

StarOffice Basic propose la fonction `InStr` pour rechercher une chaîne partielle à l'intérieur d'une autre :

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

Le paramètre `SearchString` spécifie la chaîne à rechercher dans `MyString`. La fonction renvoie un nombre contenant la position à laquelle la chaîne `SearchString` apparaît pour la première fois dans `MyString`. Pour rechercher d'autres occurrences de la chaîne, la fonction propose une option permettant de spécifier la position à partir de laquelle StarOffice Basic commence à chercher. Dans ce cas, la syntaxe de la fonction est :

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

Dans les exemples ci-dessus, `InStr` ne fait pas la distinction entre les majuscules et les minuscules. Pour effectuer une recherche avec `InStr` en respectant la casse, ajoutez le paramètre `0`, comme indiqué dans l'exemple suivant :

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

En utilisant les fonctions d'édition de chaînes indiquées ci-dessus, les programmeurs peuvent rechercher et remplacer une chaîne à l'intérieur d'une autre :

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

 Result = ""
 StartPos = 1
 CurrentPos = 1
```

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

La fonction recherche dans la chaîne transférée `Search` dans une boucle en utilisant `InStr` dans le terme d'origine `Source`. Si elle trouve le terme recherché, elle prend la partie précédant l'expression et l'écrit dans le tampon de retour `Result`. Elle ajoute la section `NewPart` au niveau du terme recherché `Search`. Si aucune occurrence supplémentaire du terme recherché n'est trouvée, la fonction détermine la partie de la chaîne restante et l'ajoute au tampon de retour. Elle renvoie la chaîne ainsi créée comme résultat du processus de remplacement.

Comme le remplacement de parties de séquences de caractères est une des fonctions les plus fréquemment utilisées, la fonction `Mid` de StarOffice Basic a été étendue de manière à pouvoir effectuer cette tâche automatiquement. L'exemple suivant

```
Dim MyString As String

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

remplace trois caractères par la chaîne `est` à partir de la sixième position de la chaîne `MyString`.

Formatage des chaînes

La fonction `Format` formate les nombres sous forme de chaînes. Pour ce faire, la fonction attend une expression `Format`, qui sert de modèle pour le formatage des nombres. Chaque substituant à l'intérieur du modèle assure que cet élément est bien formaté en conséquence dans la valeur de sortie. Les cinq substituants les plus importants à l'intérieur d'un modèle sont *le zéro* (0), *le signe dièse* (#), *le point* (.), *la virgule* (,) et *le signe dollar* ($).

Le caractère *zéro* dans le modèle assure que l'emplacement correspondant sera toujours occupé par un nombre. Si aucun chiffre n'est fourni, un 0 est affiché à sa place.

Un *point* correspond au séparateur de décimales défini par le système d'exploitation dans les paramètres nationaux.

L'exemple ci-dessous montre comment les caractères `zéro` et *point* peuvent définir le nombre de décimales d'une expression :

```
MyFormat = "0.00"

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

De la même manière, il est possible d'ajouter des zéros devant un nombre pour lui donner la longueur souhaitée :

```
MyFormat = "0000.00"

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

Une *virgule* représente le caractère utilisé par le système d'exploitation comme séparateur de milliers, et le *signe dièse* un chiffre ou un emplacement qui n'est affiché que si la chaîne en entrée le nécessite.

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

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

À la place du *signe dollar*, la fonction `Format` affiche le symbole monétaire adapté, défini par le système :

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

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

Remarque – Les instructions de formatage utilisées dans VBA pour la date et l'heure ne sont pas prises en charge par StarOffice Basic.

Date et heure

StarOffice Basic dispose du type de données `Date` qui enregistre les informations de date et d'heure dans un format binaire.

Spécification de la date et de l'heure dans le code de programme

Vous pouvez assigner une date à une variable de date par la simple assignation d'une chaîne :

```
Dim MyDate As Date

MyDate = "1.1.2002"
```

Cette assignation peut fonctionner correctement, car StarOffice Basic convertit automatiquement la valeur de date définie sous forme de chaîne en variable de date. Ce type d'assignation peut cependant provoquer des erreurs, les valeurs de date et d'heure étant définies et affichées différemment selon les pays.

Comme StarOffice Basic utilise les paramètres nationaux du système d'exploitation lors de la conversion d'une chaîne en valeur de date, l'expression indiquée plus haut ne fonctionne correctement que si les paramètres nationaux correspondent au format de la chaîne de caractères.

Afin d'éviter ce problème, il est recommandé d'utiliser la fonction `DateSerial` pour assigner une valeur fixe à une variable de date :

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

Le paramètre de la fonction doit correspondre à l'ordre : année, mois, jour. Cette fonction garantit que la variable reçoit bien la valeur correcte, quels que soient les paramètres nationaux.

La fonction `TimeSerial` formate les informations d'heure de la même manière que la fonction `DateSerial` formate les dates :

```
Dim MyVar As Date

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

Leurs paramètres doivent être spécifiés dans l'ordre : heures, minutes, secondes.

Extraction de la date et de l'heure

Les fonctions suivantes constituent le pendant des fonctions `DateSerial` et `TimeSerial` :

- **Day(MyDate)** : renvoie le jour du mois de `MyDate`.
- **Month(MyDate)** : renvoie le mois de `MyDate`.
- **Year(MyDate)** : renvoie l'année de `MyDate`.
- **Weekday(MyDate)** : renvoie le numéro du jour de la semaine de `MyDate`.
- **Hour(MyTime)** : renvoie les heures de `MyTime`.
- **Minute(MyTime)** : renvoie les minutes de `MyTime`.
- **Second(MyTime)** : renvoie les secondes de `MyTime`.

Ces fonctions extraient les sections de date ou d'heure d'une variable `Date` donnée. L'exemple :

```
Dim MyDate As Date

' ... Initialisation de MyDate

If Year(MyDate) = 2003 Then

   ' ... La date spécifiée est située dans l'année 2003

End If
```

vérifie si la date enregistrée dans `MyDate` se trouve dans l'année 2003. De la même manière, l'exemple

```
Dim MyTime As Date

' ... Initialisation de MyTime

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

   ' ... L'heure spécifiée est comprise entre 12 et 14 heures

End If
```

vérifie si `MyTime` est comprise entre 12 et 14 heures.

La fonction `Weekday` renvoie le numéro du jour de la semaine pour la date indiquée :

```
Dim MyDate As Date
Dim MyWeekday As String

' ... initialise MyDate

Select Case WeekDay(MyDate)
case 1
   MyWeekday = "Sunday"
case 2
```

```
   MyWeekday = "Monday"
case 3
   MyWeekday = "Tuesday"
case 4
   MyWeekday = "Wednesday"
case 5
   MyWeekday = "Thursday"
case 6
   MyWeekday = "Friday"
case 7
   MyWeekday = "Saturday"
End Select
```

Remarque – Le dimanche est considéré comme le premier jour de la semaine.

Obtention de l'heure et de la date système

Les fonctions suivantes de StarOffice Basic permettent d'obtenir l'heure et la date système :

- **Date** : renvoie la date actuelle.
- **Time** : renvoie l'heure actuelle.
- **Now** : renvoie le point présent dans le temps (la date et l'heure combinées dans une seule valeur).

Fichiers et répertoires

L'utilisation des fichiers est une des tâches de base d'une application. L'API StarOffice propose un grand nombre d'objets permettant de créer, d'ouvrir et de modifier des documents Office. Ils sont présentés en détail au Chapitre 4. Indépendamment de cela, il peut y avoir des circonstances où vous aurez besoin d'accéder directement au système de fichiers, de rechercher dans les répertoires ou d'éditer des fichiers texte. La bibliothèque d'exécution de StarOffice Basic propose plusieurs fonctions fondamentales pour ces tâches.

Remarque – Certaines fonctions spécifiques à DOS pour les fichiers et les répertoires ne sont plus proposées par StarOffice 8, ou seulement avec des fonctionnalités limitées. Par exemple, les fonctions `ChDir`, `ChDrive` et `CurDir` ne sont plus prises en charge. Certaines propriétés spécifiques à DOS ne sont plus utilisées dans les fonctions utilisant des propriétés de fichier comme paramètres (par exemple, pour distinguer les fichiers cachés et les fichiers système). Cette évolution était devenue nécessaire pour assurer une indépendance maximum de StarOffice par rapport aux différentes plates-formes.

Administration des fichiers

Recherche dans les répertoires

La fonction `Dir` de StarOffice Basic permet de parcourir les répertoires et sous-répertoires à la recherche de fichiers. Lors de la première requête, vous devez assigner une chaîne contenant le chemin des répertoires dans lesquels effectuer la recherche comme premier paramètre de la fonction `Dir`. Le second paramètre de la fonction `Dir` spécifie le fichier ou le répertoire à rechercher. StarOffice Basic renvoie le nom de la première entrée de répertoire trouvée. Pour obtenir l'entrée suivante, vous devez appeler la fonction `Dir` sans lui passer de paramètres. Si la fonction `Dir` ne trouve pas d'entrée supplémentaire, elle renvoie une chaîne vide.

L'exemple suivant montre comment utiliser la fonction `Dir` pour obtenir la liste de tous les fichiers d'un répertoire. La procédure enregistre les différents noms de fichier dans la variable `AllFiles`, puis l'affiche dans une boîte de message.

```
Sub ShowFiles
   Dim NextFile As String
   Dim AllFiles As String

   AllFiles = ""
   NextFile = Dir("C:\", 0)

   While NextFile  <> ""
      AllFiles = AllFiles & Chr(13) &  NextFile
      NextFile  = Dir
   Wend

   MsgBox AllFiles
End Sub
```

Le `0` utilisé comme second paramètre de la fonction `Dir` indique à celle-ci de ne renvoyer que les noms des fichiers et d'ignorer les répertoires. Les paramètres suivants peuvent être spécifiés ici :

- `0` : renvoie les fichiers normaux.

- `16` : sous-répertoires.

L'exemple suivant est pratiquement identique au précédent, si ce n'est que la fonction `Dir` est appelée avec une valeur 16 en paramètre, et renvoie donc les sous-répertoires d'un dossier et non plus les noms des fichiers.

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

Remarque – Lorsqu'elle est appelée dans StarOffice Basic, la fonction `Dir` utilisée avec le paramètre 16 renvoie uniquement les sous-répertoires d'un dossier. Dans VBA, la fonction renvoie également les noms des fichiers standard, ce qui oblige à opérer un traitement supplémentaire pour n'obtenir que les répertoires. Lorsque l'on utilise la fonction `CompatibilityMode ( true )`, StarOffice Basic, utilisé avec le paramètre 16, se comporte comme les fonctions VBA et Dir, et renvoie des sous-répertoires et des fichiers standard.

Remarque – Les options proposées par VBA pour rechercher dans des répertoires uniquement les fichiers possédant les propriétés *caché, fichier système, archive* et *nom de volume* n'existent pas dans StarOffice Basic, car les fonctions de système de fichiers correspondantes n'existent pas dans tous les systèmes d'exploitation.

Remarque – Les indications de chemin utilisées avec la fonction `Dir` peuvent contenir les substituants * et ? dans VBA comme dans StarOffice Basic. Cependant, dans StarOffice Basic, le substituant * ne peut être que le dernier caractère du nom d'un fichier ou de son extension, ce qui n'est pas le cas dans VBA.

Création et suppression de répertoires

StarOffice Basic dispose de la fonction `MkDir` pour créer des répertoires.

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

Cette fonction crée des répertoires et des sous-répertoires. Tous les répertoires nécessaires à la hiérarchie sont également créés, le cas échéant. Par exemple, si seul le répertoire `C:\SubDir1` existe, un appel

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

crée à la fois le répertoire `C:\SubDir1\SubDir2` et le répertoire `C:\SubDir1\SubDir2\SubDir3`.

La fonction `RmDir` supprime des répertoires.

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

Si le répertoire contient des sous-répertoires ou des fichiers, ceux-ci sont *également supprimés.* Utilisez donc la fonction `RmDir` avec prudence.

Remarque – Dans VBA, les fonctions `MkDir` et `RmDir` ne s'appliquent qu'au répertoire actif. En revanche, dans StarOffice Basic, les fonctions `MkDir` et `RmDir` peuvent servir à créer ou à supprimer plusieurs niveaux de répertoires.

Remarque – Dans VBA, `RmDir` génère un message d'erreur lorsqu'un répertoire contient un fichier. Dans StarOffice Basic, le répertoire *ainsi que tous ses fichiers* sont supprimés. Si vous utilisez la fonction `CompatibilityMode ( true )`, StarOffice Basic se comporte comme VBA.

Copie, attribution d'un nouveau nom, suppression et vérification de l'existence de fichiers

L'appel

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

crée une copie du fichier `Source` sous le nom de `Destination`.

Grâce à la fonction

```
Name OldName As NewName
```

vous pouvez renommer le fichier `OldName` en `NewName`. Le mot clé `As` et l'absence de virgule remontent aux origines du langage Basic.

L'appel

```
Kill(Filename)
```

supprime le fichier `Filename`. Pour supprimer un répertoire (avec tous ses fichiers), utilisez la fonction `RmDir`.

La fonction `FileExists` peut être utilisée pour vérifier l'existence d'un fichier :

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

Lecture et modification des propriétés d'un fichier

Lorsque vous utilisez des fichiers, il peut être important de pouvoir en connaître les propriétés, comme la date de leur dernière modification et leur longueur.

L'appel

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

renvoie certaines des propriétés d'un fichier. La valeur de retour est fournie sous la forme d'un masque de bits qui peut avoir les valeurs suivantes :

- 1 : fichier en lecture seule.
- 16 : nom d'un répertoire.

L'exemple

```
Dim FileMask As Integer
Dim FileDescription As String

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

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

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

MsgBox FileDescription
```

détermine le masque de bits du fichier `test.txt` et vérifie s'il est en lecture seule et s'il s'agit d'un répertoire. Si aucun de ces deux cas ne s'applique, `FileDescription` reçoit la chaîne 'normal'.

Remarque – Les drapeaux utilisés dans VBA pour obtenir les propriétés de fichier *caché*, *fichier système*, *archivé* et *nom de volume* ne sont pas pris en charge par StarOffice Basic, car ils sont propres à Windows et ne sont pas, ou pas complètement, disponibles dans les autres systèmes.

La fonction SetAttr permet de modifier les propriétés d'un fichier. L'appel

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

peut donc être utilisé pour passer un fichier en lecture seule. Il est possible de supprimer un statut de lecture avec l'appel suivant :

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

La date et l'heure de la dernière modification d'un fichier peuvent être obtenues avec la fonction FileDateTime. La date est formatée ici selon les paramètres régionaux utilisés sur le système.

```
FileDateTime("test.txt")   ' Renvoie la date et l'heure
                                          ' de la dernière
                                          ' modification du fichier.
```

La fonction FileLen détermine la longueur d'un fichier en octets (au format entier long).

```
FileLen('test.txt')      ' Renvoie la longueur du fichier en octets.
```

Écriture et lecture de fichiers texte

StarOffice Basic propose un large éventail de méthodes pour lire et écrire des fichiers. Les explications qui suivent portent sur l'utilisation de fichiers texte (*et non* de documents texte).

Écriture de fichiers texte

Pour pouvoir accéder à un fichier texte, vous devez l'avoir ouvert au préalable. Pour ce faire, il faut un *descripteur de fichier* libre, qui identifie clairement le fichier pour les accès ultérieurs.

La fonction FreeFile sert à créer un descripteur de fichier libre. Ce descripteur est passé comme paramètre à l'instruction Open, qui ouvre le fichier. Pour ouvrir un fichier de manière à pouvoir le spécifier comme fichier texte, l'appel Open est :

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

Filename est une chaîne de caractères contenant le nom du fichier. FileNo est le descripteur créé par la fonction FreeFile.

Une fois le fichier ouvert, vous pouvez décrire l'instruction Print ligne par ligne :

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

FileNo désigne ici aussi le descripteur de fichier. Le second paramètre spécifie le texte à enregistrer comme ligne du fichier texte.

Une fois le processus d'écriture achevé, le fichier doit être refermé avec un appel `Close` :

```
    Close #FileNo
```

Là encore, il faut spécifier le descripteur du fichier.

L'exemple suivant montre comment ouvrir, décrire et refermer un fichier texte :

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

Filename = "c:\data.txt"  ' Définit le nom du fichier FileNo = Freefile
                          ' Génère un descripteur de fichier libre

Open Filename For Output As #FileNo  ' Ouvre le fichier (en mode écriture)
Print #FileNo, 'This is a line of text'      ' Enregistre une ligne
Print #FileNo, 'This is another line of text'   ' Enregistre une ligne
Close #FileNo                  ' Ferme le fichier
```

Lecture de fichiers texte

Les fichiers texte sont lus de la même manière qu'ils sont écrits. L'instruction `Open` utilisée pour ouvrir le fichier contient l'expression `For Input` à la place de l'expression `For Output`, et au lieu de la commande Print permettant d'écrire des données, c'est la commande Line Input qui est utilisée pour lire les données.

Enfin, lorsque vous accédez à un fichier texte, l'instruction

```
eof(FileNo)
```

sert à vérifier si vous avez atteint la fin du fichier.

L'exemple suivant montre comment lire un fichier texte :

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

' Définit le nom du fichier
Filename = 'c:\data.txt'

' Génère un descripteur de fichier libre
FileNo = Freefile

' Ouvre le fichier (en mode lecture)
Open Filename For Input As FileNo

' Vérifie si la fin du fichier a été atteinte

Do While not eof(FileNo)
```

```
   ' Read line
   Line Input #FileNo, CurrentLine
   If CurrentLine <>"" then
      Msg = Msg & CurrentLine & Chr(13)
   end if
Loop

' Close file
Close #FileNo

Msgbox Msg
```

Les différentes lignes sont extraites dans une boucle `Do While`, enregistrées dans la variable `Msg`, puis affichées à la fin dans une boîte de message.

Boîtes de message et zones de saisie

StarOffice Basic dispose des fonctions `MsgBox` et `InputBox` pour communiquer facilement avec l'utilisateur.

Affichage des messages

`MsgBox` affiche une boîte d'informations de base, qui peut contenir un ou plusieurs boutons. Dans sa version la plus simple

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

la boîte de message `MsgBox` ne contient que du texte et un bouton OK.

Vous pouvez modifier l'apparence de la boîte d'information avec un paramètre. Le paramètre permet d'ajouter des boutons supplémentaires, de définir un bouton par défaut et d'ajouter un symbole informatif. Les valeurs pour sélectionner les boutons sont :

- 0 : bouton OK.
- 1 : boutons OK et Annuler.
- 2 : boutons Annuler et Réessayer.
- 3 : boutons Oui, Non et Annuler.
- 4 : boutons Oui et Non.
- 5 : boutons Réessayer et Annuler.

Pour définir un bouton par défaut, ajoutez une des valeurs suivantes à la valeur du paramètre choisie dans la liste de sélection des boutons. Par exemple, pour créer trois boutons Oui, Non et Annuler (valeur 3), où Annuler est le bouton par défaut, le paramètre doit avoir une valeur de 3 + 512 = 515.

- 0 : le premier bouton est sélectionné par défaut.
- 256 : le deuxième bouton est sélectionné par défaut.
- 512 : le troisième bouton est sélectionné par défaut.

Enfin, les symboles informatifs suivants sont disponibles et peuvent également être affichés en ajoutant les valeurs appropriées au paramètre :

- 16 : signe stop.
- 32 : point d'interrogation.
- 48 : point d'exclamation.
- 64 : icône Astuce.

L'appel

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

affiche une boîte d'information avec les boutons Oui et Non (valeur 4), dont le second (Non) est sélectionné par défaut (valeur 256), et contenant également un point d'interrogation (valeur 32), 4 + 256 + 32 = 292

Si une boîte d'informations contient plusieurs boutons, il faut utiliser une valeur de retour pour déterminer le bouton sur lequel l'utilisateur a cliqué. Les valeurs de retour possibles dans ce cas sont les suivantes :

- 1 : OK.
- 2 : Annuler.
- 4 : Réessayer.
- 5 : Ignorer.
- 6 : Oui.
- 7 : Non.

Pour l'exemple précédent, le contrôle de la valeur de retour peut se faire de la manière suivante :

```
If MsgBox ("Do you want to continue?",  292) = 6 Then
   ' Bouton Oui sélectionné
Else
   ' Bouton Non sélectionné
End IF
```

En plus du texte d'information et du paramètre pour arranger la boîte d'information, `MsgBox` propose un troisième paramètre pour définir le titre de la boîte de message :

```
MsgBox 'Do you want to continue?',  292, 'Box Title'
```

Si aucun titre n'est indiqué pour la boîte de message, la valeur par défaut est 'soffice'.

Zone de saisie pour demander des chaînes simples

La fonction `InputBox` demande à l'utilisateur de saisir une chaîne simple. Elle constitue donc une alternative simple pour configurer des boîtes de dialogue. `InputBox` réclame trois paramètres standard :

- un texte d'information ;
- un titre pour la boîte de message ;
- une valeur par défaut pouvant être ajoutée dans la zone de saisie.

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

Comme valeur de retour, la fonction `InputBox` renvoie la chaîne saisie par l'utilisateur.

Autres fonctions

Beep

La fonction `Beep` fait émettre un son au système pour avertir l'utilisateur d'une action incorrecte. `Beep` ne réclame aucun paramètre :

```
Beep   ' produit un son d'avertissement
```

Shell

Il est possible de lancer des programmes externes grâce à la fonction Shell.

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

`Pathname` définit le chemin du programme à exécuter. `Windowstyle` définit la fenêtre dans laquelle le programme est lancé. Les valeurs suivantes peuvent être utilisées :

- 0 : le programme est mis en focus et lancé dans une fenêtre cachée.
- 1 : le programme est mis en focus et lancé dans une fenêtre de taille normale.
- 2 : le programme est mis en focus et lancé dans une fenêtre réduite.
- 3 : le programme est mis en focus et lancé dans une fenêtre agrandie.
- 4 : le programme est lancé dans une fenêtre de taille normale, sans être mis en focus.

- 6 : le programme est lancé dans une fenêtre réduite, la fenêtre active reste en focus.
- 10 : le programme est lancé en mode plein écran.

Le troisième paramètre, `Param`, permet d'indiquer des paramètres de ligne de commande à passer au programme à lancer.

Wait

La fonction `Wait` suspend l'exécution du programme pour une durée donnée. La durée d'attente est spécifiée en millisecondes. La commande

```
Wait 2000
```

spécifie d'attendre 2 secondes (2 000 millisecondes).

Environ

La fonction `Environ` renvoie les variables d'environnement du système d'exploitation. Ces variables dépendent du système et de sa configuration. L'appel

```
Dim TempDir

TempDir=Environ ('TEMP')
```

détermine les variables d'environnement du répertoire temporaire du système d'exploitation.

CHAPITRE 4

Introduction à l'API StarOffice

L'API StarOffice est une interface de programmation universelle d'accès à StarOffice. Vous pouvez utiliser l'API StarOffice pour créer, ouvrir, modifier et imprimer des documents StarOffice. Cette interface permet d'étendre les fonctions de StarOffice (grâce à des macros personnelles) et de concevoir des boîtes de dialogue personnalisées.

L'API StarOffice peut être utilisée avec StarOffice Basic, mais également avec d'autres langages de programmation comme Java et C++, grâce à la technique *Universal Network Objects* (UNO) qui fournit une interface compatible avec différents langages de programmation.

Ce chapitre se concentre sur la manière d'utiliser StarOffice dans StarOffice Basic à l'aide d'UNO. Il décrit les principaux concepts d'UNO du point de vue d'un programmeur StarOffice Basic. Vous trouverez dans les chapitres suivants des informations sur la façon de travailler avec les différents composants de l'API StarOffice.

Universal Network Objects (UNO)

StarOffice fournit une interface de programmation sous la forme d'une architecture UNO (Universal Network Objects). Il s'agit d'une interface de programmation orientée objet que StarOffice divise en différents objets permettant un accès contrôlé par programme au package Office.

StarOffice Basic étant un langage de programmation procédural, il a fallu lui ajouter plusieurs constructions linguistiques pour pouvoir utiliser UNO.

Afin de pouvoir utiliser une architecture Universal Network Object dans StarOffice Basic, vous devez déclarer une variable pour l'objet associé. Cette déclaration se fait avec l'instruction `Dim` (reportez-vous au Chapitre 2). Pour déclarer une variable d'objet, vous devez utiliser la désignation du type `Object` :

```
Dim Obj As Object
```

Cet appel déclare une variable d'objet nommée `Obj`.

La variable d'objet ainsi créée doit ensuite être initialisée à l'aide de la fonction `createUnoService` :

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

Cet appel assigne à la variable `Obj` une référence à l'objet qui vient d'être créé. `com.sun.star.frame.Desktop` ressemble à un type d'objet, mais dans la terminologie UNO, on parle plus volontiers de *service* que de type. Selon la philosophie UNO, un Obj est décrit comme étant une référence à un objet prenant en charge le service `com.sun.star.frame.Desktop`. Le terme 'service' employé dans StarOffice Basic correspond donc aux termes *type* et *classe* employés dans d'autres langages de programmation.

Il existe cependant une différence fondamentale : la technologie Universal Network Object peut prendre en charge plusieurs services simultanément. Certains services UNO prennent également en charge d'autres services, si bien qu'à travers un seul objet, vous pouvez accéder à tout un éventail de services. Il est possible, par exemple, que l'objet mentionné précédemment, basé sur le service `com.sun.star.frame.Desktop`, inclue également d'autres services pour charger des documents et mettre fin au programme.

Remarque – Alors que dans VBA la structure d'un objet est définie par la classe à laquelle il appartient, dans StarOffice Basic sa structure est définie par les services qu'il prend en charge. Un objet VBA est toujours assigné à une classe unique. Un objet StarOffice Basic peut quant à lui prendre en charge plusieurs services.

Propriétés et méthodes

Les objets dans StarOffice Basic fournissent plusieurs propriétés et méthodes qui peuvent être appelées à l'aide de ces objets.

Propriétés

Les *propriétés* ressemblent aux propriétés d'un objet : `Filename` et `Title` pour un objet `Document`, par exemple.

Les propriétés sont définies grâce à une simple assignation :

```
Document.Title = 'StarOffice 8 Basic Programmer's Guide'
Document.Filename = 'progman.sxv'
```

Une propriété, tout comme une variable normale, possède un type définissant les valeurs qu'elle peut enregistrer. Les propriétés `Filename` et `Title` évoquées précédemment sont de type chaîne.

Propriétés réelles et propriétés imitées

La plupart des propriétés d'un objet dans StarOffice Basic sont définies comme telles dans la description UNO du service. En plus de ces propriétés 'réelles', il existe également dans StarOffice Basic des propriétés constituées de deux méthodes au niveau UNO. L'une sert à obtenir la valeur de la propriété et l'autre à la définir : il s'agit des méthodes `get` et `set`. La propriété a été virtuellement imitée à partir de deux méthodes. Les objets de caractère dans UNO, par exemple, proposent les méthodes `getPosition` et `setPosition` qui permettent de connaître et de modifier le point clé associé. Le programmeur StarOffice Basic peut ainsi accéder à ces valeurs via la propriété `Position`. Indépendamment de cela, les méthodes d'origine sont également disponibles (dans notre exemple, `getPosition` et `setPosition`).

Méthodes

On peut considérer les méthodes comme des fonctions directement liées à un objet qui permettent d'appeler cet objet. L'objet `Document` précédent, par exemple, propose une méthode `Save`, pouvant être appelée de la manière suivante :

```
Document.Save()
```

Les méthodes, à l'instar des fonctions, peuvent contenir des paramètres et des valeurs de retour. La syntaxe de ces appels de méthode est orientée vers les fonctions classiques. L'appel

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

spécifie également le paramètre `True` pour l'objet Document en invoquant la méthode Save. Une fois la méthode terminée, `Save` enregistre une valeur de retour dans la variable `Ok`.

Modules, services et interfaces

StarOffice fournit des centaines de services. Afin que l'utilisateur en ait une meilleure vue générale, ils ont été regroupés en modules. Ces modules n'ont aucun autre intérêt fonctionnel pour les programmeurs StarOffice Basic. Lorsque vous spécifiez le nom d'un service, seul le nom du module importe, puisqu'il doit aussi apparaître dans le nom indiqué. Le nom complet d'un service est constitué de l'expression

com.sun.star, qui spécifie qu'il s'agit d'un service StarOffice, suivie du nom du module, frame par exemple, et enfin du nom du service en lui-même, comme Desktop. Le nom complet dans ce cas serait alors :

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

En plus des termes module et service, UNO utilise le terme *interface*, que les programmeurs Java connaissent certainement mais qui n'apparaît pas en Basic.

Une interface rassemble plusieurs méthodes. Au sens le plus strict du mot, un service dans UNO ne prend pas en charge des méthodes, mais plutôt des interfaces qui proposent quant à elles différentes méthodes. En d'autres termes, les méthodes sont assignées (sous forme de combinaisons) au service dans des interfaces. Ce détail peut être intéressant en particulier pour les programmeurs Java ou C++ : dans ces langages, en effet, il faut une interface pour appeler une méthode. Dans StarOffice Basic, cela n'a aucune importance. Ici, les méthodes sont appelées directement via l'objet concerné.

Pour bien comprendre le fonctionnement de l'API, il est cependant utile de maîtriser l'assignation des méthodes à différentes interfaces, car un grand nombre d'interfaces sont utilisées dans les différents services. Si vous êtes familier avec une interface, vous pourrez appliquer les connaissances acquises avec un service à un autre.

Certaines interfaces centrales sont utilisées si fréquemment qu'elles sont de nouveau traitées à la fin de ce chapitre, appelées par différents services.

Outils pour l'utilisation d'UNO

Il reste à savoir quels objets – ou services, pour employer la terminologie UNO – prennent en charge quelles propriétés, méthodes et interfaces et comment les déterminer. En plus de ce manuel, vous pouvez vous référer aux sources suivantes pour obtenir des informations complémentaires sur les objets : la méthode supportsService, les méthodes de débogage, sans oublier le Developer's Guide (Guide du développeur) et la référence de l'API.

Méthode supportsService

Certains objets UNO prennent en charge la méthode supportsService qui permet de déterminer si un objet prend en charge ou non un service spécifique. L'appel

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

détermine par exemple si l'objet TextElement prend en charge le service com.sun.star.text.Paragraph.

Propriétés de débogage

Chaque objet UNO de StarOffice Basic sait quelles propriétés, méthodes et interfaces il contient. Il dispose de propriétés qui permettent d'en renvoyer la liste. Les propriétés correspondantes sont :

DBG_properties : renvoie une chaîne contenant toutes les propriétés d'un objet.

DBG_methods : renvoie une chaîne contenant toutes les méthodes d'un objet.

DBG_supportetInterfaces : renvoie une chaîne contenant toutes les interfaces prenant en charge un objet.

Le code de programme suivant montre comment utiliser `DBG_properties` et `DBG_methods` dans des applications réelles. Il commence par créer le service `com.sun.star.frame.Desktop`, puis affiche les propriétés et les méthodes prises en charge dans des boîtes de message.

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

MsgBox Obj.DBG_Propierties MsgBox Obj.DBG_methods
```

Lorsque vous utilisez `DBG_properties`, n'oubliez pas que la fonction renvoie l'ensemble des propriétés qu'un service donné peut théoriquement prendre en charge. Vous n'avez cependant aucune garantie que l'objet concerné puisse également les utiliser. Avant d'appeler une propriété, vous devez donc utiliser la fonction `IsEmpty` pour vérifier qu'elle est effectivement disponible.

Référence de l'API

Vous trouverez des informations complémentaires sur les services disponibles ainsi que leurs interfaces, méthodes et propriétés dans la référence pour l'API StarOffice, sur le site `www.openoffice.org` :

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

Présentation de quelques interfaces centrales

Certaines interfaces de StarOffice se retrouvent dans de nombreux composants de l'API StarOffice. Elles définissent des ensembles de méthodes pour des tâches abstraites pouvant s'appliquer à différents types de problèmes. Voici une présentation des interfaces les plus courantes.

L'origine des objets sera expliquée plus loin dans ce manuel. Nous nous contentons pour l'instant de traiter certains aspects abstraits des objets pour lesquels l'API StarOffice propose certaines interfaces centrales.

Création d'objets contextuels

L'API StarOffice propose deux options de création d'objets. La première se trouve dans la fonction `createUnoService` mentionnée au début de ce chapitre. `createUnoService` crée un objet qui peut être utilisé de manière universelle. Ces objets et services sont également connus sous le nom de *services indépendants du contexte*.

Il existe également des *services contextuels* dont les objets ne sont utiles que s'ils sont utilisés conjointement avec un autre objet. Un objet de dessin de classeur, par exemple, ne peut exister que conjointement avec ce document précis.

Interface `com.sun.star.lang.XMultiServiceFactory`

Les objets contextuels sont généralement créés par une méthode d'objet dont dépend l'objet. La méthode `createInstance`, définie dans l'interface `XMultiServiceFactory`, est notamment utilisée dans les objets Document.

L'objet de dessin mentionné précédemment peut, par exemple, être créé de la manière suivante en utilisant un objet Spreadsheet :

```
Dim RectangleShape As Object

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

Vous pouvez créer un modèle de paragraphe dans un document texte de la même manière :

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

Accès nommé aux objets subordonnés

Les interfaces `XNameAccess` et `XNameContainer` sont utilisées dans des objets contenant des objets subordonnés, qui peuvent être adressés par un nom en langage naturel.

Alors que `XNamedAccess` permet d'accéder aux objets individuels, `XNameContainer` se charge d'insérer, de modifier et de supprimer des éléments.

Interface com.sun.star.container.XNameAccess

L'objet Sheet d'un classeur fournit un bon exemple d'utilisation de XNameAccess. Il regroupe les différentes pages du classeur. Il accède aux différentes pages grâce à la méthode getByName de XNameAccess :

```
Dim Sheets As Object Dim Sheet As Object

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

La méthode getElementNames permet d'obtenir un aperçu des noms de tous les éléments. Elle renvoie comme résultat un champ de données contenant les différents noms. L'exemple suivant montre comment déterminer de cette manière les noms de tous les éléments d'un classeur et les afficher dans une boucle :

```
Dim Sheets As Object Dim SheetNames Dim I As Integer

Sheets = Spreadsheet.Sheets SheetNames = Sheets.getElementNames

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

La méthode hasByName de l'interface XNameAccess indique si un objet subordonné portant un nom donné existe dans l'objet de base. L'exemple suivant affiche un message indiquant à l'utilisateur si l'objet Spreadsheet contient une page nommée Feuille1.

```
Dim Sheets As Object

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

Interface com.sun.star.container.XNameContainer

L'interface XNameContainer permet d'insérer, de supprimer et de modifier des éléments subordonnés dans un objet de base. Les fonctions concernées sont insertByName, removeByName et replaceByName.

Voici un exemple pratique : un document texte contenant un objet StyleFamilies est appelé et l'objet est utilisé pour rendre les modèles de paragraphe (ParagraphStyles) du document disponibles.

```
Dim StyleFamilies As Objects
Dim ParagraphStyles As Objects
Dim NewStyle As Object

StyleFamilies = Textdoc.StyleFamilies
ParagraphStyles = StyleFamilies.getByName("ParagraphStyles")
```

```
ParagraphStyles.insertByName("NewStyle", NewStyle)
ParagraphStyles.replaceByName("ChangingStyle", NewStyle)
ParagraphStyles.removeByName("OldStyle")
```

La ligne `insertByName` insère le style `NewStyle` sous le même nom dans l'objet `ParagraphStyles`. La ligne `replaceByName` change l'objet derrière `ChangingStyle` en `NewStyle`. Enfin, l'appel `removeByName` supprime l'objet derrière `OldStyle` de `ParagraphStyles`.

Accès par index aux objets subordonnés

Les interfaces `XIndexAccess` et `XIndexContainer` sont utilisées dans les objets contenant des objets subordonnés pouvant être adressés par index.

`XIndexAccess` fournit les méthodes d'accès aux différents objets. `XIndexContainer` fournit les méthodes d'insertion et de suppression des éléments.

Interface `com.sun.star.container.XIndexAccess`

`XIndexAccess` propose les méthodes `getByIndex` et `getCount` pour appeler les objets subordonnés : `getByIndex` renvoie un objet avec un index donné et `getCount` renvoie le nombre d'objets disponibles.

```
Dim Sheets As Object Dim Sheet As Object Dim I As Integer

Sheets = Spreadsheet.Sheets

For I = 0 to Sheets.getCount() - 1
  Sheet = Sheets.getByIndex(I)
```

L'exemple montre une boucle parcourant toutes les feuilles d'un classeur l'une après l'autre et enregistrant une référence pour chaque feuille dans la variable d'objet `Sheet`. Notez que lorsque vous travaillez avec des index, `getCount` renvoie le nombre d'éléments. Les éléments de `getByIndex` sont numérotés à partir de 0. La variable servant de compteur à la boucle oscille donc entre 0 et `getCount()-1`.

Interface `com.sun.star.container.XIndexContainer`

L'interface `XIndexContainer` propose les fonctions `insertByIndex` et `removeByIndex`. Les paramètres sont structurés de la même manière que les fonctions correspondantes dans `XNameContainer`.

Accès itératif aux objets subordonnés

Dans certaines instances, un objet peut contenir une liste d'objets subordonnés qui ne peuvent pas être adressés au moyen d'un nom ou d'un index. C'est alors que les interfaces `XEnumeration` et `XenumerationAccess` se révèlent utiles. Elles proposent un mécanisme permettant d'atteindre les différents éléments subordonnés d'un objet, l'un après l'autre, sans recourir à une méthode d'adressage direct.

Interfaces `com.sun.star.container.XEnumeration` et `XenumerationAccess`

L'objet de base doit proposer l'interface `XEnumerationAccess`, qui contient uniquement la méthode `createEnumeration`. Elle renvoie un objet auxiliaire, qui fournit à son tour l'interface `XEnumeration` avec les méthodes `hasMoreElements` et `nextElement`. Ces méthodes permettent d'accéder aux objets subordonnés.

L'exemple suivant parcourt les différents paragraphes d'un texte :

```
Dim ParagraphEnumeration As Object Dim Paragraph As Object

ParagraphEnumeration = Textdoc.Text.createEnumeration

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

L'exemple commence par créer un objet auxiliaire `ParagraphEnumeration`. Il renvoie les paragraphes du texte individuellement, dans une boucle. La boucle se termine dès que la méthode `hasMoreElements` renvoie la valeur `False`, qui signale que la fin du texte a été atteinte.

CHAPITRE 5

Utilisation de documents StarOffice

La structure de l'API StarOffice est telle que vous pouvez utiliser les parties de votre choix pour réaliser différentes tâches. Elle comprend les interfaces et services de création, ouverture, enregistrement, conversion et impression de documents ainsi que d'administration de modèles. Étant donné que ces zones de fonction sont disponibles dans tous les types de documents, elles sont expliquées en premier dans ce chapitre.

StarDesktop

Lors de l'utilisation de documents, deux services sont fréquemment utilisés :

- Le service `com.sun.star.frame.Desktop`, qui est similaire au service noyau de StarOffice et fournit les fonctions de l'objet Frame de StarOffice, sous lequel sont classées toutes les fenêtres de document. Ce service permet également de créer, d'ouvrir et d'importer des documents.
- Le service `com.sun.star.document.OfficeDocument` permet d'effectuer les principales opérations liées aux objets individuels de type Document. Il fournit les méthodes d'enregistrement, d'export et d'impression des documents.

Le service `com.sun.star.frame.Desktop` s'ouvre automatiquement au démarrage de StarOffice. Pour cela, StarOffice crée un objet accessible par saisie du nom global `StarDesktop`.

`com.sun.star.frame.XComponentLoader` est l'interface principale de `StarDesktop`. Elle englobe essentiellement la méthode `loadComponentFromURL`, qui permet la création, l'import et l'ouverture de documents.

Le nom de l'objet `StarDesktop` provient de StarOffice 5, dans laquelle toutes les fenêtres de document étaient intégrées dans une même application appelée `StarDesktop`. Le terme `StarDesktop` n'apparaît plus dans la version actuelle de StarOffice. Il a toutefois été retenu pour l'objet Frame de StarOffice, car il indique clairement qu'il s'agit d'un objet Basic pour l'intégralité de l'application.

L'objet `StarDesktop` assume la position de successeur de l'objet `Application` de StarOffice 5 qui s'appliquait auparavant comme un objet Root. Cependant, contrairement à l'ancien objet `Application`, il permet principalement d'ouvrir des documents. Les fonctions de l'ancien objet `Application` pour l'affichage à l'écran de StarOffice (par exemple, `FullScreen`, `FunctionBarVisible`, `Height`, `Width`, `Top`, `Visible`) ne sont plus utilisées.

Remarque – Alors qu'on accède au document actif via `Application.ActiveDocument` sous Word et via `Application.ActiveWorkbook` sous Excel, dans StarOffice, c'est `StarDesktop` qui se charge de cette tâche. L'objet Document actif est accessible dans StarOffice 7 via la propriété `StarDesktop.CurrentComponent`.

Informations élémentaires sur les documents StarOffice

Lors de l'utilisation de documents StarOffice, il est très utile de pouvoir gérer les questions de base de leur administration dans StarOffice. Cela comprend la manière dont les noms de fichier sont structurés pour les documents StarOffice, ainsi que le format dans lequel les fichiers sont enregistrés.

Noms de fichier en notation URL

Comme StarOffice est une application indépendante de la plate-forme sur laquelle elle s'exécute, elle utilise la notation URL (qui est indépendante des différents systèmes d'exploitation), selon la norme Internet RFC 1738 relative aux noms de fichier. Les noms de fichier standard utilisant ce système commencent par le préfixe

```
file:///
```

suivi du chemin d'accès local. Si le nom du fichier contient des sous-répertoires, ceux-ci sont indiqués par une simple *barre oblique* et non une barre oblique inverse comme sous Windows. Le chemin suivant fait référence au fichier `test.sxw` dans le répertoire doc sur l'unité C.

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

Pour convertir les noms de fichier locaux en URL, StarOffice dispose de la fonction `ConvertToUrl`. Pour convertir un URL en nom de fichier local, StarOffice dispose de la fonction `ConvertFromUrl` :

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

```
'  fournit (sous Windows) c:\doc\test.sxw
```

Cet exemple convertit un nom de fichier local en URL et l'affiche dans une boîte de message. Il convertit ensuite l'URL en nom de fichier local et affiche également le résultat.

La norme Internet RFC 1738, utilisée ici, autorise l'emploi des caractères `0 à 9`, `a à z` et `A à Z`. Tous les autres caractères sont insérés dans les URL sous forme de codes d'échappement. Pour ce faire, ils sont convertis en leur valeur hexadécimale dans le jeu de caractères ISO 8859-1 (ISO-Latin) et précédés du signe de pourcentage. Un espace dans un nom de fichier local, par exemple, devient un `%20` dans l'URL.

Format de fichier XML

Depuis la version 6.0, StarOffice fournit un format de fichier basé sur le langage XML. Grâce à l'emploi de XML, l'utilisateur a la possibilité d'ouvrir et d'éditer les fichiers dans d'autres programmes.

Compression de fichiers

Comme XML utilise des fichiers texte standard, les fichiers obtenus sont généralement très volumineux, c'est pourquoi StarOffice les compresse et enregistre sous forme de fichier ZIP. Grâce à une option de la méthode `storeAsURL`, l'utilisateur peut enregistrer directement les fichiers XML d'origine. Pour plus d'informations, consultez la section "Options de la méthode `storeAsURL`" à la page 87.

Création, ouverture et import de documents

La méthode StarDesktop.loadComponentFromURL(URL, Frame, _ SearchFlags, FileProperties)

`permet d'ouvrir, d'importer et de créer des documents.`

Le premier paramètre de la méthode `loadComponentFromURL` indique l'URL du fichier associé.

Au deuxième paramètre, `loadComponentFromURL` attend un nom pour l'objet Frame de la fenêtre que StarOffice crée en interne pour son administration. Le nom prédéfini `_blank` est généralement utilisé à ce stade et indique à StarOffice de créer une fenêtre. Il est également possible de spécifier la valeur `_hidden`, qui permet de charger le document correspondant tout en le gardant invisible.

L'utilisateur peut ouvrir un document StarOffice avec ces seuls paramètres, car les deux derniers peuvent ne contenir que des substituants (des valeurs factices) :

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

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

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

L'appel ci-dessus ouvre le fichier `text.sxw` et l'affiche dans une nouvelle fenêtre.

Cette méthode permet d'ouvrir autant de documents que souhaités dans StarOffice Basic et de les éditer à l'aide des objets Document renvoyés.

Remarque – `StarDesktop.loadComponentFromURL` remplace les méthodes `Documents.Add` et `Documents.Open` de l'ancienne API StarOffice.

Remplacement du contenu de la fenêtre de document

Les valeurs `_blank` et `_hidden` du paramètre `Frame` indiquent à StarOffice de créer une fenêtre à chaque appel de `loadComponentFromURL`. Il s'avère parfois utile de remplacer le contenu d'une fenêtre existante. Si tel est le cas, l'objet Frame de la fenêtre doit contenir un nom explicite. Ce nom ne doit pas commencer par un trait de soulignement. Par ailleurs, le paramètre `SearchFlags` doit être défini pour que la structure correspondante soit créée, si elle n'existe pas déjà. La constante correspondante de `SearchFlags` est :

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

L'exemple ci-dessous indique comment remplacer le contenu d'une fenêtre ouverte à l'aide du paramètre Frame et de `SearchFlags` :

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

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

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

MsgBox "Cliquez sur OK pour afficher le deuxième document."

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

L'exemple commence par ouvrir le fichier `test.sxw` dans une nouvelle fenêtre avec le nom de cadre `MyFrame`. Lorsque l'utilisateur clique sur OK dans la boîte de message, le contenu de la fenêtre est remplacé par le fichier `test2.sxw`.

Options de la méthode `loadComponentFromURL`

Le quatrième paramètre de la fonction `loadComponentFromURL` est un champ de données `PropertyValue` qui fournit à StarOffice de nombreuses options d'ouverture et de création de documents. Le champ de données doit contenir une structure `PropertyValue` pour chaque option. Le nom de l'option y est enregistré sous forme de chaîne, tout comme sa valeur associée.

`loadComponentFromURL` prend en charge les options suivantes :

- **AsTemplate (Boolean)** : si la valeur est True, la méthode charge un nouveau document sans titre à partir de l'URL spécifié. Si la valeur est False, les fichiers de modèle sont chargés en vue de leur édition.
- **CharacterSet (String)** : définit le jeu de caractères utilisé par un document.
- **FilterName (String)** : spécifie un filtre élaboré pour la fonction `loadComponentFromURL`. Les noms de filtre disponibles sont définis dans le fichier `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **FilterOptions (String)** : définit les options supplémentaires pour les filtres.
- **JumpMark (String)** : le programme passe à la position définie dans JumpMark, une fois un document ouvert.
- **Password (String)** : transfère le mot de passe d'un fichier protégé.
- **ReadOnly (Boolean)** : charge un document en lecture seule.

L'exemple ci-dessous indique comment utiliser l'option `FilterName` pour ouvrir un fichier texte séparé par des virgules dans StarOffice Calc.

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

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

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

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

Le champ de données `FileProperties` englobe une seule valeur, car il contient une seule option. La propriété `Filtername` définit si StarOffice doit utiliser un filtre de texte StarOffice Calc pour ouvrir les fichiers.

Création de documents

StarOffice crée automatiquement un document si celui indiqué dans l'URL est un modèle.

Il est également possible de spécifier un URL `private:factory` si seul un document vide sans aucune adaptation est nécessaire :

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

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

Cet appel crée un document StarOffice Writer vide.

Objets Document

La fonction `loadComponentFromURL` présentée dans la section précédente renvoie un objet Document. Elle prend en charge le service `com.sun.star.document.OfficeDocument`, qui fournit à son tour deux interfaces centrales :

- l'interface `com.sun.star.frame.XStorable`, qui permet d'enregistrer les documents ;
- l'interface `com.sun.star.view.XPrintable`, qui contient les méthodes nécessaires à l'impression des documents.

Remarque – Si vous migrez vers StarOffice 8, vous constaterez que la portée des objets Document est restée sensiblement la même. On y retrouve, par exemple, des méthodes pour enregistrer ou imprimer des documents. Les noms et les paramètres de ces méthodes ont, en revanche, été modifiés.

Enregistrement et export de documents

Les documents StarOffice sont enregistrés directement via l'objet Document. La méthode `store` de l'interface `com.sun.star.frame.XStorable` a été prévue pour cela :

```
Doc.store()
```

Cet appel fonctionne à condition qu'un espace mémoire ait déjà été attribué au document. Ce n'est pas le cas des nouveaux documents. Dans cette instance, la méthode utilisée est `storeAsURL`. Cette méthode est également définie dans `com.sun.star.frame.XStorable` et peut servir à définir l'emplacement du document :

```
Dim URL As String
Dim Dummy()

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

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

Outre les méthodes ci-dessus, `com.sun.star.frame.XStorable` fournit également certaines méthodes d'aide utiles pour l'enregistrement de documents, à savoir :

- **hasLocation()** : indique si un URL a déjà été assigné au document.
- **isReadonly()** : indique si un document est protégé en lecture seule.
- **isModified()** : indique si un document a été modifié depuis son dernier enregistrement.

Le code permettant d'enregistrer un document peut être étendu par ces options pour que celui-ci ne soit enregistré que si l'objet a effectivement été modifié et que le nom de fichier n'est demandé que s'il est réellement nécessaire :

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

Cet exemple commence par vérifier si le document concerné a été modifié depuis son dernier enregistrement. Il ne poursuit le processus d'enregistrement que dans ce cas. Si un URL a déjà été assigné au document et s'il ne s'agit pas d'un document en lecture seule, il est enregistré sous l'URL existant. S'il ne possède pas d'URL ou s'il a été ouvert en lecture seule, il est enregistré sous un nouvel URL.

Options de la méthode `storeAsURL`

Tout comme avec la méthode `loadComponentFromURL`, certaines options peuvent être spécifiées sous forme de champ de données `PropertyValue` grâce à la méthode `storeAsURL`. Elles déterminent la procédure qu'utilise StarOffice pour l'enregistrement d'un document. `storeAsURL` comporte les options suivantes :

- **CharacterSet (String)** : définit le jeu de caractères utilisé par un document.
- **FilterName (String)** : spécifie un filtre élaboré pour la fonction `loadComponentFromURL`. Les noms de filtre disponibles sont définis dans le fichier `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **FilterOptions (String)** : définit les options supplémentaires pour les filtres.
- **Overwrite (Boolean)** : permet d'écraser un fichier existant sans confirmation.
- **Password (String)** : transfère le mot de passe d'un fichier protégé.
- **Unpacked (Boolean)** : enregistre le document (non compressé) dans des sous-répertoires.

L'exemple ci-dessous indique comment utiliser l'option `Overwrite` en conjonction avec `storeAsURL` :

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

```
Dim Url As String

' ... Initialiser le document

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

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

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

L'exemple enregistre ensuite le `Document` sous le nom de fichier spécifié si un fichier existant porte déjà ce nom.

Impression de documents

Tout comme pour l'enregistrement, les documents sont imprimés directement à l'aide de l'objet Document. La méthode `Print` de l'interface `com.sun.star.view.Xprintable` est fournie à cet effet : Sous sa forme la plus simple, l'appel de print est le suivant :

```
Dim Dummy()

Doc.print(Dummy())
```

Comme pour la méthode `loadComponentFromURL`, le paramètre Dummy est un champ de données `PropertyValue` permettant à StarOffice de spécifier plusieurs options d'impression.

Options de la méthode print

La méthode `print` attend en argument un champ de données `PropertyValue` correspondant au paramétrage de la boîte de dialogue d'impression de StarOffice :

- **CopyCount (Integer)** : indique le nombre d'exemplaires à imprimer.
- **FileName (String)** : imprime le document dans le fichier spécifié.
- **Collate (Boolean)** : indique à l'imprimante qu'elle doit rassembler les pages de chaque exemplaire.
- **Sort (Boolean)** : trie les pages lors de l'impression de plusieurs exemplaires (`CopyCount` > 1).
- **Pages (String)** : contient la liste des pages à imprimer (selon la syntaxe spécifiée dans la boîte de dialogue d'impression).

L'exemple suivant indique comment imprimer plusieurs pages d'un document à l'aide de l'option `Pages` :

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

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

Doc.print(PrintProperties())
```

Sélection et paramétrage de l'imprimante

L'interface `com.sun.star.view.XPrintable` fournit la propriété `Printer`, qui permet de sélectionner l'imprimante. Cette propriété reçoit un champ de données `PropertyValue` avec les paramètres suivants :

- **Na,e (String)** : indique le nom de l'imprimante.
- **PaperOrientation (Enum)** : indique l'orientation du papier (valeur `com.sun.star.view.PaperOrientation.PORTRAIT` pour l'orientation portrait, `com.sun.star.view.PaperOrientation.LANDSCAPE` pour l'orientation paysage).
- **PaperFormat (Enum)** : indique le format du papier (`com.sun.star.view.PaperFormat.A4` pour le format DIN A4 ou `com.sun.star.view.PaperFormat.Letter` pour le format US Letter, par exemple).
- **PaperSize (Size)** : indique la taille du papier en centièmes de millimètre.

L'exemple suivant illustre comment changer d'imprimante et définir le format du papier à l'aide de la propriété `Printer`.

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


PaperSize.Width = 20000   ' correspond à 20 cm
PaperSize.Height = 20000   ' correspond à 20 cm

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

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

Doc.Printer = PrinterProperties()
```

L'exemple définit un objet nommé `PaperSize` avec le type `com.sun.star.awt.Size`. Cet objet est nécessaire pour spécifier le format de papier. De plus, il crée un champ de données pour deux entrées `PropertyValue` nommé `PrinterProperties`. Il est ensuite initialisé avec les valeurs à définir et assigné à la propriété `Printer`. Du point de vue d'UNO, l'imprimante n'est pas une propriété réelle, mais en imite une.

Modèles

Les modèles sont des listes nommées contenant des attributs de formatage. Ils couvrent toutes les applications de StarOffice et permettent de simplifier considérablement le formatage. Si l'utilisateur modifie l'un des attributs d'un modèle, StarOffice modifie automatiquement toutes les sections du document en fonction de l'attribut. L'utilisateur peut donc, par exemple, modifier le type de police de tous les titres de niveau un via une modification centrale dans le document. Selon le type de documents, StarOffice reconnaît toute une gamme de types de modèles différents.

StarOffice Writer prend en charge

- les modèles de caractère ;
- les modèles de paragraphe ;
- les modèles de cadre ;
- les modèles de page ;
- les modèles de numérotation.

StarOffice Calc prend en charge

- les modèles de cellule ;
- les modèles de page.

StarOffice Impress prend en charge

- les modèles d'élément de caractères ;
- les modèles de présentation.

Dans la terminologie de StarOffice, les différents types de modèles sont appelés `StyleFamilies` selon le service `com.sun.star.style.StyleFamily` sur lequel ils sont basés. Les objets `StyleFamilies` sont accessibles depuis l'objet Document :

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object

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

Cet exemple utilise la propriété `StyleFamilies` d'un classeur pour établir une liste de tous les modèles de cellules disponibles.

Il est possible d'accéder séparément à chaque modèle au moyen d'un index :

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim CellStyles As Object
```

```
Dim CellStyle As Object
Dim I As Integer

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

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

La boucle ajoutée depuis l'exemple précédent affiche successivement les noms de tous les modèles de cellule dans une boîte de message.

Informations relatives à diverses options de formatage

Chaque type de modèle comporte une gamme complète de propriétés individuelles de formatage. Les principales propriétés de formatage sont expliquées aux emplacements suivants :

- section Propriétés de caractère du chapitre 6 (Documents texte), service`com.sun.star.style.CharacterProperties;`
- section Propriétés de paragraphe du chapitre 6 (Documents texte), service `com.sun.star.text.Paragraph;`
- section Propriétés de cellule du chapitre 7 (Classeurs), service `com.sun.star.table.CellProperties;`
- section Propriétés de page du chapitre 7 (Classeurs), service `com.sun.star.style.PageStyle;`
- section Propriétés d'élément de caractère du chapitre 7 (Classeurs), services divers.

Les propriétés de format ne se limitent en aucun cas aux applications dans lesquelles elles sont expliquées et peuvent être utilisées universellement. La plupart des propriétés de page décrites au Chapitre 7 peuvent par exemple être utilisées non seulement dans StarOffice Calc, mais également dans StarOffice Writer.

Pour plus d'informations sur l'utilisation de modèles, consultez la section *Valeurs par défaut des propriétés de caractère et de paragraphe* du Chapitre 6.

CHAPITRE 6

Documents texte

Outre les chaînes pures, les documents texte peuvent également contenir des informations de formatage. Ces informations peuvent apparaître à un emplacement quelconque du texte. La structure est rendue encore plus complexe par les tableaux. En effet, ces derniers comportent non seulement des chaînes unidimensionnelles, mais également des champs à deux dimensions. La plupart des programmes de traitement de texte offrent maintenant la possibilité d'insérer des objets de dessin, des cadres texte et d'autres objets à l'intérieur d'un texte. Ceux-ci peuvent se trouver en dehors des enchaînements et se situer à un emplacement quelconque sur la page.

Ce chapitre présente les interfaces et services centraux des documents texte. La première section, qui traite de la structure des documents texte, décrit la manière dont un programme StarOffice Basic peut être utilisé pour procéder à des itérations dans un document StarOffice Basic. Elle est axée sur les paragraphes, les portions de paragraphe et leur formatage.

La deuxième section traite de la manière de travailler efficacement avec des documents texte. Pour cela, StarOffice fournit plusieurs objets d'aide, tels que l'objet `TextCursor`, dont le champ d'application dépasse ceux spécifiés dans la première section.

La troisième section va au-delà du travail sur le texte. Elle traite, entre autres, des tableaux, des cadres texte, des champs et repères de texte et des répertoires de contenu.

La création, l'ouverture, l'enregistrement et l'impression des documents sont décrits au Chapitre 5, car ils s'appliquent à tout type de document et pas seulement aux documents texte.

Structure des documents texte

Un document texte peut contenir quatre types d'informations :

- le texte proprement dit ;
- des modèles de formatage des caractères, des paragraphes et des pages ;
- des éléments non textuels tels que des tableaux, des images et des objets de dessin ;
- des paramètres globaux pour le document texte.

Cette section se concentre particulièrement sur le texte et les options de formatage associées.

Remarque – La conception de l'API de 8 StarOffice pour Writer diffère de celle de la version précédente. L'ancienne version de l'API était axée sur l'utilisation de l'objet Selection, orientée principalement en fonction de l'interface dédiée à l'utilisateur final, elle-même axée sur la mise en évidence à l'aide de la souris.

L'API de StarOffice 8 a remplacé ces connexions entre interface utilisateur et interface programmeur. Le programmeur peut ainsi accéder parallèlement à toutes les parties d'une application et travailler avec des objets simultanément à partir de différentes subdivisions d'un document. L'objet Selection n'est plus disponible.

Paragraphes et portions de paragraphe

La partie essentielle d'un document texte est constituée d'une suite de paragraphes. Ceux-ci n'étant ni nommés ni indexés, il est impossible d'accéder directement à l'un d'eux. Ils peuvent néanmoins être parcourus de manière séquentielle à l'aide de l'objet `Enumeration` décrit au Chapitre 4. C'est ainsi que les paragraphes peuvent être édités.

Lorsque vous utilisez l'objet `Enumeration`, il faut néanmoins prendre en compte une particularité : il ne renvoie pas uniquement les paragraphes, mais également les tableaux (à strictement parler, dans StarOffice Writer, un tableau est un type de paragraphe particulier). C'est pourquoi, avant d'accéder à un objet renvoyé, vous devez vérifier qu'il prend en charge le service `com.sun.star.text.Paragraph` (s'il s'agit d'un paragraphe) ou le service `com.sun.star.text.TextTable` (s'il s'agit d'un tableau).

L'exemple qui suit parcourt le contenu d'un document texte dans une boucle et utilise un message dans chaque cas pour indiquer à l'utilisateur si l'objet en question est un paragraphe ou un tableau.

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

```
Dim TextElement As Object

' Crée un objet
Doc = StarDesktop.CurrentComponent

' Crée un objet Enumeration
Enum = Doc.Text.createEnumeration

' boucle parcourant tous les éléments du texte
While Enum.hasMoreElements
   TextElement = Enum.nextElement

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

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

Cet exemple crée un objet `Doc` qui référence le document StarOffice actif. L'objet `Doc` est ensuite utilisé pour créer un objet `Enumeration` qui parcourt chaque partie du texte (paragraphes et tableaux) et assigne l'élément actif à l'objet `TextElement`. La méthode `supportsService` vérifie si `TextElement` est un paragraphe ou un tableau.

Paragraphes

Le service `com.sun.star.text.Paragraph` donne accès au contenu d'un paragraphe. Le texte du paragraphe peut être extrait et modifié à l'aide de la propriété String :

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

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

While Enum.hasMoreElements
   TextElement = Enum.nextElement

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

Cet exemple ouvre le document texte actif et le parcourt à l'aide de l'objet Enumeration. Il utilise la propriété `TextElement.String` dans tous les paragraphes pour accéder aux paragraphes concernés et remplace les chaînes `you`, `too` et `for`

par les caractères `U, 2 et 4`. La fonction `Replace` utilisée pour la substitution n'entre pas dans le domaine linguistique standard de StarOffice Basic. Il s'agit d'une nouvelle utilisation de la fonction d'exemple décrite au Chapitre 3, dans la section "Recherche et remplacement" à la page 54.

Remarque – La procédure d'accès aux paragraphes d'un texte qui est décrite ici est comparable à la liste Paragraphs disponible dans les objets `Range` et `Document` de VBA. Cependant, contrairement à VBA où les paragraphes sont accessibles par leur indice (via l'appel `Paragraph(1)`, par exemple), dans StarOffice Basic il est nécessaire d'utiliser l'objet `Enumeration` décrit ci-dessus.

StarOffice Basic ne comporte aucun équivalent des listes `Characters, Sentences` et `Words` fournies dans VBA. En revanche, vous pouvez utiliser un objet `TextCursor` permettant la navigation à l'échelle des caractères, des phrases et des mots (voir *TextCursor*).

Portions de paragraphe

L'exemple précédent peut effectuer les substitutions souhaitées dans le texte, mais il peut également détruire le formatage.

En effet, un paragraphe est constitué de sous-objets individuels. Chacun de ces sous-objets contient ses propres informations de formatage. Par exemple, si un paragraphe contient en son milieu un mot en gras, il est représenté dans StarOffice par trois portions de paragraphe : la portion précédant le mot en gras, le mot en gras lui-même et la portion située après, définie de nouveau comme normale.

Si le texte du paragraphe est modifié à l'aide de la propriété `String` du paragraphe, StarOffice commence par supprimer les anciennes portions du paragraphe et en insère une nouvelle. Le formatage des sections précédentes est alors perdu.

Pour éviter cela, l'utilisateur peut accéder aux portions concernées du paragraphe au lieu du paragraphe entier. C'est pourquoi les paragraphes comportent leur propre objet `Enumeration`. L'exemple suivant montre une double boucle qui parcourt tous les paragraphes d'un document texte et les portions de paragraphes qu'ils contiennent, puis applique les remplacements de l'exemple précédent :

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object

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

' boucle parcourant tous les éléments du texte
While Enum1.hasMoreElements
```

```
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration

      ' boucle parcourant tous les sous-paragaphes
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         MsgBox "'" & TextPortion.String & "'"
         TextPortion.String = Replace(TextPortion.String, "you", "U")
         TextPortion.String = Replace(TextPortion.String, "too", "2")
         TextPortion.String = Replace(TextPortion.String, "for", "4")

      Wend

   End If

Wend
```

L'exemple parcourt un document texte par une double boucle. La boucle externe se rapporte aux paragraphes du texte. La boucle interne traite les portions de paragraphe à l'intérieur de ces paragraphes. Le contenu de chacune de ces portions de paragraphe est modifié à l'aide de la propriété `String` de la chaîne, comme l'exemple précédent le faisait pour les paragraphes. Mais les portions de paragraphe étant modifiées directement, leur formatage est conservé lors du remplacement de la chaîne.

Formatage

Le texte peut être formaté de différentes façons. La manière la plus simple consiste à assigner les propriétés de format directement à la séquence de texte. On parle dans ce cas de formatage direct. Le formatage direct est particulièrement utilisé dans les documents courts, car les formats peuvent être assignés par l'utilisateur à l'aide de la souris. Vous pouvez, par exemple, mettre en valeur un mot à l'intérieur du texte en le mettant en gras ou centrer une ligne.

En plus du formatage direct, vous pouvez également formater le texte à l'aide de modèles. On parle alors de formatage indirect. Avec le formatage indirect, l'utilisateur assigne un modèle prédéfini à la portion de texte concernée. Si la présentation du texte est modifiée par la suite, il suffit à l'utilisateur de modifier le modèle. StarOffice modifie alors la manière dont toutes les portions de texte utilisant ce modèle sont représentées.

Remarque – Dans VBA, les propriétés de formatage d'un objet se répartissent généralement sur toute une gamme de sous-objets (par exemple `Range.Font`, `Range.Borders`, `Range.Shading`, `Range.ParagraphFormat`). On accède à ces propriétés par des expressions en cascade (par exemple `Range.Font.AllCaps`). Dans StarOffice Basic, en revanche, les propriétés de formatage sont accessibles directement, à l'aide des objets concernés (`TextCursor, Paragraph`, etc.). Les propriétés de caractère et de paragraphe disponibles dans StarOffice sont présentées dans les deux sections suivantes.

Remarque – Dans l'ancienne API StarOffice, un texte était essentiellement formaté à l'aide de l'objet `Selection` et des objets subordonnés (`Selection.Font`, `Selection.Paragraph` et `Selection.Border`, par exemple). Dans la nouvelle API, les propriétés de formatage figurent dans chaque objet (`Paragraph`, `TextCursor`, etc.) et peuvent être appliquées directement. Les propriétés de caractère et de paragraphe disponibles sont décrites ci-dessous.

Propriétés de caractère

Les propriétés de format relatives à un caractère sont décrites comme propriétés de caractère. Elles comprennent notamment le type de gras et le type de police. Les objets permettant de définir des propriétés de caractère doivent prendre en charge le service `com.sun.star.style.CharacterProperties`. StarOffice reconnaît une gamme étendue de services prenant en charge ce service. C'est le cas des services `com.sun.star.text.Paragraph` pour les paragraphes et de `com.sun.star.text.TextPortion` pour les portions de paragraphes.

Le service `com.sun.star.style.CharacterProperties` ne fournit aucune interface, mais une gamme de propriétés permettant de définir et d'appeler des propriétés de caractère. Une liste complète des propriétés de caractère se trouve dans la référence de l'API StarOffice. Les plus importantes sont répertoriées ci-dessous :

- **CharFontName (String)** : nom du type de police sélectionné.
- **CharColor (Long)** : couleur du texte.
- **CharHeight (Float)** : hauteur des caractères en points (pt).
- **CharUnderline (Constant group)** : type de soulignage (constantes selon `com.sun.star.awt.FontUnderline`).
- **CharWeight (Constant group)** : graisse de police (constantes selon `com.sun.star.awt.FontWeight`).
- **CharBackColor (Long)** : couleur d'arrière-plan.
- **CharKeepTogether (Boolean)** : suppression des retours à la ligne automatiques.
- **CharStyleName (String)** : nom du modèle de caractère.

Propriétés de paragraphe

Les informations de formatage qui ne se rapportent pas à des caractères individuels, mais à un paragraphe entier, sont considérées comme des propriétés de paragraphe. Cela inclut les interlignes et la distance entre le paragraphe et le bord de la page. Les propriétés de paragraphe sont accessibles via le service `com.sun.star.style.ParagraphProperties`.

Même les propriétés de paragraphe sont disponibles dans divers objets. Tous les objets supportant le service `com.sun.star.text.Paragraph` prennent également en charge les propriétés de paragraphe dans `com.sun.star.style.ParagraphProperties`.

Une liste complète des propriétés de paragraphe se trouve dans la référence de l'API StarOffice. Les plus courantes sont les suivantes :

- **ParaAdjust (enum)** : orientation verticale du texte (constantes selon `com.sun.star.style.ParagraphAdjust`).
- **ParaLineSpacing (struct)** : interligne (structure selon `com.sun.star.style.LineSpacing`).
- **ParaBackColor (Long)** : couleur d'arrière-plan.
- **ParaLeftMargin (Long)** : marge gauche en centièmes de millimètre.
- **ParaRightMargin (Long)** : marge droite en centièmes de millimètre.
- **ParaTopMargin (Long)** : marge supérieure en centièmes de millimètre.
- **ParaBottomMargin (Long)** : marge inférieure en centièmes de millimètre.
- **ParaTabStops (Array of struct)** : type et position des tabulations (tableau de structures du type `com.sun.star.style.TabStop`).
- **ParaStyleName (String)** : nom du modèle de paragraphe.

Exemple : export HTML simple

L'exemple suivant montre comment manipuler des informations de formatage. Il procède par itérations à l'intérieur d'un document texte et crée un fichier HTML simple. Dans ce but, chaque paragraphe est enregistré dans son propre élément HTML `<P>`. Les portions de paragraphe affichées en gras sont marquées d'un élément HTML `<B>` lors de leur export.

```
Dim FileNo As Integer, Filename As String, CurLine As String

Dim Doc As Object
Dim Enum1 As Object, Enum2 As Object
Dim TextElement As Object, TextPortion As Object

Filename = "c:\text.html"
FileNo = Freefile
Open Filename For Output As #FileNo
```

```
Print #FileNo, "<HTML><BODY>"

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

' boucle parcourant tous les paragraphes
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration
      CurLine = "<P>"

      ' boucle parcourant toutes les portions de paragraphe
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         If TextPortion.CharWeight = com.sun.star.awt.FontWeight.BOLD THEN
            CurLine = CurLine & "<B>" & TextPortion.String & "</B>"
         Else
            CurLine = CurLine & TextPortion.String
         End If
      Wend

      ' exporte la ligne
      CurLine = CurLine & "</P>"
      Print #FileNo, CurLine

   End If
Wend

' écrit le pied de page HTML
Print #FileNo, "</BODY></HTML>"
Close #FileNo
```

La structure de base de cet exemple reprend celle des exemples parcourant les portions de paragraphe d'un texte, présentés plus haut. Les fonctions d'écriture du fichier HTML, ainsi qu'un bloc test vérifiant la graisse des portions de texte correspondantes et mettant en gras des portions de paragraphe par une balise HTML correspondante, ont été ajoutées.

Valeurs par défaut des propriétés de caractère et de paragraphe

Le formatage *direct* est toujours prioritaire sur le formatage *indirect*. Autrement dit, une priorité inférieure est attribuée au formatage à l'aide de modèles par rapport au formatage direct dans un texte.

Il n'est pas facile d'établir si une section d'un document a été formatée directement ou indirectement. Les barres de symboles fournies par StarOffice affichent les propriétés de texte courantes telles que le type de police, la graisse et la taille. Mais elles n'indiquent pas si ces propriétés ont été définies par un modèle ou directement dans le texte.

StarOffice Basic comporte la méthode `getPropertyState`, permettant aux programmeurs de vérifier comment une propriété a été formatée. Elle prend le nom de la propriété en argument et renvoie une constante indiquant l'origine du formatage. Les réponses possibles, définies dans l'énumération `com.sun.star.beans.PropertyState`, sont les suivantes :

- **com.sun.star.beans.PropertyState.DIRECT_VALUE** : la propriété est définie directement dans le texte (formatage direct).
- **com.sun.star.beans.PropertyState.DEFAULT_VALUE** : la propriété est définie par un modèle (formatage indirect).
- **com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE** : la propriété est inconnue. Cet état apparaît, par exemple, lorsque la méthode interroge la propriété graisse d'un paragraphe dont certains mots sont en gras et d'autres non.

L'exemple suivant montre comment les propriétés de format peuvent être éditées dans StarOffice. Il recherche dans un texte les portions de paragraphe qui ont été définies en gras par formatage direct. S'il en trouve une, il supprime le formatage direct à l'aide de la méthode `setPropertyToDefault` et assigne à la portion de paragraphe un modèle de caractère `MyBold`.

```
Dim Doc As Object
Dim Enum1 As Object
Dim Enum2 As Object
Dim TextElement As Object
Dim TextPortion As Object

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

' Boucle parcourant toutes les portions de paragraphes
While Enum1.hasMoreElements
   TextElement = Enum1.nextElement
   If TextElement.supportsService("com.sun.star.text.Paragraph") Then
      Enum2 = TextElement.createEnumeration

      ' Boucle parcourant toutes les portions de paragraphes
      While Enum2.hasMoreElements
         TextPortion = Enum2.nextElement
         If TextPortion.CharWeight = _
           com.sun.star.awt.FontWeight.BOLD AND _
           TextPortion.getPropertyState("CharWeight") = _
           com.sun.star.beans.PropertyState.DIRECT_VALUE Then

            TextPortion.setPropertyToDefault("CharWeight")
            TextPortion.CharStyleName = "MyBold"

         End If

      Wend

   End If

Wend
```

Édition de documents texte

La section précédente traitait d'un vaste ensemble d'options dédiées à l'édition de documents texte axées sur les services `com.sun.star.text.TextPortion` et `com.sun.star.text.Paragraph`, permettant d'accéder aux portions de paragraphe et aux paragraphes. Ces services sont adaptés aux applications dans lesquelles le contenu du texte doit être édité par un passage dans une boucle. Pour les tâches plus complexes, StarOffice comporte le service `com.sun.star.text.TextCursor`, notamment pour le parcours d'un document en sens inverse, phrase par phrase ou mot par mot, plutôt que par objets `TextPortions`.

Objet TextCursor

Un objet `TextCursor` dans l'API StarOffice est comparable au curseur visible utilisé dans un document StarOffice. Il marque un certain point à l'intérieur d'un document texte et peut être déplacé dans diverses directions au moyen de commandes. Toutefois, les objets `TextCursor` disponibles dans StarOffice Basic ne doivent pas être confondus avec le curseur visible. Ils correspondent à deux choses différentes.

Remarque – La terminologie diffère de celle de VBA : en ce qui concerne la portée de la fonction, l'objet Range de VBA peut être comparé à l'objet TextCursor de StarOffice et non — comme son nom porterait à le croire — à l'objet Range de StarOffice.

L'objet TextCursor de StarOffice, par exemple, fournit des méthodes de navigation et de modification du texte qui sont incluses dans l'objet Range de VBA (par exemple, MoveStart, MoveEnd, InsertBefore et InsertAfter). Les équivalents correspondants de l'objet TextCursor de StarOffice sont décrits dans les sections suivantes.

Navigation à l'intérieur d'un texte

L'objet `TextCursor` de StarOffice Basic agit indépendamment du curseur visible dans un document texte. Un déplacement d'objet `TextCursor` commandé par le programme n'a aucune incidence sur le curseur visible. Plusieurs objets `TextCursor` peuvent même être ouverts pour le même document et utilisés à divers emplacements indépendamment les uns des autres.

Un objet `TextCursor` est créé par l'appel `createTextCursor` :

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

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

L'objet Cursor créé de cette manière prend en charge le service `com.sun.star.text.TextCursor`, qui comporte toute une gamme de méthodes de navigation dans les documents texte. L'exemple qui suit commence par déplacer l'objet `TextCursor` de dix caractères vers la gauche, et le déplace ensuite de trois caractères vers la droite :

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

Un objet `TextCursor` peut mettre en évidence toute une zone. On peut comparer cela à la mise en évidence d'un emplacement du texte à l'aide de la souris. Le paramètre `False` dans la fonction ci-dessus spécifie si la zone traversée par le curseur doit ou non être mise en évidence. Ainsi, dans l'exemple suivant :

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

l'objet `TextCursor` commence par se déplacer de dix caractères vers la droite sans mise en évidence, puis revient trois caractères en arrière et met ces caractères en évidence. La zone mise en évidence commence donc après le septième caractère du texte et se termine après le dixième.

Les principales méthodes offertes par le service `com.sun.star.text.TextCursor` pour la navigation sont les suivantes :

- **goLeft (Count, Expand)** : déplacement de Count caractères vers la gauche.
- **goRight (Count, Expand)** : déplacement de Count caractères vers la droite.
- **gotoStart (Expand)** : déplacement au début du document texte.
- **gotoEnd (Expand)** : déplacement à la fin du document texte.
- **gotoRange (TextRange, Expand)** : déplacement à l'objet `TextRange` spécifié.
- **gotoStartOfWord (Expand)** : déplacement au début du mot actif.
- **gotoEndOfWord (Expand)** : déplacement à la fin du mot actif.
- **gotoNextWord (Expand)** : déplacement au début du mot suivant.
- **gotoPreviousWord (Expand)** : déplacement au début du mot précédent.
- **isStartOfWord ()** : renvoie la valeur `True` si l'objet `TextCursor` est au début d'un mot.
- **isEndOfWord ()** : renvoie la valeur `True` si l'objet `TextCursor` est à la fin d'un mot.
- **gotoStartOfSentence (Expand)** : déplacement au début de la phrase active.
- **gotoEndOfSentence (Expand)** : déplacement à la fin de la phrase active.
- **gotoNextSentence (Expand)** : déplacement à la fin de la phrase suivante.
- **gotoPreviousSentence (Expand)** : déplacement au début de la phrase précédente.

- **isStartOfSentence ()** : renvoie la valeur `True` si l'objet `TextCursor` est au début d'une phrase.
- **isEndOfSentence ()** : renvoie la valeur `True` si l'objet `TextCursor` est à la fin d'une phrase.
- **gotoStartOfParagraph (Expand)** : déplacement au début du paragraphe actif.
- **gotoEndOfParagraph (Expand)** : déplacement à la fin du paragraphe actif.
- **gotoNextParagraph (Expand)** : déplacement au début du paragraphe suivant.
- **gotoPreviousParagraph (Expand)** : déplacement au début du paragraphe précédent.
- **isStartOfParagraph ()** : renvoie la valeur `True` si l'objet `TextCursor` est au début d'un paragraphe.
- **isEndOfParagraph ()** : renvoie la valeur `True` si l'objet `TextCursor` est à la fin d'un paragraphe.

Le texte est découpé en phrases en fonction des symboles de phrase. Par exemple, les points sont interprétés comme des symboles indiquant la fin des phrases.

Le paramètre `Expand` est une valeur de type Boolean indiquant si la zone traversée lors du déplacement doit être ou non mise en évidence. Toutes les méthodes de navigation renvoient un paramètre indiquant si la navigation a réussi ou si l'action s'est terminée faute de texte.

Voici une liste de plusieurs méthodes d'édition de zones en évidence à l'aide d'un objet `TextCursor` et prenant également en charge le service `com.sun.star.text.TextCursor` :

- **collapseToStart ()** : réinitialise la mise en évidence et positionne l'objet `TextCursor` au début de la zone précédemment mise en évidence.
- **collapseToEnd ()** : réinitialise la mise en évidence et positionne l'objet `TextCursor` à la fin de la zone précédemment mise en évidence.
- **isCollapsed ()** : renvoie la valeur `True` si l'objet `TextCursor` ne recouvre aucune zone mise en évidence.

Formatage du texte avec `TextCursor`

Le service `com.sun.star.text.TextCursor` prend en charge toutes les propriétés de caractère et de paragraphe présentées au début de ce chapitre.

L'exemple suivant montre comment ces propriétés peuvent être associées à un objet `TextCursor`. Il parcourt un document complet et met en gras le premier mot de chaque phrase.

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean
```

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

Do

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

Loop While Proceed
```

L'exemple commence par créer un objet Document pour le texte qui vient d'être ouvert. Il parcourt ensuite par itérations le texte entier, phrase par phrase, met en évidence le premier mot et le formate en gras.

Extraction et modification du contenu d'un texte

Si un objet `TextCursor` contient une zone mise en évidence, ce texte est accessible par la propriété `String` de l'objet `TextCursor`. L'exemple suivant utilise la propriété `String` pour afficher les premiers mots d'une phrase dans une boîte de message :

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

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

Do

   Cursor.gotoEndOfWord(True)
   MsgBox Cursor.String
   Proceed = Cursor.gotoNextSentence(False)
   Cursor.gotoNextWord(False)
Loop While Proceed
```

Le premier mot de chaque phrase peut être modifié de la même manière à l'aide de la propriété `String` :

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

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

Do

   Cursor.gotoEndOfWord(True)
   Cursor.String = "Ups"
   Proceed = Cursor.gotoNextSentence(False)
   Cursor.gotoNextWord(False)
```

```
Loop While Proceed
```

Si l'objet `TextCursor` contient une zone en surbrillance, une assignation de la propriété `String` la remplace par le nouveau texte. Si aucune zone n'est mise en évidence, le texte est inséré à la position de l'objet `TextCursor`.

Insertion de codes de contrôle

Dans certaines situations, la partie à modifier n'est pas le texte du document mais sa structure. StarOffice fournit des codes de contrôle à cet usage. Ces codes s'insèrent dans le texte et influent sur sa structure. Les codes de contrôle sont définis dans le groupe de constantes `com.sun.star.text.ControlCharacter` Les codes de contrôle disponibles dans StarOffice sont les suivants :

- **PARAGRAPH_BREAK** : saut de paragraphe.
- **LINE_BREAK** : retour à la ligne à l'intérieur d'un paragraphe.
- **SOFT_HYPHEN** : point possible de coupure de mot.
- **HARD_HYPHEN** : point obligatoire de coupure de mot.
- **HARD_SPACE** : espace protégé qui ne sera ni étendu, ni compressé dans un texte justifié.

Pour insérer des codes de contrôle, non seulement le curseur, mais également les objets de document texte associés sont nécessaires. L'exemple suivant insère un saut de paragraphe après le 20^{e} caractère d'un texte :

```
Dim Doc As Object
Dim Cursor As Object
Dim Proceed As Boolean

Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor
Cursor.goRight(20, False)

Doc.Text.insertControlCharacter(Cursor, _
   com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK, False)
```

Le paramètre `False` passé dans l'appel de la méthode `insertControlCharacter` assure que la zone mise en évidence par l'objet `TextCursor` est conservée après l'insertion. Si le paramètre `True` est passé ici, `insertControlCharacter` remplace le texte actif.

Recherche de portions de texte

Il est souvent nécessaire de rechercher un terme particulier dans un texte et de l'éditer. Tous les documents StarOffice fournissent pour cela une interface spéciale qui fonctionne toujours selon le même principe : avant chaque recherche, il faut créer ce

que l'on appelle communément un objet `SearchDescriptor`. Il définit ce que StarOffice recherche dans un document. `SearchDescriptor` est un objet prenant en charge le service `com.sun.star.util. SearchDescriptor`. Il peut être créé à l'aide de la méthode `createSearchDescriptor` d'un document :

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

Lorsque l'objet `SearchDescriptor` a été créé, il reçoit le texte à rechercher :

```
SearchDesc.searchString='any text'
```

Du point de vue de son fonctionnement, l'objet `SearchDescriptor` est comparable à la boîte de dialogue de recherche de StarOffice. Comme dans cette boîte de dialogue, les paramètres nécessaires à une recherche peuvent être définis dans l'objet `SearchDescriptor`.

Les propriétés sont fournies par le service `com.sun.star.util.SearchDescriptor` :

- **SearchBackwards (Boolean)** : effectue la recherche en parcourant le texte en arrière et non en avant.
- **SearchCaseSensitive (Boolean)** : distingue les majuscules des minuscules lors de la recherche.
- **SearchRegularExpression (Boolean)** : traite la chaîne à rechercher comme une expression régulière.
- **SearchStyles (Boolean)** : recherche dans le texte le modèle de paragraphe spécifié.
- **SearchWords (Boolean)** : ne recherche que des mots entiers.

La fonction `SearchSimilarity` de StarOffice (ou 'correspondance floue') est également disponible dans StarOffice Basic. Avec cette fonction, StarOffice recherche une expression similaire, mais pas nécessairement exactement semblable à l'expression spécifiée. Le nombre de caractères supplémentaires, supprimés ou modifiés de ces expressions peut être défini individuellement. Les propriétés associées au service `com.sun.star.util.SearchDescriptor` sont les suivantes :

- **SearchSimilarity (Boolean)** : effectue une recherche de similarité.
- **SearchSimilarityAdd (Short)** : nombre de caractères pouvant être ajoutés dans une recherche de similarité.
- **SearchSimilarityExchange (Short)** : nombre de caractères pouvant être remplacés dans une recherche de similarité.
- **SearchSimilarityRemove (Short)** : nombre de caractères pouvant être supprimés dans une recherche de similarité.
- **SearchSimilarityRelax (Boolean)** : tient compte simultanément de toutes les règles de déviation pour l'expression recherchée.

Une fois que l'objet `SearchDescriptor` a été paramétré de façon appropriée, il peut être appliqué au document texte. Pour cela, les documents StarOffice disposent des méthodes `findFirst` et `findNext` :

```
Found = Doc.findFirst (SearchDesc)

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

Loop
```

Cet exemple recherche toutes les correspondances à l'intérieur d'une boucle et renvoie un objet `TextRange`, qui se rapporte au passage de texte trouvé.

Exemple : recherche de similarité

Cet exemple montre comment rechercher le terme 'turnover' dans un texte et mettre le résultat en gras. Une recherche de similarité permet de trouver non seulement le terme 'turnover', mais également sa forme plurielle 'turnovers', ainsi que ses variantes, telles que 'turnover's'. Les expressions trouvées diffèrent au maximum de deux lettres de l'expression recherchée :

```
Found = Doc.findFirst (SearchDesc)

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

Loop
```

Remarque – Le concept de base de la recherche et remplacement dans StarOffice est comparable à celui utilisé dans VBA. Les deux interfaces fournissent un objet, permettant de définir les propriétés de recherche et de remplacement. Cet objet est ensuite appliqué à la zone de texte appropriée pour effectuer l'action. Dans VBA, l'objet auxiliaire chargé de cela est accessible par la propriété Find de l'objet Range ; dans StarOffice Basic, il faut appeler la méthode `createSearchDescriptor` ou `createReplaceDescriptor` du document. Même les propriétés et méthodes de recherche disponibles sont différentes.

Comme dans l'ancienne API StarOffice, la recherche et le remplacement de texte sont effectués via l'objet Document. Cependant, il existait auparavant un objet `SearchSettings` dédié à la définition des options de recherche, tandis que, dans la nouvelle API, les recherches sont désormais effectuées à l'aide d'un objet `SearchDescriptor` ou `ReplaceDescriptor` pour le remplacement automatique de texte. Ces objets ne couvrent pas uniquement les options, mais également le texte recherché et, le cas échéant, le texte de remplacement associé. Les objets Descriptor sont créés à l'aide de l'objet Document, complétés selon les requêtes concernées, puis transférés de nouveau à l'objet Document sous forme de paramètres passés dans les méthodes de recherche.

Remplacement de portions de texte

Tout comme la fonction de recherche, la fonction de remplacement de StarOffice est disponible dans StarOffice Basic. Les deux fonctions sont traitées de manière identique. Un objet spécial stockant les paramètres du processus est tout d'abord nécessaire au processus de remplacement. Il s'appelle `ReplaceDescriptor` et prend en charge le service `com.sun.star.util.ReplaceDescriptor`. Toutes les propriétés de l'objet `SearchDescriptor` décrites dans le paragraphe précédent sont également prises en charge par `ReplaceDescriptor`. Par exemple, la distinction entre majuscules et minuscules peut également être activée et désactivée dans un processus de remplacement, et les recherches de similarité sont également possibles.

L'exemple suivant montre l'utilisation d'objets `ReplaceDescriptor` dans une recherche sur un document StarOffice.

```
Dim I As Long
Dim Doc As Object
Dim Replace As Object
Dim BritishWords(5) As String
Dim USWords(5) As String

BritishWords() = Array("colour", "neighbour", "centre", "behaviour", _
   "metre", "through")

USWords() = Array("color", "neighbor", "center", "behavior", _
   "meter", "thru")

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor

For O = 0 To 5
   Replace.SearchString = BritishWords(I)
   Replace.ReplaceString = USWords(I)
   Doc.replaceAll(Replace)
Next n
```

Les expressions de recherche et de remplacement sont définies à l'aide des propriétés `SearchString` et `ReplaceString` des objets `ReplaceDescriptor`. Le processus réel de remplacement est finalement implémenté à l'aide de la méthode `replaceAll` de l'objet Document, qui remplace toutes les occurrences de l'expression recherchée.

Exemple : recherche et remplacement de texte avec des expressions régulières

La fonction de remplacement de StarOffice est particulièrement efficace lorsqu'elle est associée à des expressions régulières. Ces derniers permettent de définir une expression de recherche variable avec des substituants et des caractères spéciaux au lieu d'une valeur fixe.

Les expressions régulières prises en charge par StarOffice sont décrites dans l'aide en ligne de StarOffice. Voici quelques exemples :

- Un point à l'intérieur d'une expression à rechercher remplace tout caractère. L'expression rais.n peut ainsi s'appliquer à la fois à `raison` et à `raisin`.
- Le caractère ^ marque le début d'un paragraphe. Toutes les occurrences du nom `Pierre` situées au début d'un paragraphe peuvent ainsi être trouvées à l'aide de l'expression de recherche `^Pierre`.
- Le caractère $ marque la fin d'un paragraphe. Toutes les occurrences du nom `Pierre` situées à la fin d'un paragraphe peuvent ainsi être trouvées à l'aide de l'expression de recherche `Pierre$`.
- Un astérisque (*) indique que le caractère précédent peut être répété un nombre de fois quelconque. Il peut être combiné avec le point en tant que substituant de tout caractère. L'expression `tempér.*e` peut par exemple, permettre de rechercher les expressions `tempérance` et `température`.

L'exemple suivant montre comment toutes les lignes vides d'un document texte peuvent être supprimées à l'aide d'expressions régulières ^$:

```
Dim Doc As Object
Dim Replace As Object
Dim I As Long

Doc = StarDesktop.CurrentComponent
Replace = Doc.createReplaceDescriptor

Replace.SearchRegularExpression = True
Replace.SearchString = "^$"
Replace.ReplaceString = ""

Doc.replaceAll(Replace)
```

Documents texte : plus qu'un simple mot

Jusqu'à présent, ce chapitre ne traitait que des paragraphes de texte et de portions de paragraphe. Mais les documents texte peuvent également contenir d'autres objets. Il peut s'agir de tableaux, de dessins, de champs de texte et de répertoires. Tous ces objets peuvent être ancrés à un point quelconque du texte.

Grâce à ces fonctions communes, tous ces objets de StarOffice prennent en charge un service de base commun appelé `com.sun.star.text.TextContent`. Ce dernier fournit les propriétés suivantes :

- **AnchorType (Enum)** : détermine le type d'ancre de l'objet `TextContent` (valeurs par défaut selon l'énumération `com.sun.star.text.TextContentAnchorType`).

- **AnchorTypes (sequence of Enum)** : énumération de tous les `AnchorTypes` qui prennent en charge un objet `TextContent` spécial.
- **TextWrap (Enum)** : détermine le type d'habillage d'un objet `TextContent` (valeurs par défaut selon l'énumération `com.sun.star.text.WrapTextMode`).

Les objets `TextContent` partagent également certaines méthodes, notamment celles permettant de créer, d'insérer et de supprimer des objets.

- Un nouvel objet `TextContent` est **créé** à l'aide de la méthode `createInstance` de l'objet Document.
- Un objet est **inséré** en utilisant la méthode `insertTextContent` de l'objet Text.
- Les objets `TextContent` sont **supprimés** à l'aide de la méthode `removeTextContent`.

Ces méthodes sont utilisées dans les exemples des sections suivantes.

Tableaux

L'exemple suivant crée un tableau à l'aide de la méthode createInstance décrite plus haut.

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

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

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
```

Lorsqu'il est créé, son nombre de lignes et de colonnes est défini à l'aide d'un appel `initialize`, puis inséré dans le document texte à l'aide de `insertTextContent`.

Comme le montre cet exemple, la méthode `insertTextContent` attend non seulement l'insertion de l'objet `Content`, mais également celle de deux autres paramètres :

- un objet `Cursor` déterminant la position d'insertion ;
- une variable de type Boolean indiquant si l'objet `Content` doit remplacer la sélection active du curseur (valeur `True`) ou être inséré dans le texte devant la sélection active (`False`).

Remarque – Lors de la création et de l'insertion de tableaux dans un document texte, les objets utilisés dans StarOffice Basic sont similaires à ceux disponibles dans VBA : l'objet Document et un objet `TextCursor` dans StarOffice Basic ou son équivalent `Range` dans VBA. Dans VBA, la méthode `Document.Tables.Add` gère la création et le paramétrage du tableau ; dans StarOffice Basic, celui-ci est créé à l'aide de la méthode `createInstance` puis initialisé et inséré dans le document via `insertTextContent`, comme dans l'exemple ci-dessus.

Il est possible de déterminer les tableaux insérés dans un document texte à l'aide d'une simple boucle. On utilise pour cela la méthode `getTextTables()` de l'objet Text document :

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim I As Integer
Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()
For I = 0 to TextTables.count - 1

   Table = TextTables(I)
   ' Editing table
Next I
```

Remarque – Les tableaux de texte sont disponibles dans StarOffice 8 via la liste `TextTables` de l'objet Document. Celle-ci remplace la liste de tableaux fournie précédemment dans l'objet `Selection`. L'exemple précédent montre comment créer un tableau de texte. Les options permettant d'accéder aux tableaux de texte sont décrites dans la section suivante.

Édition des tableaux

Un tableau est constitué de cellules individuelles. Ces cellules peuvent à leur tour contenir diverses autres cellules. À strictement parler, les colonnes de tableau n'existent pas dans StarOffice. Elles sont produites implicitement par la juxtaposition des cellules l'une en dessous de l'autre. Pour simplifier l'accès aux tableaux, StarOffice fournit néanmoins certaines méthodes fonctionnant avec des colonnes. Elles peuvent être utiles si aucune cellule du tableau n'a été fusionnée.

Commençons par examiner les propriétés du tableau lui-même. Elles sont définies dans le service `com.sun.star.text.TextTable`. Les propriétés les plus importantes de l'objet Table sont les suivantes :

- **BackColor (Long)** : couleur d'arrière-plan du tableau.
- **BottomMargin (Long)** : marge inférieure en centièmes de millimètre.

- **LeftMargin (Long)** : marge gauche en centièmes de millimètre.
- **RightMargin (Long)** : marge droite en centièmes de millimètre.
- **TopMargin (Long)** : marge supérieure en centièmes de millimètre.
- **RepeatHeadline (Boolean)** : répétition ou non de l'en-tête du tableau sur chaque page.
- **Width (Long)** : largeur absolue du tableau en centièmes de millimètre.

Lignes

Un tableau est constitué d'une liste de lignes. L'exemple suivant montre comment accéder aux lignes d'un tableau et le formater.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim Row As Object
Dim I As Integer
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)
Rows = Table.getRows
For I = 0 To Rows.getCount() - 1
   Row = Rows.getByIndex(I)
   Row.BackColor = &HFF00FF
Next
```

Cet exemple commence par créer une liste contenant toutes les lignes utilisant un appel `Table.getRows`. Les méthodes `getCount` et `getByIndex` permettent de poursuivre le traitement de la liste et appartiennent à l'interface `com.sun.star.table.XtableRows`. La méthode `getByIndex` renvoie un objet Row, prenant en charge le service `com.sun.star.text.TextTableRow`.

Les méthodes centrales de l'interface `com.sun.star.table.XtableRows` sont les suivantes :

- **getByIndex(Integer)** : renvoie un objet Row pour l'indice spécifié.
- **getCount()** : renvoie le nombre d'objets Row.
- **insertByIndex(Index, Count)** : insère Count lignes dans le tableau à partir de la position `Index`.
- **removeByIndex(Index, Count)** : supprime Count lignes du tableau à partir de la position `Index`.

Alors que les méthodes `getByIndex` et `getCount` sont disponibles dans tous les tableaux, les méthodes `insertByIndex` et `removeByIndex` ne peuvent être utilisées que dans les tableaux qui ne contiennent pas de cellules fusionnées.

Le service com.sun.star.text.TextTableRow fournit les propriétés suivantes :

- **BackColor (Long)** : couleur d'arrière-plan de la ligne.
- **Height (Long)** : hauteur de la ligne en centièmes de millimètre.
- **IsAutoHeight (Boolean)** : la hauteur de la ligne s'adapte automatiquement au contenu.
- **VertOrient (const)** : orientation verticale du cadre texte – détails sur l'orientation verticale du texte à l'intérieur du tableau (valeurs selon com.sun.star.text.VertOrientation).

Colonnes

Les colonnes sont accessibles de la même manière que les lignes, à l'aide des méthodes getByIndex, getCount, insertByIndex et removeByIndex sur l'objet Column, qui est atteint via getColumns. Ces méthodes ne peuvent néanmoins être utilisées que dans des tableaux ne contenant pas de cellules fusionnées. Les cellules ne peuvent pas être formatées par colonne dans StarOffice Basic. Pour arriver à ce résultat, il faut appliquer les méthodes de formatage des cellules individuelles.

Cellules

Chaque cellule d'un document StarOffice porte un nom unique. Si le curseur de StarOffice se trouve dans une cellule, le nom de cette cellule apparaît dans la barre d'état. La cellule supérieure gauche est habituellement appelée A1 et la cellule inférieure droite Xn, où X représente la lettre de la dernière colonne et n le numéro de la dernière ligne. Les objets Cell sont accessibles par la méthode getCellByName() de l'objet Table. L'exemple suivant montre une boucle qui parcourt toutes les cellules d'un tableau et y insère les numéros de ligne et lettres de colonne correspondants.

```
Dim Doc As Object
Dim Table As Object
Dim Cursor As Object
Dim Rows As Object
Dim RowIndex As Integer
Dim Cols As Object
Dim ColIndex As Integer
Dim CellName As String
Dim Cell As Object

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

Table = Doc.createInstance("com.sun.star.text.TextTable")
Table.initialize(5, 4)

Doc.Text.insertTextContent(Cursor, Table, False)

Rows = Table.getRows
```

```
Cols = Table.getColumns

For RowIndex = 1 To Rows.getCount()
   For ColIndex = 1 To Cols.getCount()
      CellName = Chr(64 + ColIndex) & RowIndex
      Cell = Table.getCellByName(CellName)
      Cell.String = "row: " & CStr(RowIndex) + ", column: " & CStr(ColIndex)
   Next
Next
```

Une cellule de tableau est comparable à du texte standard. Elle prend en charge l'interface `createTextCursor` permettant la création d'un objet `TextCursor` associé.

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

Toutes les options de formatage pour les caractères individuels et les paragraphes sont donc automatiquement disponibles.

L'exemple qui suit effectue une recherche dans tous les tableaux d'un document texte et aligne à droite toutes les cellules comportant des valeurs numériques à l'aide de la propriété de paragraphe correspondante.

```
Dim Doc As Object
Dim TextTables As Object
Dim Table As Object
Dim CellNames
Dim Cell As Object
Dim CellCursor As Object
Dim I As Integer
Dim J As Integer

Doc = StarDesktop.CurrentComponent
TextTables = Doc.getTextTables()

For I = 0 to TextTables.count - 1
   Table = TextTables(I)
   CellNames = Table.getCellNames()

   For J = 0 to UBound(CellNames)
      Cell = Table.getCellByName(CellNames(J))
      If IsNumeric(Cell.String) Then
         CellCursor = Cell.createTextCursor()
         CellCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.RIGHT
      End If
   Next
Next
```

Cet exemple crée une liste `TextTables` contenant tous les tableaux d'un texte, qui sont parcourus dans une boucle. StarOffice crée ensuite une liste des noms des cellules associés pour chacun de ces tableaux. Ces cellules sont ensuite parcourues à leur tour dans une boucle. Si une cellule contient une valeur numérique, l'exemple modifie le formatage en conséquence. Pour ce faire, il commence par créer un objet `TextCursor` renvoyant au contenu de la cellule, puis modifie les propriétés de paragraphe de cette dernière.

Cadres texte

Les cadres texte sont considérés comme des objets `TextContent`, tout comme les tableaux et les images. Ils consistent généralement en texte standard, mais ils peuvent être placés à tout emplacement sur la page et ne sont pas inclus dans l'enchaînement.

Comme pour tous les objets `TextContent`, la création d'un cadre texte et son insertion dans le document se font à deux étapes différentes.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")
Doc.Text.insertTextContent(Cursor, Frame, False)
```

Le cadre texte est créé à l'aide de la méthode `createInstance` de l'objet Document. Le cadre texte créé de cette manière peut ensuite être inséré dans le document à l'aide de la méthode `insertTextContent` de l'objet `Text`. Pour cela, le nom du service approprié `com.sun.star.text.TextFrame` doit être spécifié.

L'emplacement d'insertion du cadre texte est déterminé par un objet `Cursor`, également exécuté lors de son insertion.

Remarque – Les cadres texte sont l'équivalent dans StarOffice des cadres de position utilisés dans Word. VBA utilise la méthode `Document.Frames.Add` à cet effet, tandis que dans StarOffice Basic, la création est effectuée via la procédure ci-dessus, à l'aide d'un objet `TextCursor` et de la méthode `createInstance` de l'objet Document.

Les objets TextFrame fournissent toute une gamme de propriétés permettant d'influer sur l'emplacement et le comportement du cadre. La plupart de ces propriétés sont définies dans le service `com.sun.star.text.BaseFrameProperties`, également pris en charge par chaque service `TextFrame`. Les propriétés centrales sont les suivantes :

- **BackColor (Long)** : couleur d'arrière-plan du cadre texte.
- **BottomMargin (Long)** : marge inférieure en centièmes de millimètre.
- **LeftMargin (Long)** : marge gauche en centièmes de millimètre.
- **RightMargin (Long)** : marge droite en centièmes de millimètre.
- **TopMargin (Long)** : marge supérieure en centièmes de millimètre.
- **Height (Long)** : hauteur du cadre texte en centièmes de millimètre.
- **Width (Long)** : largeur du cadre texte en centièmes de millimètre.
- **HoriOrient (const)** : orientation horizontale du cadre texte (selon `com.sun.star.text.HoriOrientation`).

- **VertOrient (const)** : orientation verticale du cadre texte (selon `com.sun.star.text.VertOrientation`).

L'exemple suivant crée un cadre texte à l'aide des propriétés ci-dessus :

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object

Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Cursor.gotoNextWord(False)
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000
Frame.AnchorType = com.sun.star.text.TextContentAnchorType.AS_CHARACTER
Frame.TopMargin = 0
Frame.BottomMargin = 0
Frame.LeftMargin = 0
Frame.RightMargin = 0
Frame.BorderDistance = 0
Frame.HoriOrient = com.sun.star.text.HoriOrientation.NONE
Frame.VertOrient = com.sun.star.text.VertOrientation.LINE_TOP

Doc.Text.insertTextContent(Cursor, Frame, False)
```

Cet exemple crée un objet `TextCursor` servant de marque d'insertion pour le cadre texte. Ce dernier est positionné entre le premier et le deuxième mot du texte. Le cadre texte est créé à l'aide de `Doc.createInstance`. Les propriétés des objets Text frame sont définies aux valeurs initiales nécessaires.

Il convient de noter ici l'interaction entre les propriétés `AnchorType` (du service `TextContent`) et `VertOrient` (du service `BaseFrameProperties`). `AnchorType` reçoit la valeur `AS_CHARACTER`. Le cadre texte est donc inséré directement dans l'enchaînement et se comporte comme un caractère. Il peut ainsi être déplacé à la ligne suivante en cas de retour à la ligne, par exemple. La valeur `LINE_TOP` de la propriété `VertOrient` garantit que le bord supérieur du cadre texte se trouve à la même hauteur que le bord supérieur du caractère.

Lorsque l'initialisation est terminée, le cadre texte est finalement inséré dans le document texte à l'aide d'un appel de la méthode `insertTextContent`.

Pour éditer le contenu d'un cadre texte, l'utilisateur fait appel à l'objet `TextCursor`, déjà mentionné à de nombreuses reprises et également disponible pour les cadres texte.

```
Dim Doc As Object
Dim TextTables As Object
Dim Cursor As Object
Dim Frame As Object
Dim FrameCursor As Object
```

```
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()
Frame = Doc.createInstance("com.sun.star.text.TextFrame")

Frame.Width = 3000
Frame.Height = 1000

Doc.Text.insertTextContent(Cursor, Frame, False)

FrameCursor = Frame.createTextCursor()
FrameCursor.charWeight = com.sun.star.awt.FontWeight.BOLD
FrameCursor.paraAdjust = com.sun.star.style.ParagraphAdjust.CENTER
FrameCursor.String = "This is a small Test!"
```

Cet exemple crée un cadre texte, l'insère dans le document actif et ouvre un objet `TextCursor` pour le cadre. Ce curseur est utilisé pour définir en gras la police du cadre et centrer le paragraphe. Enfin, la chaîne 'This is a small Test!' est assignée au cadre texte.

Champs de texte

Les champs de texte sont des objets `TextContent`, car ils ajoutent une extension logique au-delà du texte pur. Des champs de texte peuvent être insérés dans un document texte en utilisant les mêmes méthodes que pour les autres objets `TextContent` :

```
Dim Doc As Object
Dim DateTimeField As Object
Dim Cursor As Object
Doc = StarDesktop.CurrentComponent
Cursor = Doc.Text.createTextCursor()

DateTimeField = Doc.createInstance("com.sun.star.text.TextField.DateTime")
DateTimeField.IsFixed = False
DateTimeField.IsDate = True
Doc.Text.insertTextContent(Cursor, DateTimeField, False)
```

Cet exemple insère un champ de texte correspondant à la date du jour au début du document texte actif. Si la propriété `IsDate` a la valeur `True`, seule la date s'affiche, sans l'heure. Si `IsFixed` a la valeur `False`, la date est actualisée automatiquement à l'ouverture du document.

Remarque – Dans VBA, le type d'un champ est spécifié par un paramètre de la méthode `Document.Fields.Add` ; dans StarOffice Basic, il est défini par le nom du service prenant en charge ce type de champ.

Dans les anciennes versions de StarOffice, on accédait aux champs de texte par tout un ensemble de méthodes disponibles dans l'objet `Selection` (`InsertField`, `DeleteUserField`, `SetCurField`, par exemple).

Dans StarOffice 8, les champs sont administrés à l'aide d'un concept orienté objet. Pour créer un champ de texte particulier, il faut d'abord créer un champ de texte du type requis et l'initialiser avec les propriétés adéquates. Ce champ de texte est ensuite inséré dans le document à l'aide de la méthode `insertTextContent`. Un texte source correspondant est illustré dans l'exemple précédent. Les principaux types de champ et leurs propriétés sont décrits dans les sections suivantes.

Outre l'insertion de champs de texte, la recherche des champs dans un document peut également être une tâche fastidieuse. L'exemple suivant montre comment parcourir tous les champs de texte d'un document texte dans une boucle et contrôler leur type.

```
Dim Doc As Object
Dim TextFieldEnum As Object
Dim TextField As Object
Dim I As Integer

Doc = StarDesktop.CurrentComponent

TextFieldEnum = Doc.getTextFields.createEnumeration

While TextFieldEnum.hasMoreElements()

   TextField = TextFieldEnum.nextElement()

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

Wend
```

La détermination des champs de texte présents se base sur la liste `TextFields` de l'objet Document. Dans l'exemple, un objet `Enumeration` est créé en fonction de cette liste. Il permet d'interroger successivement tous les champs de texte dans une boucle. À l'aide de la méthode `supportsService`, l'exemple vérifie ensuite le service pris en charge par chaque champ de texte trouvé. Si le champ s'avère être un champ de date/heure ou une annotation, le type de champ correspondant s'affiche dans une boîte d'information. S'il s'agit d'un autre champ, l'exemple affiche l'information 'unknown'.

La liste ci-dessous répertorie les champs de texte les plus importants et leurs propriétés associées. Une liste complète de tous les champs de texte est fournie dans le module `com.sun.star.text.TextField` de la référence de l'API. (Lorsque vous répertoriez le nom du service d'un champ de texte, des majuscules et des minuscules doivent être utilisées dans StarOffice Basic, comme dans l'exemple précédent.)

Nombre de pages, de mots et de caractères

Les champs de texte :

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

renvoient respectivement le nombre de pages, de mots ou de caractères d'un texte. Ils prennent en charge la propriété suivante :

- **NumberingType (const)** : format de numérotation (constantes selon `com.sun.star.style.NumberingType`).

Page active

Le numéro de la page active peut être inséré dans un document à l'aide du champ de texte `com.sun.star.text.TextField.PageNumber`. Vous pouvez spécifier les propriétés suivantes :

- **NumberingType (const)** : format de numérotation (constantes selon `com.sun.star.style.NumberingType`).
- **Offset (short)** : décalage ajouté au nombre de pages (ce nombre peut être négatif).

L'exemple qui suit montre comment insérer le nombre de pages dans le pied de page d'un document.

```
Dim Doc As Object
Dim DateTimeField As Object
Dim PageStyles As Object
Dim StdPage As Object
Dim FooterCursor As Object
Dim PageNumber As Object

Doc = StarDesktop.CurrentComponent

PageNumber = Doc.createInstance("com.sun.star.text.TextField.PageNumber")
PageNumber.NumberingType = com.sun.star.style.NumberingType.ARABIC

PageStyles = Doc.StyleFamilies.getByName("PageStyles")

StdPage = PageStyles("Default")
StdPage.FooterIsOn = True

FooterCursor = StdPage.FooterTextLeft.Text.createTextCursor()
StdPage.FooterTextLeft.Text.insertTextContent(FooterCursor, PageNumber, False)
```

L'exemple commence par créer un champ de texte prenant en charge le service `com.sun.star.text.TextField.PageNumber`. L'en-tête et le pied de page étant définies à l'intérieur des modèles de page de StarOffice, elles sont initialement établies à l'aide de la liste de tous les objets `PageStyles`.

Pour garantir que la ligne du pied de page est visible, la propriété `FooterIsOn` a la valeur `True`. Le champ de texte est alors inséré dans le document à l'aide de l'objet Text associé de la ligne de pied de page gauche.

Annotations

Dans le texte, les champs d'annotation (`com.sun.star.text.TextField.Annotation`) sont repérables à un petit symbole jaune. Si vous cliquez sur ce symbole, un champ de texte s'ouvre pour vous permettre d'enregistrer un commentaire à cet emplacement du texte. Un champ d'annotation comporte les propriétés suivantes.

- **Author (String)** : nom de l'auteur.
- **Content (String)** : texte du commentaire.
- **Date (Date)** : date à laquelle a été écrite l'annotation.

Date / heure

Un champ de date/heure (`com.sun.star.text.TextField.DateTime`) représente la date du jour ou l'heure qu'il est. Il prend en charge les propriétés suivantes :

- **IsFixed (Boolean)** : si cette propriété a la valeur `True`, l'heure de l'insertion reste inchangée ; si elle a la valeur `False`, l'heure de l'insertion est actualisée à chaque ouverture du document.
- **IsDate (Boolean)** : si cette propriété a la valeur `True`, le champ affiche la date du jour, sinon il affiche l'heure actuelle.
- **DateTimeValue (struct)** : contenu actuel du champ (structure `com.sun.star.util.DateTime`).
- **NumberFormat (const)** : format dans lequel la date ou l'heure sont décrites.

Nom / Numéro de chapitre

Le nom du chapitre actif est accessible par un champ de texte du type `com.sun.star.text.TextField.Chapter`. Sa forme peut être définie à l'aide de deux propriétés.

- **ChapterFormat (const)** : définit si l'élément représenté est le nom du chapitre ou son numéro (selon `com.sun.star.text.ChapterFormat`).
- **Level (Integer)** : détermine le niveau du chapitre dont le nom et/ou le numéro est affiché. La valeur 0 représente le plus haut niveau disponible.

Repères de texte

Les repères de texte (Service `com.sun.star.text.Bookmark`) sont des objets `TextContent`. Leur création et leur insertion fait appel à un concept déjà décrit plus haut :

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

Doc = StarDesktop.CurrentComponent

Cursor = Doc.Text.createTextCursor()

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

L'exemple crée un objet `Cursor`, qui marque la position d'insertion du repère de texte, puis crée l'objet repère de texte proprement dit (`Bookmark`). Un nom est ensuite assigné au repère de texte et ce dernier est inséré à l'emplacement du curseur dans le document, par la méthode `insertTextContent`.

Les repères de texte d'un texte sont accessibles par une liste appelée `Bookmarks`. Il est possible d'accéder aux repères de texte par leur numéro ou par leur nom.

L'exemple qui suit montre comment trouver un repère de texte à l'intérieur d'un texte et insérer du texte à son emplacement.

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

Doc = StarDesktop.CurrentComponent

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

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

Dans cet exemple, la méthode `getByName` est utilisée pour trouver le repère de texte requis au moyen de son nom. L'appel `createTextCursorByRange` crée ensuite un objet `Cursor`, qui est placé à la position d'ancrage du repère de texte. Le curseur insère ensuite le texte requis à cet endroit.

CHAPITRE 7

Classeurs

StarOffice Basic fournit une interface complète pour la création et l'édition de classeurs commandées par programme. Ce chapitre décrit comment piloter les services, méthodes et propriétés concernés des classeurs.

La première section aborde la structure de base des classeurs et indique comment accéder au contenu de chaque cellule pour l'éditer.

La seconde section, axée sur les zones de cellules et sur les options de recherche et de remplacement du contenu des cellules, explique comment éditer efficacement les classeurs.

Remarque – L'objet `Range`, qui permet d'accéder à toutes les zones d'une table, a été étendu dans la nouvelle API.

Structure des documents basés sur des tables (classeurs)

L'objet Document d'un classeur est basé sur le service `com.sun.star.sheet.SpreadsheetDocument`. Chaque document de ce type peut comporter plusieurs feuilles de calcul. Dans ce guide, un *document basé sur des tables* ou *classeur* désigne le document entier, tandis qu'une *feuille de calcul* (ou, de façon abrégée, une *feuille*) est une feuille (table) du document.

Remarque – La terminologie concernant les feuilles de calcul et leur contenu est différente dans VBA et dans StarOffice Basic. Alors que l'objet Document est appelé *Workbook* dans VBA et les pages qu'il contient *Worksheets*, ces objets sont respectivement appelés *SpreadsheetDocument* et *Sheet* dans StarOffice Basic.

Classeurs

Vous pouvez accéder à chaque feuille de calcul d'un classeur via la liste `Sheets`.

Les exemples qui suivent indiquent comment accéder à une feuille par son numéro ou par son nom.

Exemple 1 : accès par le numéro (la numérotation commence à 0)

```
Dim Doc As Object
Dim Sheet As Object

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

Exemple 2 : accès par le nom

```
Dim Doc As Object
Dim Sheet As Object

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

Dans le premier exemple, on accède à la feuille par son numéro (la numérotation commençant à 0). Dans le second, on y accède par son nom et par la méthode `getByName`.

L'objet `Sheet` obtenu par la méthode `getByName` prend en charge le service `com.sun.star.sheet.Spreadsheet`. Outre les différentes interfaces qu'il propose pour l'édition du contenu, ce service fournit les propriétés suivantes :

- **IsVisible (Boolean)** : la feuille est visible.
- **PageStyle (String)** : nom du modèle de page de la feuille.

Création, suppression et attribution de nom à des feuilles

La liste `Sheets` d'un `classeur` permet également de créer, de supprimer et de renommer des feuilles individuelles. L'exemple ci-dessous utilise la méthode `hasByName` pour vérifier si une feuille nommée *MySheet* existe. Si c'est le cas, la méthode détermine une référence d'objet correspondante à l'aide de la méthode `getByName`, puis enregistre cette référence dans une variable de `Sheet`. Si la feuille correspondante n'existe pas, elle est créée par l'appel `createInstance` et insérée dans le classeur par la méthode `insertByName`.

```
Dim Doc As Object
Dim Sheet As Object

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

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

Les méthodes `getByName` et `insertByName` proviennent de l'interface `com.sun.star.container.XnameContainer`, dont vous trouverez une description dans le Chapitre 4.

Lignes et colonnes

Chaque feuille contient une liste de ses propres lignes et colonnes. Ces listes sont disponibles via les propriétés `Rows` et `Columns` de l'objet Spreadsheet et prennent en charge les services `com.sun.star.table.TableColumns` et/ou `com.sun.star.table.TableRows`.

L'exemple suivant crée deux objets qui renvoient à la première ligne et à la première colonne d'une feuille, et stocke les références dans les variables d'objets `FirstCol` et `FirstRow`.

```
Dim Doc As Object
Dim Sheet As Object
Dim FirstRow As Object
Dim FirstCol As Object

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

FirstCol = Sheet.Columns(0)
FirstRow = Sheet.Rows(0)
```

Les objets Column prennent en charge le service `com.sun.star.table.TableColumn`, dont les propriétés sont les suivantes :

- **Width (long)** : largeur d'une colonne en centièmes de millimètre.
- **OptimalWidth (Boolean)** : définit la largeur optimale d'une colonne.
- **IsVisible (Boolean)** : affiche une colonne.
- **IsStartOfNewPage (Boolean)** : insère un saut de page avant une colonne lors de l'impression.

Pour obtenir une largeur de colonne optimale, la propriété `OptimalWidth` doit être définie sur `True`. Il est possible de modifier la largeur d'une cellule individuelle sans modifier la largeur de la colonne qui la contient. Sur un plan pratique, `OptimalWidth` est davantage une méthode qu'une propriété.

Les objets Row sont basés sur le service `com.sun.star.table.RowColumn`, dont les propriétés sont les suivantes :

- **Height (long)** : hauteur de la ligne en centièmes de millimètre.
- **OptimalHeight (Boolean)** : définit la hauteur optimale de la ligne.
- **IsVisible (Boolean)** : affiche une ligne.
- **IsStartOfNewPage (Boolean)** : insère un saut de page avant une ligne lors de l'impression.

Si la propriété `OptimalHeight` d'une ligne est définie sur `True`, sa hauteur change automatiquement lorsque celle d'une cellule qu'elle contient est modifiée. L'optimisation automatique se poursuit jusqu'à ce qu'une hauteur absolue soit assignée à la ligne par la propriété `Height`.

L'exemple suivant active l'optimisation automatique de la hauteur de ligne pour les cinq premières lignes de la feuille et rend la deuxième colonne invisible.

```
Dim Doc As Object
Dim Sheet As Object
Dim Row As Object
Dim Col As Object
Dim I As Integer

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

For I = 0 To 4
   Row = Sheet.Rows(I)
   Row.OptimalHeight = True
Next I

Col = Sheet.Columns(1)
Col.IsVisible = False
```

Remarque – Vous pouvez accéder aux listes `Rows` et `Columns` via un index dans StarOffice Basic. Contrairement à VBA, l'index de la première colonne est 0 (et non 1).

Insertion et suppression de lignes et de colonnes

Les objets `Rows` et `Columns` d'une feuille peuvent accéder à des lignes et colonnes existantes, mais également en insérer et en supprimer.

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

```
Dim NewColumn As Object

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

Sheet.Columns.insertByIndex(3, 1)
Sheet.Columns.removeByIndex(5, 1)
```

Cet exemple utilise la méthode `insertByIndex` pour insérer une nouvelle colonne à l'emplacement de la quatrième colonne dans la feuille (index 3, la numérotation commençant à 0). Le second paramètre spécifie le nombre de colonnes à insérer (dans cet exemple, une).

La méthode `removeByIndex` supprime la sixième colonne (index 5). Là aussi, le second paramètre spécifie le nombre de colonnes à supprimer.

Les méthodes d'insertion et de suppression de lignes utilisent la fonction de l'objet `Rows` de la même manière que les méthodes d'édition des colonnes utilisent l'objet `Columns`.

Cellules

Une feuille de calcul est constituée d'une liste de cellules à deux dimensions. Chaque cellule est définie par ses positions X et Y par rapport à la cellule supérieure gauche, dont la position est (0,0).

L'exemple suivant crée un objet qui renvoie à la cellule supérieure gauche et insère du texte dans cette cellule :

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

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

Cell = Sheet.getCellByPosition(0, 0)
Cell.String = "Test"
```

Outre ses coordonnées numériques, chaque cellule d'une feuille porte un nom : par exemple, la cellule supérieure gauche (0,0) d'une feuille est nommée `A1`. La lettre `A` représente la colonne et le chiffre `1` la ligne. Il convient de ne pas confondre le *nom* et la *position* d'une cellule, car le comptage des lignes commence à 1 pour les noms et à 0 pour les positions.

Dans StarOffice, une cellule peut être vide, ou bien contenir du texte, des nombres ou des formules. Le type de cellule n'est pas déterminé par le contenu qui y est enregistré, mais par la propriété de l'objet utilisé pour son entrée. Les nombres peuvent être insérés et appelés à l'aide de la propriété `Value`, le texte à l'aide de la propriété `String` et les formules à l'aide de la propriété `Formula`.

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

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

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = "Test"

Cell = Sheet.getCellByPosition(0, 2)
Cell.Formula = "=A1"
```

Cet exemple insère un nombre, du texte et une formule dans les champs A1, A2 et A3.

Remarque – Les propriétés `Value`, `String` et `Formula` remplacent la méthode `PutCell` pour la définition des valeurs d'une cellule de tableau.

StarOffice utilise la propriété `String` comme du texte pour traiter le contenu saisi dans les cellules, même si ce contenu est un nombre. Les nombres sont alignés à gauche dans la cellule et non pas à droite. Lors de l'utilisation de formules, il convient également de différencier le texte et les nombres :

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

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

Cell = Sheet.getCellByPosition(0, 0)
Cell.Value = 100

Cell = Sheet.getCellByPosition(0, 1)
Cell.String = 1000

Cell = Sheet.getCellByPosition(0, 2)
Cell.Formula = "=A1+A2"

MsgBox Cell.Value
```

Bien que la cellule A1 contienne la valeur 100 et la cellule A2 la valeur 1 000, la formule A1+A2 renvoie la valeur 100. En effet, le contenu de la cellule A2 a été saisi comme une chaîne et non comme un nombre.

Pour savoir si une cellule contient un nombre ou une chaîne, utilisez la propriété `Type` :

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

```
Dim Cell As Object

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

Cell.Value = 1000

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

La propriété `Cell.Type` renvoie une valeur pour l'énumération `com.sun.star.table.CellContentType` qui identifie le type de contenu d'une cellule. Les valeurs possibles sont les suivantes :

- **EMPTY** : aucune valeur.
- **VALUE** : nombre.
- **TEXT** : chaîne.
- **FORMULA** : formule.

Insertion, suppression, copie et déplacement de cellules

Outre la modification directe du contenu d'une cellule, StarOffice Calc offre également une interface permettant d'insérer, de supprimer, de copier ou de fusionner des cellules. L'interface `com.sun.star.sheet.XRangeMovement`, accessible par l'objet Spreadsheet, fournit quatre méthodes de modification du contenu des cellules.

La méthode `insertCell` sert à insérer des cellules dans une feuille.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

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

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.insertCells(CellRangeAddress, com.sun.star.sheet.CellInsertMode.DOWN)
```

Cet exemple insère une plage de cellules comportant deux lignes sur deux colonnes dans la deuxième colonne et la deuxième ligne (portant toutes deux le numéro 1) de la première feuille (numéro 0) du classeur. Toutes les valeurs existantes de la plage de cellules spécifiée sont déplacées sous cette dernière.

Pour définir la plage de cellules à insérer, utilisez la structure `com.sun.star.table.CellRangeAddress`. Cette structure comporte les valeurs suivantes :

- **Sheet (short)** : numéro de la feuille (la numérotation commençant à 0).
- **StartColumn (long)** : première colonne de la plage de cellules (la numérotation commençant à 0).
- **StartRow (long)** : première ligne de la plage de cellules (la numérotation commençant à 0).
- **EndColumn (long)** : dernière colonne de la plage de cellules (la numérotation commençant à 0).
- **EndRow (long)** : dernière ligne de la plage de cellules (la numérotation commençant à 0).

La structure `CellRangeAddress` complétée doit être passée comme premier paramètre à la méthode `insertCells`. Le second paramètre de `insertCells` contient une valeur de l'énumération `com.sun.star.sheet.CellInsertMode` et définit le traitement réservé aux valeurs situées devant la position d'insertion. L'énumération `CellInsertMode` reconnaît les valeurs suivantes :

- **NONE** : les valeurs actives conservent leur position actuelle.
- **DOWN** : les cellules situées au niveau de la position d'insertion et en dessous sont déplacées vers le bas.
- **RIGHT** : les cellules situées au niveau de la position d'insertion et à droite sont déplacées vers la droite.
- **ROWS** : les lignes après la position d'insertion sont déplacées vers le bas.
- **COLUMNS** : les colonnes après la position d'insertion sont déplacées vers la droite.

La méthode `removeRange` est l'équivalent de la méthode `insertCells`. Cette méthode supprime de la feuille la plage de cellules définie dans la structure `CellRangeAddress`.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress

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

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
```

```
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

Sheet.removeRange(CellRangeAddress, com.sun.star.sheet.CellDeleteMode.UP)
```

Cet exemple supprime la plage de cellules `B2:C3` de la feuille et déplace de deux colonnes vers le haut les cellules situées en dessous. Ce type de suppression est défini par l'une des valeurs suivantes provenant de l'énumération `com.sun.star.sheet.CellDeleteMode` :

- **NONE** : les valeurs actives conservent leur position actuelle.
- **UP** : les cellules situées au niveau de la position d'insertion et en dessous sont déplacées vers le haut.
- **LEFT** : les cellules situées au niveau de la position d'insertion et à droite sont déplacées vers la gauche.
- **ROWS** : les lignes après la position d'insertion sont déplacées vers le haut.
- **COLUMNS** : les colonnes après la position d'insertion sont déplacées vers la gauche.

L'interface `XRangeMovement` comporte deux méthodes supplémentaires pour déplacer (`moveRange`) ou copier (`copyRange`) les plages de cellules. L'exemple suivant déplace la plage `B2:C3` de façon à ce qu'elle commence à la position `A6` :

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRangeAddress As New com.sun.star.table.CellRangeAddress
Dim CellAddress As New com.sun.star.table.CellAddress

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

CellRangeAddress.Sheet = 0
CellRangeAddress.StartColumn = 1
CellRangeAddress.StartRow = 1
CellRangeAddress.EndColumn = 2
CellRangeAddress.EndRow = 2

CellAddress.Sheet = 0
CellAddress.Column = 0
CellAddress.Row = 5

Sheet.moveRange(CellAddress, CellRangeAddress)
```

Outre la structure `CellRangeAdress`, la méthode `moveRange` attend une structure `com.sun.star.table.CellAddress` pour définir l'origine de la zone de destination du déplacement. La méthode `CellAddress` fournit les valeurs suivantes :

- **Sheet (short)** : numéro de la feuille (la numérotation commençant à 0).
- **Column (long)** : numéro de la colonne concernée (la numérotation commençant à 0).
- **Row (long)** : numéro de la ligne concernée (la numérotation commençant à 0).

Le contenu des cellules de la plage cible est toujours écrasé par la méthode `moveRange`. Contrairement à la méthode `InsertCells`, la méthode `removeRange` ne fournit pas de paramètre permettant des déplacements automatiques.

Les fonctions de la méthode `copyRange` sont identiques à celles de la méthode `moveRange`, si ce n'est que `copyRange` insère une copie de la plage de cellules au lieu de la déplacer.

Remarque – Du point de vue de leur fonction, les méthodes `insertCell`, `removeRange` et `copyRange` de StarOffice Basic sont comparables aux méthodes `Range.Insert`, `Range.Delete` et `Range.Copy` de VBA. Les méthodes sont toutefois appliquées à l'objet `Range` correspondant dans VBA, tandis que dans StarOffice Basic, elles sont appliquées à l'objet `Sheet` associé.

Formatage

Un classeur fournit des propriétés et des méthodes de formatage des cellules et des pages.

Propriétés de cellules

Il existe de nombreuses options de formatage des cellules, telles que la spécification du type de police et de la taille du texte. Chaque cellule prend en charge les services `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`, dont les principales propriétés sont décrites au Chapitre 6. Le formatage spécial des cellules est traité par le service `com.sun.star.table.CellProperties`. Les principales propriétés de ce service sont décrites dans les sections qui suivent.

Vous pouvez appliquer toutes les propriétés nommées à des cellules individuelles et à des plages de cellules.

Remarque – L'objet `CellProperties` de l'API StarOffice est comparable à l'objet `Interior` de VBA, qui définit également les propriétés propres aux cellules.

Couleur d'arrière-plan et ombres

Le service `com.sun.star.table.CellProperties` fournit les propriétés suivantes pour la définition des couleurs d'arrière-plan et des ombres :

- **CellBackColor (Long)** : couleur d'arrière-plan de la cellule du tableau.

- **IsCellBackgroundTransparent (Boolean)** : définit une couleur d'arrière-plan transparente.
- **ShadowFormat (struct)** : spécifie l'ombre à appliquer aux cellules (structure correspondant à `com.sun.star.table.ShadowFormat`).

La structure `com.sun.star.table.ShadowFormat` et les spécifications détaillées des ombres de cellule ont la structure suivante :

- **Location (enum)** : position de l'ombre (valeur provenant de la structure `com.sun.star.table.ShadowLocation`).
- **ShadowWidth (Short)** : taille de l'ombre en centièmes de millimètre.
- **IsTransparent (Boolean)** : définit une ombre transparente.
- **Color (Long)** : couleur de l'ombre.

L'exemple suivant inscrit le nombre 1 000 dans la cellule B2, applique la couleur rouge à l'arrière-plan à l'aide de la propriété `CellBackColor`, puis crée une ombre gris clair pour la cellule, qui est déplacée de 1 mm vers la gauche et vers le bas.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object
Dim ShadowFormat As New com.sun.star.table.ShadowFormat

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

Cell.Value = 1000

Cell.CellBackColor = RGB(255, 0, 0)

ShadowFormat.Location = com.sun.star.table.ShadowLocation.BOTTOM_RIGHT
ShadowFormat.ShadowWidth = 100
ShadowFormat.Color = RGB(160, 160, 160)

Cell.ShadowFormat = ShadowFormat
```

Justification

StarOffice fournit diverses fonctions permettant de modifier la justification d'un texte dans une cellule de tableau.

Les propriétés suivantes définissent la justification horizontale et verticale d'un texte :

- **HoriJustify (enum)** : justification horizontale du texte (valeur provenant de `com.sun.star.table.CellHoriJustify`).
- **VertJustify (enum)** : justification verticale du texte (valeur provenant de `com.sun.star.table.CellVertJustify`).
- **Orientation (enum)** : orientation du texte (valeur correspondant à `com.sun.star.table.CellOrientation`).

- **IsTextWrapped (Boolean)** : autorise les retours à la ligne automatiques à l'intérieur de la cellule.
- **RotateAngle (Long)** : angle de rotation du texte en centièmes de degré.

L'exemple suivant indique comment empiler le contenu d'une cellule pour que chaque caractère s'imprime l'un en dessous de l'autre dans le coin supérieur gauche de la cellule. Les caractères ne subissent aucune rotation.

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

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

Cell.Value = 1000

Cell.HoriJustify = com.sun.star.table.CellHoriJustify.LEFT
Cell.VertJustify = com.sun.star.table.CellVertJustify.TOP
Cell.Orientation = com.sun.star.table.CellOrientation.STACKED
```

Formats de nombre, de date et de texte

StarOffice propose plusieurs formats de date et d'heure prédéfinis. Chacun de ces formats comporte un numéro de série utilisé pour assigner le format correspondant aux cellules à l'aide de la propriété `NumberFormat`. StarOffice fournit les méthodes `queryKey` et `addNew` qui permettent d'accéder à des formats numériques existants et de créer ses propres formats numériques. Ces méthodes sont accessibles par l'appel d'objet suivant :

```
NumberFormats = Doc.NumberFormats
```

Un format se spécifie à l'aide d'une chaîne de format structurée de la même façon que la fonction de format de StarOffice Basic. Il existe toutefois une différence importante : alors que le format de commande requiert des abréviations, symboles décimaux et caractères séparateurs de milliers anglais, les abréviations propres au pays doivent être utilisées dans la structure d'un format de commande pour l'objet `NumberFormats`.

L'exemple suivant formate la cellule B2 pour que les nombres s'affichent avec trois décimales et qu'ils utilisent les virgules comme séparateurs de milliers.

```
Dim Doc As Object
Dim Sheet As Object
Dim Cell As Object
Dim NumberFormats As Object
Dim NumberFormatString As String
Dim NumberFormatId As Long
Dim LocalSettings As New com.sun.star.lang.Locale
```

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

Cell.Value = 23400.3523565

LocalSettings.Language = "en"
LocalSettings.Country = "us"

NumberFormats = Doc.NumberFormats
NumberFormatString = "#,##0.000"

NumberFormatId = NumberFormats.queryKey(NumberFormatString, LocalSettings, True)
If NumberFormatId = -1 Then
   NumberFormatId = NumberFormats.addNew(NumberFormatString, LocalSettings)
End If

MsgBox NumberFormatId
Cell.NumberFormat = NumberFormatId
```

La boîte de dialogue **Formatage des cellules** de StarOffice présente un aperçu des différentes options de formatage des cellules.

Propriétés de page

Les propriétés de page sont des options de formatage permettant de positionner le contenu d'un document sur une page, ainsi que des éléments visuels qui se répètent d'une page à l'autre. Il s'agit notamment des éléments suivants :

- formats de papier ;
- marges ;
- en-têtes et pieds de page.

La procédure de définition des formats de page diffère des autres formes de formatage. En effet, alors que les formats de cellules, de paragraphes et de caractères peuvent être appliqués directement, les formats de page peuvent également être définis et appliqués indirectement à l'aide des styles de page. Par exemple, les en-têtes ou les pieds de page sont ajoutés au style de page.

Les sections qui suivent décrivent les principales options de formatage des pages de feuilles de calcul. La plupart des styles décrits peuvent également être appliqués aux documents texte. Les propriétés de page valides pour les deux types de documents sont définies dans le service `com.sun.star.style.PageProperties`. Les propriétés de page qui ne s'appliquent qu'aux classeurs sont définies dans le service `com.sun.star.sheet.TablePageStyle`.

Remarque – Les propriétés de page (marges, bordures, etc.) d'un document Microsoft Office sont définies par un objet `PageSetup` au niveau de l'objet `Worksheet` (Excel) ou `Document` (Word). Dans StarOffice, ces propriétés sont définies à l'aide d'un style de page lié au document associé.

Arrière-plan de page

Le service `com.sun.star.style.PageProperties` définit les propriétés d'arrière-plan de page suivantes :

- **BackColor (long)** : couleur d'arrière-plan.
- **BackGraphicURL (String)** : URL de l'image d'arrière-plan à utiliser.
- **BackGraphicFilter (String)** : nom du filtre d'interprétation de l'image d'arrière-plan.
- **BackGraphicLocation (Enum)** : position de l'image d'arrière-plan (valeur correspondant à l'énumération)
- **BackTransparent (Boolean)** : rend l'arrière-plan transparent.

Format de page

Le format de page est défini à l'aide des propriétés suivantes du service `com.sun.star.style.PageProperties` :

- **IsLandscape (Boolean)** : format paysage.
- **Width (long)** : largeur de page en centièmes de millimètre.
- **Height (long)** : hauteur de page en centièmes de millimètre.
- **PrinterPaperTray (String)** : nom du bac d'alimentation d'imprimante à utiliser.

L'exemple suivant règle la taille de page du style de page par défaut (Default) sur le format paysage DIN A5 (hauteur 14,8 cm, largeur 21 cm) :

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.IsLandscape = True
DefPage.Width = 21000
```

```
DefPage.Height = 14800
```

Marge, bordure et ombre

Le service `com.sun.star.style.PageProperties` fournit les propriétés suivantes pour l'ajustement des marges, des bordures et des ombres :

- **LeftMargin (long)** : largeur de la marge gauche en centièmes de millimètre.
- **RightMargin (long)** : largeur de la marge droite en centièmes de millimètre
- **TopMargin (long)** : largeur de la marge supérieure en centièmes de millimètre.
- **BottomMargin (long)** : largeur de la marge inférieure en centièmes de millimètre.
- **LeftBorder (struct)** : spécifications de la bordure de page gauche (structure `com.sun.star.table.BorderLine`).
- **RightBorder (struct)** : spécifications de la bordure de page droite (structure `com.sun.star.table.BorderLine`).
- **TopBorder (struct)** : spécifications de la bordure de page supérieure (structure `com.sun.star.table.BorderLine`).
- **BottomBorder (struct)** : spécifications de la bordure de page inférieure (structure `com.sun.star.table.BorderLine`).
- **LeftBorderDistance (long)** : distance entre la bordure gauche de la page et le contenu de la page, exprimée en centièmes de millimètre.
- **RightBorderDistance (long)** : distance entre la bordure droite de la page et le contenu de la page, exprimée en centièmes de millimètre.
- **TopBorderDistance (long)** : distance entre la bordure supérieure de la page et le contenu de la page, exprimée en centièmes de millimètre.
- **BottomBorderDistance (long)** : distance entre la bordure inférieure de la page et le contenu de la page, exprimée en centièmes de millimètre.
- **ShadowFormat (struct)** : spécifications de l'ombre de la zone de contenu de la page (structure `com.sun.star.table.ShadowFormat`).

L'exemple suivant définit les bordures gauche et droite du style de page par défaut (Default) à 1 centimètre.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.LeftMargin = 1000
```

```
DefPage.RightMargin = 1000
```

En-têtes et pieds de page

Les en-têtes et pieds de page d'un document font partie des propriétés de la page et sont définis à l'aide du service `com.sun.star.style.PageProperties`. Les propriétés de formatage des en-têtes sont les suivantes :

- **HeaderIsOn (Boolean)** : l'en-tête est activé.
- **HeaderLeftMargin (long)** : distance entre l'en-tête et la marge gauche en centièmes de millimètre.
- **HeaderRightMargin (long)** : distance entre l'en-tête et la marge droite en centièmes de millimètre.
- **HeaderBodyDistance (long)** : distance entre l'en-tête et le corps du document en centièmes de millimètre.
- **HeaderHeight (long)** : hauteur de l'en-tête en centièmes de millimètre.
- **HeaderIsDynamicHeight (Boolean)** : la hauteur de l'en-tête s'adapte automatiquement au contenu.
- **HeaderLeftBorder (struct)** : détails de la bordure gauche du cadre entourant l'en-tête (structure `com.sun.star.table.BorderLine`).
- **HeaderRightBorder (struct)** : détails de la bordure droite du cadre entourant l'en-tête (structure `com.sun.star.table.BorderLine`).
- **HeaderTopBorder (struct)** : détails de la bordure supérieure entourant l'en-tête (structure `com.sun.star.table.BorderLine`).
- **HeaderBottomBorder (struct)** : détails de la bordure inférieure entourant l'en-tête (structure `com.sun.star.table.BorderLine`).
- **HeaderLeftBorderDistance (long)** : distance entre la bordure gauche et le contenu de l'en-tête en centièmes de millimètre.
- **HeaderRightBorderDistance (long)** : distance entre la bordure droite et le contenu de l'en-tête en centièmes de millimètre.
- **HeaderTopBorderDistance (long)** : distance entre la bordure supérieure et le contenu de l'en-tête en centièmes de millimètre.
- **HeaderBottomBorderDistance (long)** : distance entre la bordure inférieure et le contenu de l'en-tête en centièmes de millimètre.
- **HeaderIsShared (Boolean)** : le contenu des en-têtes de pages paires et impaires est identique (voir `HeaderText`, `HeaderTextLeft` et `HeaderTextRight`).
- **HeaderBackColor (long)** : couleur d'arrière-plan de l'en-tête.
- **HeaderBackGraphicURL (String)** : URL de l'image d'arrière-plan à utiliser.
- **HeaderBackGraphicFilter (String)** : nom du filtre d'interprétation de l'image d'arrière-plan pour l'en-tête.

- **HeaderBackGraphicLocation (Enum)** : position de l'image d'arrière-plan pour l'en-tête (valeur correspondant à l'énumération `com.sun.star.style.GraphicLocation`).
- **HeaderBackTransparent (Boolean)** : l'arrière-plan de l'en-tête est transparent.
- **HeaderShadowFormat (struct)** : détails de l'ombre de l'en-tête (structure `com.sun.star.table.ShadowFormat`).

Les propriétés de formatage des pieds de page sont les suivantes :

- **FooterIsOn (Boolean)** : indique que le pied de page est activé.
- **FooterLeftMargin (long)** : distance entre le pied de page et la marge gauche en centièmes de millimètre.
- **FooterRightMargin (long)** : distance entre le pied de page et la marge droite en centièmes de millimètre.
- **FooterBodyDistance (long)** : distance entre le pied de page et le corps du document en centièmes de millimètre.
- **FooterHeight (long)** : hauteur du pied de page en centièmes de millimètre.
- **FooterIsDynamicHeight (Boolean)** : la hauteur du pied de page s'adapte automatiquement au contenu.
- **FooterLeftBorder (struct)** : détails de la bordure gauche entourant le pied de page (structure `com.sun.star.table.BorderLine`).
- **FooterRightBorder (struct)** : détails de la bordure droite entourant le pied de page (structure `com.sun.star.table.BorderLine`).
- **FooterTopBorder (struct)** : détails de la bordure supérieure entourant le pied de page (structure `com.sun.star.table.BorderLine`).
- **FooterBottomBorder (struct)** : détails de la bordure inférieure entourant le pied de page (structure `com.sun.star.table.BorderLine`).
- **FooterLeftBorderDistance (long)** : distance entre la bordure gauche et le contenu du pied de page en centièmes de millimètre.
- **FooterRightBorderDistance (long)** : distance entre la bordure droite et le contenu du pied de page, exprimée en centièmes de millimètre.
- **FooterTopBorderDistance (long)** : distance entre la bordure supérieure et le contenu du pied de page, exprimée en centièmes de millimètre.
- **FooterBottomBorderDistance (long)** : distance entre la bordure inférieure et le contenu du pied de page, exprimée en centièmes de millimètre.
- **FooterIsShared (Boolean)** : le contenu des pieds de page des pages paires et impaires est identique (voir `FooterText`, `FooterTextLeft` et `FooterTextRight`).
- **FooterBackColor (long)** : couleur d'arrière-plan du pied de page.
- **FooterBackGraphicURL (String)** : URL de l'image d'arrière-plan à utiliser.
- **FooterBackGraphicFilter (String)** : nom du filtre d'interprétation de l'image d'arrière-plan pour le pied de page.

- **FooterBackGraphicLocation (Enum)** : position de l'image d'arrière-plan pour le pied de page (valeur correspondant à l'énumération `com.sun.star.style.GraphicLocation`).
- **FooterBackTransparent (Boolean)** : l'arrière-plan du pied de page est transparent.
- **FooterShadowFormat (struct)** : détails de l'ombre du pied de page (structure `com.sun.star.table.ShadowFormat`).

Modification du texte des en-têtes et pieds de pages

Le contenu des en-têtes et pieds de pages d'un classeur est accessible via les propriétés suivantes :

- **LeftPageHeaderContent (Object)** : contenu des en-têtes des pages paires (service `com.sun.star.sheet.HeaderFooterContent`).
- **RightPageHeaderContent (Object)** : contenu des en-têtes des pages impaires (service `com.sun.star.sheet.HeaderFooterContent`).
- **LeftPageFooterContent (Object)** : contenu des pieds des pages paires (service `com.sun.star.sheet.HeaderFooterContent`).
- **RightPageFooterContent (Object)** : contenu des pieds des pages impaires (service `com.sun.star.sheet.HeaderFooterContent`).

Si vous n'avez pas besoin de faire la distinction entre pages paires et impaires dans les en-têtes ou les pieds de page (la valeur de la propriété `FooterIsShared` est `False`), définissez les propriétés des en-têtes et des pieds de page sur les pages impaires.

Tous les objets nommés renvoient un objet prenant en charge le service `com.sun.star.sheet.HeaderFooterContent`. À l'aide des propriétés non authentiques `LeftText`, `CenterText` et `RightText`, ce service fournit trois éléments de texte aux en-têtes et pieds de page de StarOffice Calc.

L'exemple ci-dessous écrit la valeur 'Just a Test' (ceci est un test) dans le champ texte situé à gauche de l'en-tête à partir du modèle Default.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object
Dim HContent As Object
Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
```

```
HText.String = "Just a Test."
DefPage.RightPageHeaderContent = HContent
```

Notez bien la dernière ligne de l'exemple : une fois le texte modifié, l'objet `TextContent` doit être de nouveau assigné à l'en-tête pour que la modification prenne effet.

Un autre mécanisme de modification du texte des en-têtes et pieds de page existe pour les documents texte (StarOffice Writer), car ceux-ci sont constitués d'un bloc de texte unique. Les propriétés suivantes sont définies dans le service `com.sun.star.style.PageProperties` :

- **HeaderText (Object)** : objet Text comprenant le contenu de l'en-tête (service `com.sun.star.text.XText`).
- **HeaderTextLeft (Object)** : objet Text comprenant le contenu des en-têtes des pages de gauche (service `com.sun.star.text.XText`).
- **HeaderTextRight (Object)** : objet Text comprenant le contenu des en-têtes des pages de droite (service `com.sun.star.text.XText`).
- **FooterText (Object)** : objet Text comprenant le contenu du pied de page (service `com.sun.star.text.XText`).
- **FooterTextLeft (Object)** : objet Text comprenant le contenu des pieds des pages de gauche (service `com.sun.star.text.XText`).
- **FooterTextRight (Object)** : objet Text comprenant le contenu des pieds des pages de droite (service`com.sun.star.text.XText`).

L'exemple suivant crée un en-tête dans le style de page par défaut (Default) des documents texte et ajoute le texte 'Just a Test' (ceci est un test) à cet en-tête.

```
Dim Doc As Object
Dim Sheet As Object
Dim StyleFamilies As Object
Dim PageStyles As Object
Dim DefPage As Object
Dim HText As Object

Doc = StarDesktop.CurrentComponent
StyleFamilies = Doc.StyleFamilies
PageStyles = StyleFamilies.getByName("PageStyles")
DefPage = PageStyles.getByName("Default")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Just a Test."
```

Dans cet exemple, l'accès est fourni directement par la propriété `HeaderText` du style de page, plutôt que par l'objet `HeaderFooterContent`.

Centrage (feuilles de calcul uniquement)

Le service `com.sun.star.sheet.TablePageStyle`, utilisé uniquement dans les styles de page de StarOffice Calc, permet de centrer sur la page les plages de cellules que vous souhaitez imprimer. Ce service fournit les propriétés suivantes :

- **CenterHorizontally (Boolean)** : le contenu du tableau est centré horizontalement.
- **CenterVertically (Boolean)** : le contenu du tableau est centré verticalement.

Définition des éléments à imprimer (feuilles de calcul uniquement)

Lorsque vous formatez des feuilles, vous pouvez définir si les éléments de la page doivent être visibles ou non. Pour cela, le service `com.sun.star.sheet.TablePageStyle` propose les propriétés suivantes :

- **PrintAnnotations (Boolean)** : imprime les commentaires des cellules.
- **PrintGrid (Boolean)** : imprime les lignes de la grille.
- **PrintHeaders (Boolean)** : imprime les en-têtes de ligne et de colonne.
- **PrintCharts (Boolean)** : imprime les diagrammes contenus dans une feuille.
- **PrintObjects (Boolean)** : imprime les objets incorporés.
- **PrintDrawing (Boolean)** : imprime les objets de dessin.
- **PrintDownFirst (Boolean)** : si le contenu d'une feuille s'étend sur plusieurs pages, les cellules sont d'abord imprimées verticalement du haut vers le bas, puis de gauche à droite.
- **PrintFormulas (Boolean)** : imprime les formules à la place des valeurs calculées.
- **PrintZeroValues (Boolean)** : imprime les valeurs zéro.

Édition efficace des classeurs

Alors que la section précédente décrivait la structure principale des classeurs, la présente section est consacrée aux services permettant d'accéder facilement à des cellules individuelles ou à des plages de cellules.

Plages de cellules

En plus de l'objet destiné aux cellules individuelles (voir le service `com.sun.star.table.Cell`), StarOffice propose également des objets représentant des plages de cellules. Les objets tels que `CellRange` sont créés avec l'appel `getCellRangeByName` de l'objet Spreadsheet :

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C15")
```

Le signe deux-points (:) sert à spécifier une plage de cellules dans une feuille de calcul. Par exemple, A1:C15 représente toutes les cellules des lignes 1 à 15 et des colonnes A, B et C.

L'emplacement des cellules individuelles dans une plage de cellules peut être déterminé avec la méthode `getCellByPosition`, où les coordonnées de la cellule supérieure gauche de la plage sont (0, 0). L'exemple suivant utilise cette méthode pour créer un objet à partir de la cellule C3.

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

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("B2:D4")
Cell = CellRange.GetCellByPosition(1, 1)
```

Formatage de plages de cellules

Tout comme pour les cellules individuelles, vous pouvez appliquer un formatage à des plages de cellules en utilisant le service `com.sun.star.table.CellProperties`. Pour plus d'informations et d'autres exemples de ce service, reportez-vous à la section *Formatage*.

Calculs avec des plages de cellules

Vous pouvez utiliser la méthode `computeFunction` pour effectuer des opérations mathématiques sur des plages de cellules. La méthode `computeFunction` attend une constante comme paramètre de description de la fonction mathématique à utiliser. Les constantes associées sont définies dans l'énumération `com.sun.star.sheet.GeneralFunction`. Les valeurs disponibles sont les suivantes :

- **SUM** : somme de toutes les valeurs numériques.
- **COUNT** : nombre total de valeurs (y compris les valeurs non numériques).
- **COUNTNUMS** : nombre total de valeurs numériques.
- **AVERAGE** : moyenne de toutes les valeurs numériques.
- **MAX** : valeur numérique la plus élevée.

- **MIN** : valeur numérique la plus basse.
- **PRODUCT** : produit de toutes les valeurs numériques.
- **STDEV** : écart type.
- **VAR** : variance.
- **STDEVP** : écart type calculé sur la base de la population totale.
- **VARP** : variance calculée sur la base de la population totale.

L'exemple suivant calcule la valeur moyenne de la plage `A1:C3` et affiche le résultat dans une boîte de message :

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.getByName("Sheet 1")
CellRange = Sheet.getCellRangeByName("A1:C3")

MsgBox CellRange.computeFunction(com.sun.star.sheet.GeneralFunction.AVERAGE)
```

Suppression du contenu des cellules

La méthode `clearContents` simplifie le processus de suppression du contenu des cellules ou des plages de cellules, car elle permet de supprimer un type de contenu particulier d'une plage de cellules.

L'exemple suivant supprime toutes les chaînes et les informations de formatage direct de la plage `B2:C3`.

```
Dim Doc As Object
Dim Sheet As Object
Dim CellRange As Object
Dim Flags As Long

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets(0)
CellRange = Sheet.getCellRangeByName("B2:C3")

Flags = com.sun.star.sheet.CellFlags.STRING + _
      com.sun.star.sheet.CellFlags.HARDATTR

CellRange.clearContents(Flags)
```

Les drapeaux spécifiés dans `clearContents` proviennent de la liste de constantes `com.sun.star.sheet.CellFlags`. Cette liste fournit les éléments suivants :

- **VALUE** : valeurs numériques qui ne sont pas formatées sous forme de date ou d'heure.
- **DATETIME** : valeurs numériques formatées sous forme de date ou d'heure.

- **STRING** : chaînes.
- **ANNOTATION** : commentaires liés aux cellules.
- **FORMULA** : formules.
- **HARDATTR** : formatage direct des cellules.
- **STYLES** : formatage indirect.
- **OBJECTS** : objets de dessin liés aux cellules.
- **EDITATTR** : formatage des caractères ne s'appliquant qu'à certaines parties des cellules.

Vous pouvez également ajouter un ensemble de constantes pour supprimer différentes informations avec un appel de `clearContents`.

Recherche et remplacement du contenu des cellules

Les classeurs, comme les documents texte, proposent une fonction de recherche et remplacement.

Les objets Descriptor pour la recherche et le remplacement dans les classeurs ne sont pas créés directement au moyen de l'objet Document, mais plutôt avec la liste `Sheets`. L'exemple suivant illustre un processus de recherche et de remplacement :

```
Dim Doc As Object
Dim Sheet As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

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

ReplaceDescriptor = Sheet.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"
For I = 0 to Doc.Sheets.Count - 1
   Sheet = Doc.Sheets(I)
   Sheet.ReplaceAll(ReplaceDescriptor)
Next I
```

Cet exemple utilise la première page du document pour créer un objet `ReplaceDescriptor`, puis l'applique à toutes les pages dans une boucle.

CHAPITRE 8

Dessins et présentations

Ce chapitre propose une introduction à la création et à l'édition de dessins à partir de macros. La première section décrit la structure des dessins, et notamment les éléments de base contenus dans les dessins. La seconde section concerne les fonctions d'édition plus complexes, comme le groupement, la rotation et la mise à l'échelle d'objets.

Vous trouverez des informations sur la création, l'ouverture et l'enregistrement de dessins dans le chapitre 5, *Utilisation de documents StarOffice*.

Structure des dessins

StarOffice ne limite pas le nombre de pages que peut contenir un dessin. Vous pouvez concevoir chaque page séparément. Il n'y a pas de limite non plus sur le nombre d'éléments de dessin que vous pouvez ajouter à une page.

La présence de *couches* rend la situation légèrement plus complexe. Par défaut, chaque document contient les couches *Mise en page*, *Contrôles* et *Lignes de cote*. Tous les éléments de dessin sont ajoutés à la couche *Mise en page*. Vous avez également la possibilité d'ajouter de nouvelles couches. Pour plus d'informations sur les couches de dessin, reportez-vous au StarOffice Developer's Guide (Guide du développeur).

Pages

Les pages d'un dessin sont disponibles via la liste `DrawPages`. Vous pouvez accéder aux différentes pages soit par leur numéro, soit par leur nom. Si un document contient une seule page, nommée *Slide 1*, les exemples suivants seront équivalents.

Exemple 1 :

```
Dim Doc As Object
Dim Page As Object
```

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

Exemple 2 :

```
Dim Doc As Object
Dim Page As Object

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages.getByName("Slide 1")
```

Dans le premier exemple, on accède à la page par son numéro (la numérotation commençant à 0). Dans le second, on y accède par son nom et par la méthode `getByName`.

```
Dim sUrl As String, sFilter As String
Dim sOptions As String
Dim oSheets As Object, oSheet As Object

   oSheets = oDocument.Sheets

   If oSheets.hasByName("Link") Then
      oSheet = oSheets.getByName("Link")
   Else
      oSheet = oDocument.createInstance("com.sun.star.sheet.Spreadsheet")
      oSheets.insertByName("Link", oSheet)
      oSheet.IsVisible = False
   End If
```

L'appel ci-dessus renvoie un objet Page prenant en charge le service `com.sun.star.drawing.DrawPage`. Ce service reconnaît les propriétés suivantes :

- **BorderLeft (Long)** : bordure gauche en centièmes de millimètre.
- **BorderRight (Long)** : bordure droite en centièmes de millimètre.
- **BorderTop (Long)** : bordure supérieure en centièmes de millimètre.
- **BorderBottom (Long)** : bordure inférieure en centièmes de millimètre.
- **Width (Long)** : largeur de page en centièmes de millimètre.
- **Height (Long)** : hauteur de page en centièmes de millimètre.
- **Number (Short)** : nombre de pages (la numérotation commençant à 1), en lecture seule.
- **Orientation (Enum)** : orientation de la page correspondant à l'énumération `com.sun.star.view.PaperOrientation`.

En cas de modification de ces paramètres, *toutes* les pages du document seront affectées.

Dans l'exemple suivant, la taille de page du dessin qui vient d'être ouvert est définie sur 20 × 20 centimètres, avec une marge de 0,5 centimètre :

```
Dim Doc As Object
Dim Page As Object
```

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

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

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

Propriétés élémentaires des objets de dessin

Les objets de dessin comprennent des formes (rectangles, cercles, etc.), des lignes et des objets texte. Ils partagent tous un certain nombre de caractéristiques communes et prennent en charge le service `com.sun.star.drawing.Shape`. Ce service définit les propriétés `Size` et `Position` d'un objet de dessin.

StarOffice Basic propose également plusieurs autres services permettant de modifier ces propriétés, en appliquant du formatage ou des remplissages par exemple. Les options de formatage disponibles dépendent du type d'objet de dessin.

L'exemple suivant crée et insère un rectangle dans un dessin :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

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

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)
```

Cet exemple utilise l'appel `StarDesktop.CurrentComponent` pour déterminer quel document est ouvert. L'objet Document déterminé de cette manière renvoie la première page du dessin via l'appel `drawPages(0)`.

Les structures `Point` et `Size` contenant le point d'origine (angle gauche) et la taille de l'objet de dessin sont ensuite initialisées. Les longueurs sont spécifiées en centièmes de millimètre.

Le code du programme utilise ensuite l'appel `Doc.createInstance` pour créer l'objet de dessin rectangulaire tel qu'il a été spécifié par le service `com.sun.star.drawing.RectangleShape`. L'objet de dessin est ensuite assigné à une page avec un appel `Page.add`.

Propriétés de remplissage

Cette section décrit quatre services et, pour chacun, l'exemple de code utilise un élément de forme rectangulaire associé à différents types de formatage. Les propriétés de remplissage sont associées dans le service `com.sun.star.drawing.FillProperties`.

Au moment de remplir une zone, StarOffice reconnaît quatre types de formatage principaux. Le cas le plus simple est celui d'un remplissage uni. Les options définissant des dégradés de couleurs et des hachures permettent de faire appel à d'autres couleurs. Le quatrième cas de figure consiste à projeter des images existantes dans la zone à remplir.

Le mode Remplissage d'un objet de dessin se définit avec la propriété `FillStyle`. Les valeurs autorisées sont définies dans `com.sun.star.drawing.FillStyle`.

Remplissages unis

La principale propriété des remplissages unis est :

- **FillColor (Long)** : couleur de remplissage de la zone.

Pour utiliser le mode Remplissage, vous devez définir la propriété `FillStyle` sur le mode de Remplissage `SOLID`.

L'exemple suivant crée une forme rectangulaire et la remplit de rouge (valeur RVB 255, 0, 0) :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
```

```
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillColor = RGB(255,0,0)

Page.add(RectangleShape)
```

Dégradé de couleurs

Si vous définissez la propriété `FillStyle` sur `GRADIENT`, vous pourrez appliquer un dégradé de couleurs à toutes les zones à remplir dans un document StarOffice.

Pour appliquer un dégradé de couleurs prédéfini, il suffit d'assigner le nom associé de la propriété `FillTransparenceGradientName`. Pour définir votre propre dégradé de couleurs, vous devez constituer une structure `com.sun.star.awt.Gradient` afin d'assigner la propriété `FillGradient`. Cette propriété propose les options suivantes :

- **Style (Enum)** : type de dégradé (linéaire ou radial, par exemple), avec des valeurs par défaut correspondant à `com.sun.star.awt.GradientStyle`.
- **StartColor (Long)** : couleur de début du dégradé de couleurs.
- **EndColor (Long)** : couleur de fin du dégradé de couleurs.
- **Angle (Short)** : angle du dégradé de couleurs en dixièmes de degré.
- **XOffset (Short)** : coordonnée X correspondant au point de départ du dégradé de couleurs, exprimé en centièmes de millimètre.
- **YOffset (Short)** : coordonnée Y correspondant au point de départ du dégradé de couleurs, exprimé en centièmes de millimètre.
- **StartIntensity (Short)** : intensité de `StartColor` exprimée en pourcentage (dans StarOffice Basic, vous pouvez également spécifier des valeurs supérieures à 100 %).
- **EndIntensity (Short)** : intensité de `EndColor` exprimée en pourcentage (dans StarOffice Basic, vous pouvez également spécifier des valeurs supérieures à 100 %).
- **StepCount (Short)** : nombre de couleurs intermédiaires utilisées par StarOffice pour calculer les dégradés.

L'exemple suivant illustre l'utilisation de dégradés de couleurs avec la structure `com.sun.star.awt.Gradient` :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim Gradient As New com.sun.star.awt.Gradient

Point.x = 1000
Point.y = 1000
Size.Width = 10000
```

```
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point
Gradient.Style = com.sun.star.awt.GradientStyle.LINEAR
Gradient.StartColor = RGB(255,0,0)
Gradient.EndColor = RGB(0,255,0)
Gradient.StartIntensity = 150
Gradient.EndIntensity = 150
Gradient.Angle = 450
Gradient.StepCount = 100

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.GRADIENT
RectangleShape.FillGradient = Gradient

Page.add(RectangleShape)
```

Cet exemple crée un dégradé de couleurs linéaire (`Style = LINEAR`). Le dégradé commence en rouge (`StartColor`) dans l'angle supérieur gauche et finit, en suivant un angle de 45 degrés (`Angle`), en vert (`EndColor`) dans l'angle inférieur droit. L'intensité des couleurs de début et de fin est de 150 % (`StartIntensity` et `EndIntensity`), d'où un effet de couleurs apparemment plus claires que les valeurs spécifiées dans les propriétés `StartColor` et `EndColor`. Le dégradé de couleurs est constitué de cent couleurs intermédiaires (`StepCount`).

Hachures

Pour créer un remplissage hachuré, vous devez définir la propriété `FillStyle` sur `HATCH`. Le code de programme de définition des hachures ressemble beaucoup à celui des dégradés de couleurs. La structure auxiliaire, `com.sun.star.drawing.Hatch` en l'occurrence, sert ici à définir l'apparence des hachures. La structure de hachures possède les propriétés suivantes :

- **Style (Enum)** : type de hachure (simple, carrée ou carrée avec diagonales), avec des valeurs par défaut correspondant à `com.sun.star.awt.HatchStyle`.
- **Color (Long)** : couleur des lignes.
- **Distance (Long)** : distance entre les lignes en centièmes de millimètre.
- **Angle (Short)** : angle de la hachure en dixièmes de degré.

L'exemple suivant illustre l'utilisation d'une structure de hachures :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
```

```
Dim Hatch As New com.sun.star.drawing.Hatch

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.HATCH

Hatch.Style = com.sun.star.drawing.HatchStyle.SINGLE
Hatch.Color = RGB(64,64,64)
Hatch.Distance = 20
Hatch.Angle = 450

RectangleShape.FillHatch = Hatch

Page.add(RectangleShape)
```

Ce code crée une structure de hachures simple (`HatchStyle = SINGLE`) et fait pivoter les lignes de cette structure sur 45 degrés (`Angle`). Les lignes sont gris foncé (`Color`) et espacées de 0,2 millimètres (`Distance`).

Bitmaps

Pour utiliser une projection bitmap comme remplissage, vous devez définir la propriété `FillStyle` sur `BITMAP`. Si le bitmap est déjà disponible dans StarOffice, il vous suffit de spécifier son nom dans la propriété `FillBitMapName` et son style d'affichage (simple, carrelage ou étiré) dans la propriété `FillBitmapMode`. Les valeurs par défaut correspondent à `com.sun.star.drawing.BitmapMode`.

Pour utiliser un fichier bitmap externe, vous pouvez spécifier son URL dans la propriété `FillBitmapURL`.

L'exemple suivant crée un rectangle et dispose le bitmap Sky disponible dans StarOffice en carrelage, afin de remplir le rectangle :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
```

```
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
RectangleShape.FillBitmapName = "Sky"
RectangleShape.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT

Page.add(RectangleShape)
```

Transparence

Vous avez la possibilité d'adapter la transparence des différents remplissages que vous appliquez. La manière la plus simple de modifier la transparence d'un élément de dessin consiste à utiliser la propriété `FillTransparence`.

L'exemple suivant crée un rectangle rouge avec une transparence de 50 %.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.FillStyle = com.sun.star.drawing.FillStyle.SOLID
RectangleShape.FillTransparence = 50
RectangleShape.FillColor = RGB(255,0,0)

Page.add(RectangleShape)
```

Pour rendre le remplissage transparent, définissez la propriété `FillTransparence` sur 100.

En plus de la propriété `FillTransparence`, le service `com.sun.star.drawing.FillProperties` propose également la propriété `FillTransparenceGradient`. Elle sert à définir un dégradé pour la transparence d'une zone à remplir.

Propriétés de ligne

Tous les objets de dessin susceptibles d'avoir une ligne de bordure prennent en charge le service `com.sun.star.drawing.LineStyle`. Ce service propose notamment les propriétés suivantes :

- **LineStyle (Enum)** : type de ligne, avec des valeurs par défaut correspondant à `com.sun.star.drawing.LineStyle`.
- **LineColor (Long)** : couleur de ligne.
- **LineTransparence (Short)** : transparence de ligne.
- **LineWidth (Long)** : épaisseur de ligne en centièmes de millimètre.
- **LineJoint (Enum)** : transitions vers les points de connexion, avec des valeurs par défaut correspondant à `com.sun.star.drawing.LineJoint`.

L'exemple suivant crée un rectangle avec une bordure pleine (`LineStyle = SOLID`) de 5 millimètres d'épaisseur (`LineWidth`) et une transparence de 50 %. Les arêtes droite et gauche de la ligne se prolongent jusqu'à ce qu'elles se rejoignent (`LineJoint = MITER`) pour former un angle droit.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.LineColor = RGB(128,128,128)
RectangleShape.LineTransparence = 50
RectangleShape.LineWidth = 500
RectangleShape.LineJoint = com.sun.star.drawing.LineJoint.MITER

RectangleShape.LineStyle = com.sun.star.drawing.LineStyle.SOLID

Page.add(RectangleShape)
```

En plus des propriétés indiquées précédemment, le service `com.sun.star.drawing.LineStyle` propose des options permettant de dessiner des lignes pointillées. Pour plus d'informations, consultez la référence de l'API StarOffice.

Propriétés de texte (objets de dessin)

Les services `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties` permettent de formater du texte dans des objets de dessin. Ces services portent sur les caractères individuels et sur les paragraphes, et sont traités en détail dans le chapitre 6, *Documents texte*.

L'exemple suivant permet d'insérer du texte dans un rectangle et d'en formater la police grâce au service `com.sun.star.style.CharacterProperties`.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000
Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "This is a test"
RectangleShape.CharWeight = com.sun.star.awt.FontWeight.BOLD
RectangleShape.CharFontName = "Arial"
```

Ce code fait appel à la propriété `String` du rectangle pour insérer le texte, et aux propriétés `CharWeight` et `CharFontName` du service `com.sun.star.style.CharacterProperties` pour formater la police du texte.

Le texte ne peut être inséré qu'après que l'objet de dessin a été ajouté à la page de dessin. Vous pouvez également utiliser le service `com.sun.star.drawing.Text` pour positionner et formater du texte dans un objet de dessin. Voici quelques propriétés importantes de ce service :

- **TextAutoGrowHeight (Boolean)** : adapte la hauteur de l'élément de dessin au texte qu'il contient.
- **TextAutoGrowWidth (Boolean)** : adapte la largeur de l'élément de dessin au texte qu'il contient.
- **TextHorizontalAdjust (Enum)** : position horizontale du texte à l'intérieur de l'élément de dessin, avec des valeurs par défaut correspondant à `com.sun.star.drawing.TextHorizontalAdjust`.
- **TextVerticalAdjust (Enum)** : position verticale du texte à l'intérieur de l'élément de dessin, avec des valeurs par défaut correspondant à `com.sun.star.drawing.TextVerticalAdjust`.

- **TextLeftDistance (Long)** : écart sur la gauche séparant l'élément de dessin et le texte, exprimé en centièmes de millimètre.
- **TextRightDistance (Long)** : écart sur la droite séparant l'élément de dessin et le texte, exprimé en centièmes de millimètre.
- **TextUpperDistance (Long)** : écart séparant les bords supérieurs de l'élément de dessin et du texte, exprimé en centièmes de millimètre.
- **TextLowerDistance (Long)** : écart séparant les bords inférieurs de l'élément de dessin et du texte, exprimé en centièmes de millimètre.

L'exemple suivant illustre l'utilisation des propriétés nommées.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

Page.add(RectangleShape)

RectangleShape.String = "This is a test"   ' Ne peut se trouver
                                           ' qu'après Page.add!

RectangleShape.TextVerticalAdjust = com.sun.star.drawing.TextVerticalAdjust.TOP
RectangleShape.TextHorizontalAdjust = com.sun.star.drawing.TextHorizontalAdjust.LEFT

RectangleShape.TextLeftDistance = 300
RectangleShape.TextRightDistance = 300
RectangleShape.TextUpperDistance = 300
RectangleShape.TextLowerDistance = 300
```

Ce code insère un élément de dessin dans une page, puis ajoute du texte dans l'angle supérieur gauche de celui-ci grâce aux propriétés `TextVerticalAdjust` et `TextHorizontalAdjust`. L'écart minimum entre le texte et le bord de l'objet de dessin est de trois millimètres.

Propriétés d'ombre

Vous pouvez ajouter une ombre à la plupart des objets de dessin en utilisant le service `com.sun.star.drawing.ShadowProperties`. Ce service propose les propriétés suivantes :

- **Shadow (Boolean)** : active l'ombre.
- **ShadowColor (Long)** : couleur d'ombre.
- **ShadowTransparence (Short)** : transparence de l'ombre.
- **ShadowXDistance (Long)** : écart vertical séparant l'ombre et l'objet de dessin, exprimé en centièmes de millimètre.
- **ShadowYDistance (Long)** : écart horizontal séparant l'ombre et l'objet de dessin, exprimé en centièmes de millimètre.

L'exemple suivant crée un rectangle avec une ombre présentant un décalage vertical et un décalage horizontal de 2 millimètres. L'ombre prend une couleur gris foncé avec une transparence de 50 %.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.Shadow = True
RectangleShape.ShadowColor = RGB(192,192,192)
RectangleShape.ShadowTransparence = 50
RectangleShape.ShadowXDistance = 200
RectangleShape.ShadowYDistance = 200

Page.add(RectangleShape)
```

Présentation de différents objets de dessin

Formes rectangulaires

Les objets de forme rectangulaire (`com.sun.star.drawing.RectangleShape`) prennent en charge les services suivants pour leur formatage :

- **Propriétés de remplissage** : `com.sun.star.drawing.FillProperties`.
- **Propriétés de ligne** : `com.sun.star.drawing.LineProperties`.

- **Propriétés de texte** : `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`).
- **Propriétés d'ombre** : `com.sun.star.drawing.ShadowProperties`.
- **CornerRadius (Long)** : rayon d'arrondissement des angles en centièmes de millimètre.

Cercles et ellipses

Le service `com.sun.star.drawing.EllipseShape` gère les cercles et les ellipses, et prend en charge les services suivants :

- **Propriétés de remplissage** : `com.sun.star.drawing.FillProperties`.
- **Propriétés de ligne** : `com.sun.star.drawing.LineProperties`.
- **Propriétés de texte** : `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`).
- **Propriétés d'ombre** : `com.sun.star.drawing.ShadowProperties`.

Outre ces services, les cercles et les ellipses offrent les propriétés suivantes :

- **CircleKind (Enum)** : type de cercle ou d'ellipse, avec des valeurs par défaut correspondant à `com.sun.star.drawing.CircleKind`.
- **CircleStartAngle (Long)** : angle de départ exprimé en dixièmes de degré (pour les segments de cercle ou d'ellipse uniquement).
- **CircleEndAngle (Long)** : angle final exprimé en dixièmes de degré (pour les segments de cercle ou d'ellipse uniquement).

La propriété `CircleKind` détermine si un objet est un cercle complet, une tranche de cercle ou une section de cercle. Les valeurs disponibles sont les suivantes :

- **com.sun.star.drawing.CircleKind.FULL** : cercle entier ou ellipse entière.
- **com.sun.star.drawing.CircleKind.CUT** : section de cercle (cercle partiel dont les interfaces sont liées directement les unes aux autres).
- **com.sun.star.drawing.CircleKind.SECTION** : tranche de cercle.
- **com.sun.star.drawing.CircleKind.ARC** : angle (sans ligne de cercle).

L'exemple suivant crée une tranche de cercle avec un angle de 70 degrés, résultant de la différence entre l'angle de départ (20 degrés) et l'angle final (90 degrés).

```
Dim Doc As Object
Dim Page As Object
Dim EllipseShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
```

```
Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

EllipseShape = Doc.createInstance("com.sun.star.drawing.EllipseShape")
EllipseShape.Size = Size
EllipseShape.Position = Point

EllipseShape.CircleStartAngle = 2000
EllipseShape.CircleEndAngle = 9000
EllipseShape.CircleKind =  com.sun.star.drawing.CircleKind.SECTION

Page.add(EllipseShape)
```

Lignes

StarOffice propose le service `com.sun.star.drawing.LineShape` pour les objets Ligne. Les objets Ligne prennent en charge tous les services de formatage généraux, à l'exception des zones. Vous trouverez ci-après toutes les propriétés associées au service `LineShape` :

- **Propriétés de ligne** : `com.sun.star.drawing.LineProperties`.
- **Propriétés de texte** : `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`).
- **Propriétés d'ombre** : `com.sun.star.drawing.ShadowProperties`.

L'exemple suivant crée et formate une ligne à l'aide des propriétés nommées. L'origine de la ligne est spécifiée dans la propriété `Location`, tandis que les coordonnées répertoriées dans la propriété `Size` déterminent le point limite de la ligne.

```
Dim Doc As Object
Dim Page As Object
Dim LineShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

LineShape = Doc.createInstance("com.sun.star.drawing.LineShape")
```

```
LineShape.Size = Size
LineShape.Position = Point

Page.add(LineShape)
```

Formes polypolygonales

StarOffice prend également en charge les formes polygonales complexes via le service `com.sun.star.drawing.PolyPolygonShape`. Un *polypolygone* n'est pas à proprement parler un simple polygone : il s'agit d'un polygone multiple. Il est donc possible de spécifier plusieurs listes indépendantes contenant des points d'inflexion et de les combiner pour former un objet complet.

Comme pour les formes rectangulaires, toutes les propriétés de formatage des objets de dessin sont également disponibles pour les polypolygones :

- **Propriétés de remplissage** : `com.sun.star.drawing.FillProperties`.
- **Propriétés de ligne** : `com.sun.star.drawing.LineProperties`.
- **Propriétés de texte** : `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`).
- **Propriétés d'ombre** : `com.sun.star.drawing.ShadowProperties`.

Le service `PolyPolygonShape` propose également une propriété permettant de définir les coordonnées d'un polygone :

- `PolyPolygon (Array)` : champ contenant les coordonnées du polygone (matrice double possédant des points de type `com.sun.star.awt.Point`).

L'exemple suivant montre comment définir un triangle avec le service `PolyPolygonShape`.

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Coordinates(2) As New com.sun.star.awt.Point

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

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")
Page.add(PolyPolygonShape)   ' Page.add doit être spécifié avant les coordonnées

Coordinates(0).x = 1000
Coordinates(1).x = 7500
Coordinates(2).x = 10000
Coordinates(0).y = 1000
Coordinates(1).y = 7500
Coordinates(2).y = 5000
```

```
PolyPolygonShape.PolyPolygon = Array(Coordinates())
```

Les points d'un polygone étant définis comme des valeurs absolues, il n'est pas nécessaire d'en spécifier la taille ou la position de départ. Nous vous conseillons de créer une matrice de points, de l'intégrer à une seconde matrice (à l'aide de l'appel `Array(Coordinates()),()` puis d'assigner cette matrice au polygone. Pour que l'appel correspondant puisse s'effectuer, le polygone doit être inséré dans le document.

La matrice double de la définition permet de créer des formes complexes en fusionnant plusieurs polygones. Vous pouvez créer par exemple un rectangle et y insérer un autre rectangle afin de créer un trou dans le rectangle d'origine :

```
Dim Doc As Object
Dim Page As Object
Dim PolyPolygonShape As Object
Dim PolyPolygon As Variant
Dim Square1(3) As New com.sun.star.awt.Point
Dim Square2(3) As New com.sun.star.awt.Point
Dim Square3(3) As New com.sun.star.awt.Point

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

PolyPolygonShape = Doc.createInstance("com.sun.star.drawing.PolyPolygonShape")

Page.add(PolyPolygonShape)   ' Page.add doit être spécifié avant les coordonnées

Square1(0).x = 5000
Square1(1).x = 10000
Square1(2).x = 10000
Square1(3).x = 5000
Square1(0).y = 5000
Square1(1).y = 5000
Square1(2).y = 10000
Square1(3).y = 10000

Square2(0).x = 6500
Square2(1).x = 8500
Square2(2).x = 8500
Square2(3).x = 6500
Square2(0).y = 6500
Square2(1).y = 6500
Square2(2).y = 8500
Square2(3).y = 8500

Square3(0).x = 6500
Square3(1).x = 8500
Square3(2).x = 8500
Square3(3).x = 6500
Square3(0).y = 9000
Square3(1).y = 9000
Square3(2).y = 9500
Square3(3).y = 9500
```

```
PolyPolygonShape.PolyPolygon = Array(Square1(), Square2(), Square3())
```

Pour distinguer les zones remplies des zones à trous, StarOffice applique une règle simple : l'arête de la forme extérieure correspond toujours à la bordure extérieure du polypolygone. La ligne suivante vers l'intérieur correspond à la bordure intérieure de la forme et marque la transition vers le premier trou. S'il existe une autre ligne vers l'intérieur, elle marque la transition vers une zone remplie.

Images

Les derniers éléments de dessin présentés ici sont des objets graphiques basés sur le service `com.sun.star.drawing.GraphicObjectShape`. Ils peuvent être utilisés avec les images de StarOffice, dont l'apparence peut être adaptée grâce à une vaste gamme de propriétés.

Les objets graphiques prennent en charge deux des propriétés de formatage générales :

- **Propriétés de texte** : `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`).
- **Propriétés d'ombre** : `com.sun.star.drawing.ShadowProperties`.

Les autres propriétés prises en charge par les objets graphiques sont les suivantes :

- **GraphicURL (String)** : URL de l'image.
- **AdjustLuminance (Short)** : luminance des couleurs, exprimée en pourcentage (les valeurs négatives sont autorisées).
- **AdjustContrast (Short)** : contraste exprimé en pourcentage (les valeurs négatives sont autorisées).
- **AdjustRed (Short)** : valeur de la couleur rouge exprimée en pourcentage (les valeurs négatives sont autorisées).
- **AdjustGreen (Short)** : valeur de la couleur verte exprimée en pourcentage (les valeurs négatives sont autorisées).
- **AdjustBlue (Short)** : valeur de la couleur bleue exprimée en pourcentage (les valeurs négatives sont autorisées).
- **Gamma (Short)** : valeur gamma d'une image.
- **Transparency (Short)** : transparence d'une image exprimée en pourcentage.
- **GraphicColorMode (enum)** : mode de couleur (par exemple, standard, niveaux de gris, noir et blanc) avec des valeurs par défaut correspondant à `com.sun.star.drawing.ColorMode`.

L'exemple suivant montre comment insérer une page dans une image object.Dim Doc As Object :

```
Dim Page As Object
Dim GraphicObjectShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000         ' spécifications, non primordiales car les dernières
                       ' coordonnées

Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

GraphicObjectShape = Doc.createInstance("com.sun.star.drawing.GraphicObjectShape")

GraphicObjectShape.Size = Size
GraphicObjectShape.Position = Point

GraphicObjectShape.GraphicURL = "file:///c:/test.jpg"
GraphicObjectShape.AdjustBlue = -50
GraphicObjectShape.AdjustGreen = 5
GraphicObjectShape.AdjustBlue = 10
GraphicObjectShape.AdjustContrast = 20
GraphicObjectShape.AdjustLuminance = 50
GraphicObjectShape.Transparency = 40
GraphicObjectShape.GraphicColorMode = com.sun.star.drawing.ColorMode.STANDARD

Page.add(GraphicObjectShape)
```

Ce code permet d'insérer l'image `test.jpg` et d'adapter son apparence à l'aide des propriétés `Adjust`. Dans cet exemple, les images sont représentées avec 40 % de transparence et aucune autre conversion de couleur (`GraphicColorMode = STANDARD`).

Édition des objets de dessin

Groupement d'objets

Il est souvent utile de regrouper plusieurs objets de dessin individuels afin qu'ils se comportent comme un seul objet.

L'exemple suivant associe deux objets de dessin :

```
Dim Doc As Object
Dim Page As Object
Dim Square As Object
```

```
Dim Circle As Object
Dim Shapes As Object
Dim Group As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size
Dim NewPos As New com.sun.star.awt.Point
Dim Height As Long
Dim Width As Long

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
Point.x = 3000
Point.y = 3000
Size.Width = 3000
Size.Height = 3000
' create square drawing element
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' create circle drawing element
Circle = Doc.createInstance("com.sun.star.drawing.EllipseShape")
Circle.Size = Size
Circle.Position = Point
Circle.FillColor = RGB(255,128,128)
Circle.FillColor = RGB(0,255,0)
Page.add(Circle)
' combine square and circle drawing elements
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' centre combined drawing elements
Height = Page.Height
Width = Page.Width
NewPos.X = Width / 2
NewPos.Y = Height / 2
Height = Group.Size.Height
Width = Group.Size.Width
NewPos.X = NewPos.X - Width / 2
NewPos.Y = NewPos.Y - Height / 2
Group.Position = NewPos
```

Ce code crée un rectangle et un cercle, puis les insère dans une page. Un objet prenant en charge le service `com.sun.star.drawing.ShapeCollection` est ensuite créé et la méthode `Add` est utilisée pour ajouter le rectangle et le cercle à cet objet. Le service `ShapeCollection` est ajouté à la page à l'aide de la méthode `Group` et renvoie l'objet `Group` réel qui peut être édité individuellement comme une forme `Shape`.

Si vous souhaitez formater les différents objets d'un groupe, appliquez le formatage avant de les ajouter au groupe. Une fois qu'ils ont été insérés dans un groupe, les objets ne peuvent pas être modifiés.

Rotation et cisaillement des objets de dessin

Tous les objets de dessin décrits dans les sections précédentes peuvent également faire l'objet de rotations et de cisaillements à l'aide du service `com.sun.star.drawing.RotationDescriptor`.

Ce service offre les propriétés suivantes :

- **RotateAngle (Long)** : angle de rotation en centièmes de degré.
- **ShearAngle (Long)** : angle de cisaillement en centièmes de degré.

L'exemple suivant crée un rectangle et le fait pivoter de 30 degrés à l'aide de la propriété `RotateAngle` :

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

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

RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
RectangleShape.Position = Point

RectangleShape.RotateAngle = 3000

Page.add(RectangleShape)
```

L'exemple suivant crée le même rectangle que dans le précédent, mais en appliquant un cisaillement de 30 degrés à l'aide de la propriété `ShearAngle`.

```
Dim Doc As Object
Dim Page As Object
Dim RectangleShape As Object
Dim Point As New com.sun.star.awt.Point
Dim Size As New com.sun.star.awt.Size

Point.x = 1000
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Doc = StarDesktop.CurrentComponent
Page = Doc.drawPages(0)
RectangleShape = Doc.createInstance("com.sun.star.drawing.RectangleShape")
RectangleShape.Size = Size
```

```
RectangleShape.Position = Point

RectangleShape.ShearAngle = 3000

Page.add(RectangleShape)
```

Recherche et remplacement

Tout comme les documents texte, les documents de dessin comportent une fonction de recherche et de remplacement. Cette fonction est semblable à celle utilisée dans les documents texte et est décrite dans le chapitre 6, *Documents texte*. Dans les dessins, toutefois, les objets Descriptor pour la recherche et le remplacement ne sont pas créés directement au moyen de l'objet Document, mais par l'intermédiaire du niveau de caractère associé. L'exemple suivant illustre le processus de remplacement dans un dessin :

```
Dim Doc As Object
Dim Page As Object
Dim ReplaceDescriptor As Object
Dim I As Integer

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

ReplaceDescriptor = Page.createReplaceDescriptor()
ReplaceDescriptor.SearchString = "is"
ReplaceDescriptor.ReplaceString = "was"

For I = 0 to Doc.drawPages.Count - 1
   Page = Doc.drawPages(I)
   Page.ReplaceAll(ReplaceDescriptor)
Next I
```

Ce code utilise le premier `DrawPage` du document pour créer un `ReplaceDescriptor` avant d'appliquer ce descripteur dans une boucle à toutes les pages du dessin.

Présentations

Les présentations StarOffice sont basées sur des dessins. Chaque page de la présentation est une diapo. L'accès aux diapos s'effectue de la même manière que l'accès à un dessin standard, via la liste `DrawPages` de l'objet Document. Le service `com.sun.star.presentation.PresentationDocument` gère les présentations et fournit une version complète du service `com.sun.star.drawing.DrawingDocument`.

Utilisation des présentations

Outre les fonctions de dessin offertes par la propriété `Presentation`, les présentations contiennent un objet Presentation qui permet d'accéder aux propriétés principales et aux mécanismes de contrôle des présentations. Cet objet propose par exemple une méthode `start` permettant de lancer les présentations.

```
Dim Doc As Object
Dim Presentation As Object

Doc = StarDesktop.CurrentComponent
Presentation = Doc.Presentation
Presentation.start()
```

Le code utilisé dans cet exemple crée un objet `Doc` faisant référence à la présentation active et un objet Presentation correspondant. La méthode `start()` de l'objet permet de lancer l'exemple et d'exécuter la présentation à l'écran.

Les méthodes suivantes sont fournies comme objets Presentation :

- **start** : lance la présentation.
- **end** : met fin à la présentation.
- **rehearseTimings** : lance la présentation depuis le début et calcule son temps d'exécution.

Les propriétés suivantes sont également disponibles :

- **AllowAnimations (Boolean)** : exécute les animations de la présentation.
- **CustomShow (String)** : permet de spécifier le nom de la présentation afin de pouvoir y faire référence dans la présentation.
- **FirstPage (String)** : nom de la première diapo qui servira au lancement de la présentation.
- **IsAlwaysOnTop (Boolean)** : affiche toujours la fenêtre de la présentation au premier plan.
- **IsAutomatic (Boolean)** : exécute la présentation automatiquement.
- **IsEndless (Boolean)** : une fois la présentation terminée, la relance depuis le début.
- **IsFullScreen (Boolean)** : lance automatiquement la présentation en mode plein écran.
- **IsMouseVisible (Boolean)** : affiche la souris pendant la présentation.
- **Pause (long)** : durée pendant laquelle un écran noir s'affiche à la fin de la présentation.
- **StartWithNavigator (Boolean)** : affiche la fenêtre du navigateur au lancement de la présentation.
- **UsePn (Boolean)** : affiche le pointeur pendant la présentation.

CHAPITRE 9

Diagrammes

StarOffice peut afficher les données dans un diagramme : des liaisons graphiques sont alors créées entre les données sous forme de barres, de diagrammes à secteurs, de lignes et autres éléments. Les données peuvent être affichées sous forme d'images 2D ou 3D, et l'apparence des éléments du diagramme peut être adaptée individuellement de la même manière que pour les éléments de dessin.

Si les données sont disponibles dans une feuille de calcul, elles peuvent être liées au diagramme de façon dynamique. Toute modification apportée aux données de base peut, dans le cas présent, être immédiatement visualisée dans le diagramme assigné. Ce chapitre présente l'interface de programmation des modules de diagramme de StarOffice et s'intéresse à l'utilisation des diagrammes au sein des classeurs.

Utilisation de diagrammes dans les classeurs

Les diagrammes ne sont pas traités comme des documents indépendants dans StarOffice, mais comme les objets incorporés d'un document existant.

Les diagrammes des documents texte et de dessin restent isolés du contenu du document ; en revanche, lorsqu'ils sont utilisés dans les classeurs, un mécanisme permet d'établir une liaison entre les données du document et les diagrammes incorporés. L'exemple suivant explique l'interaction entre un classeur et un diagramme :

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
```

```
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

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

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
```

Bien que le code utilisé dans cet exemple puisse paraître complexe, les processus centraux sont limités à trois lignes : la première ligne centrale crée la variable de document `Doc`, qui fait référence au classeur actif (ligne `Doc = StarDesktop.CurrentComponent`). Le code utilisé dans l'exemple crée ensuite une liste contenant tous les diagrammes de la première feuille de calcul (ligne `Charts = Doc.Sheets(0).Charts`). Enfin, un nouveau diagramme est ajouté à la dernière ligne de cette liste à l'aide de la méthode `addNewByName`. L'utilisateur peut ensuite visualiser ce nouveau diagramme.

La dernière ligne initialise les structures auxiliaires `Rect` et `RangeAddress`, que la méthode `addNewByName` fournit également comme paramètre. `Rect` détermine la position du diagramme dans la feuille de calcul. `RangeAddress` détermine la plage de données à lier au diagramme.

L'exemple précédent crée un diagramme à barres. Si vous souhaitez créer un autre type d'image, vous devez remplacer le diagramme à barres de façon explicite :

```
Chart = Charts.getByName("MyChart").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

La première ligne définit l'objet Chart correspondant. La seconde remplace le diagramme actif par un nouveau : dans cet exemple, un diagramme linéaire.

Remarque – Dans Excel, une distinction est faite entre les diagrammes ayant été insérés comme une page séparée dans un document Excel et ceux qui sont incorporés dans une page de table. Par conséquent, deux méthodes d'accès différentes sont définies ici pour les diagrammes. Cette distinction n'existe pas dans StarOffice Basic, car les diagrammes de StarOffice Calc sont toujours créés en tant qu'objets incorporés d'une page de table. L'accès aux diagrammes s'effectue toujours à l'aide de la liste `Charts` de l'objet `Sheet` associé.

Structure des diagrammes

La structure d'un diagramme, et donc de la liste des services et interfaces qu'il prend en charge, dépend de son type. Les méthodes et propriétés de l'axe Z, par exemple, sont uniquement disponibles dans les diagrammes 3D, mais pas dans les diagrammes 2D. Dans les diagrammes à secteurs, il n'existe aucune interface permettant d'utiliser des axes.

Éléments individuels d'un diagramme

Titre, sous-titre et légende

Les titre, sous-titre et légende constituent les éléments de base de chaque diagramme. Les diagrammes possèdent leurs propres objets pour chacun de ces éléments. L'objet `Chart` offre les propriétés suivantes de gestion de ces éléments :

- **HasMainTitle (Boolean)** : active le titre.
- **Title (Object)** : objet contenant des informations détaillées sur le titre du diagramme (prend en charge le service `com.sun.star.chart.ChartTitle`).
- **HasSubTitle(Boolean)** : active le sous-titre.
- **Subtitle (Object)** : objet contenant des informations détaillées sur le sous-titre du diagramme (prend en charge le service `com.sun.star.chart.ChartTitle`).
- **HasLegend (Boolean)** : active la légende.
- **Legend (Object)** : objet contenant des informations détaillées sur la légende du diagramme (prend en charge le service `com.sun.star.chart.ChartLegendPosition`).

À de nombreux égards, les éléments spécifiés correspondent à un élément de dessin. Cela est dû au fait que les services `com.sun.star.chart.ChartTitle` et `com.sun.star.chart.ChartLegendPosition` prennent en charge le service `com.sun.star.drawing.Shape`, qui constitue la base technique du programme des éléments de dessin.

Les utilisateurs ont donc la possibilité de déterminer la position et la taille de l'élément à l'aide des propriétés `Size` et `Position`.

Les propriétés de remplissage et de ligne (services `com.sun.star.drawing.FillProperties` et `com.sun.star.drawing.LineStyle`), ainsi que les propriétés de caractère (service `com.sun.star.style.CharacterProperties`), sont fournies pour le formatage des éléments.

com.sun.star.chart.ChartTitle contient non seulement les propriétés de format nommées, mais également deux autres propriétés :

- **TextRotation (Long)** : angle de rotation du texte en centièmes de degré.
- **String (String)** : texte à afficher comme titre ou sous-titre.

La légende du diagramme (service com.sun.star.chart.ChartLegend) contient la propriété supplémentaire suivante :

- **Alignment (Enum)** : position de la légende (valeur par défaut correspondant à com.sun.star.chart.ChartLegendPosition).

L'exemple suivant crée un diagramme et lui assigne le titre 'Test', le sous-titre 'Test 2' et une légende. La légende, dont la couleur d'arrière-plan est grise, se trouve au bas du diagramme et a une taille de caractère de 7 points.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Doc = StarDesktop.CurrentComponent
Charts = Doc.Sheets(0).Charts
Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Chart.HasMainTitle = True
Chart.Title.String = "Test"

Chart.HasSubTitle = True
Chart.Subtitle.String = "Test 2"

Chart.HasLegend = True
Chart.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Chart.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Chart.Legend.FillColor = RGB(210, 210, 210)
Chart.Legend.CharHeight = 7
```

Arrière-plan

Chaque diagramme présente une zone d'arrière-plan. Chaque zone possède un objet, accessible en utilisant les propriétés suivantes de l'objet Diagram :

- **Area (Object)** : zone d'arrière-plan du diagramme (prend en charge le service `com.sun.star.chart.ChartArea`).

L'arrière-plan d'un diagramme recouvre l'ensemble de sa zone, y compris la zone située sous le titre, le sous-titre et la légende du diagramme. Le service `com.sun.star.chart.ChartArea` associé prend en charge les propriétés de ligne et de remplissage, et n'offre aucune autre propriété extensive.

Parois et planchers du diagramme

Bien que l'arrière-plan du diagramme recouvre la zone complète du diagramme, la paroi arrière du diagramme se limite à la zone se trouvant directement derrière la plage de données.

Il existe généralement deux parois dans les diagrammes 3D : l'une se situe derrière la plage de données et l'autre sert de démarcation à gauche de l'axe Y. De plus, les diagrammes 3D ont généralement un plancher.

- **Floor (Object)** : plancher du diagramme (uniquement pour les diagrammes 3D, prend en charge le service `com.sun.star.chart.ChartArea`).
- **Wall (Object)** : parois du diagramme (uniquement pour les diagrammes 3D, prend en charge le service `com.sun.star.chart.ChartArea`).

Les objets spécifiés prennent en charge le service `com.sun.star.chart.ChartArea`, qui offre à son tour les propriétés habituelles de remplissage et de ligne via les services `com.sun.star.drawing.FillProperties` et `com.sun.star.drawing.LineStyle`. Pour plus d'informations, reportez-vous au Chapitre 8.

L'accès aux parois et au plancher du diagramme s'effectue par l'intermédiaire de l'objet `Chart`, qui fait également partie de l'objet Chart :

```
Charte.Area.FillBitmapName = 'Sky'
```

L'exemple suivant montre comment une image (nommée Sky) déjà contenue dans StarOffice peut servir d'arrière-plan à un diagramme.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

Rect.X = 8000
```

```
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

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

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject

Charte.Area.FillStyle = com.sun.star.drawing.FillStyle.BITMAP
Chart.Area.FillBitmapName = 'Sky'
Chart.Area.FillBitmapMode = com.sun.star.drawing.BitmapMode.REPEAT
```

Axes

StarOffice reconnaît cinq axes différents pouvant être utilisés dans un diagramme. Dans le scénario le plus simple, il s'agit des axes X et Y. Lorsque vous utilisez des diagrammes 3D, un axe Z est parfois ajouté. Pour les diagrammes dans lesquels les valeurs des différentes lignes dévient de façon significative l'une de l'autre, StarOffice fournit des X et Y supplémentaires pour des opérations de mises à l'échelle secondaires.

Axes X, Y et Z primaires

Outre l'axe réel, il peut également y avoir, pour chacun des axes X, Y et Z primaires, un titre, une description, une grille principale et une grille secondaire. Il existe une option permettant d'afficher et de masquer tous ces éléments. L'objet Diagram offre les propriétés de gestion suivantes pour ces fonctionnalités (en prenant l'exemple d'un axe X, les propriétés des axes Y et Z étant structurées de la même manière) :

- **HasXAxis (Boolean)** : active l'axe X.
- **XAxis (Object)** : objet contenant des informations détaillées sur l'axe X (prend en charge le service `com.sun.star.chart.ChartAxis`).
- **HasXAxisDescription (Boolean)** : active la description de l'axe X.
- **HasXAxisGrid (Boolean)** : active la grille principale de l'axe X.
- **XMainGrid (Object)** : objet contenant des informations détaillées sur la grille principale de l'axe X (prend en charge le service `com.sun.star.chart.ChartGrid`).
- **HasXAxisHelpGrid (Boolean)** : active la grille secondaire de l'axe X.

- **XHelpGrid (Object)** : objet contenant des informations détaillées sur la grille secondaire de l'axe X (prend en charge le service `com.sun.star.chart.ChartGrid`).
- **HasXAxisTitle (Boolean)** : active le titre de l'axe X.
- **XAxisTitle (Object)** : objet contenant des informations détaillées sur le titre de l'axe X (prend en charge le service `com.sun.star.chart.ChartTitle`).

Axes X et Y secondaires

Les propriétés suivantes sont disponibles pour les axes X et Y secondaires (propriétés prenant exemple sur l'axe secondaire X) :

- **HasSecondaryXAxis (Boolean)** : active l'axe secondaire X.
- **SecondaryXAxis (Object)** : objet contenant des informations détaillées sur l'axe secondaire X (prend en charge le service `com.sun.star.chart.ChartAxis`).
- **HasSecondaryXAxisDescription (Boolean)** : active la description du titre de l'axe X.

Propriétés des axes

Les objets Axis d'un diagramme StarOffice prennent en charge le service `com.sun.star.chart.ChartAxis`. Outre les propriétés de caractère (`service com.sun.star.style.CharacterProperties` : reportez-vous au Chapitre 6) et de ligne (service `com.sun.star.drawing.LineStyle` : reportez-vous au chapitre 8, Chapitre 8), il offre les propriétés suivantes :

- **Max (Double)** : valeur maximale de l'axe.
- **Min (Double)** : valeur minimale de l'axe.
- **Origin (Double)** : point d'intersection des axes.
- **StepMain (Double)** : distance entre deux lignes principales de l'axe.
- **StepHelp (Double)** : distance entre deux lignes secondaires de l'axe.
- **AutoMax (Boolean)** : détermine automatiquement la valeur maximale de l'axe.
- **AutoMax (Boolean)** : détermine automatiquement la valeur minimale de l'axe.
- **AutoOrigin (Boolean)** : détermine automatiquement le point d'intersection des axes.
- **AutoStepMain (Boolean)** : détermine automatiquement la distance entre deux lignes principales d'un axe.
- **AutoStepHelp (Boolean)** : détermine automatiquement la distance entre deux lignes secondaires d'un axe.
- **Logarithmic (Boolean)** : ajuste les axes de manière logarithmique (et non de manière linéaire).

- **DisplayLabels (Boolean)** : active l'étiquette de texte des axes.
- **TextRotation (Long)** : angle de rotation de l'étiquette de texte des axes en centièmes de degré.
- **Marks (Const)** : constante spécifiant si les lignes principales de l'axe doivent se trouver à l'intérieur ou à l'extérieur de la zone du diagramme (les valeurs par défaut correspondent à `com.sun.star.chart.ChartAxisMarks`).
- **HelpMarks (Const)** : constante qui spécifiant si les lignes secondaires de l'axe doivent se trouver à l'intérieur ou à l'extérieur de la zone du diagramme (les valeurs par défaut correspondent à `com.sun.star.chart.ChartHelpMarks`).
- **Overlap (Long)** : pourcentage spécifiant l'étendue de superposition des barres des différents jeux de données. À 100 %, les barres s'affichent en superposition complète, à - 100 %, une distance égale à la largeur d'une barre les sépare.
- **GapWidth (long)** : pourcentage spécifiant la distance possible entre les différents groupes de barres d'un diagramme. À 100 %, la distance correspond à la largeur d'une barre.
- **ArrangeOrder (enum)** : détails de la position de l'inscription ; outre le positionnement sur une ligne, il est également possible de scinder l'étiquette sur deux lignes (valeur par défaut correspondant à `com.sun.star.chart.ChartAxisArrangeOrderType`).
- **TextBreak (Boolean)** : autorise les retours à la ligne.
- **TextCanOverlap (Boolean)** : autorise les chevauchements de texte.
- **NumberFormat (Long)** : format numérique (voir la section "Formats de nombre, de date et de texte" à la page 134).

Propriétés de la grille de l'axe

L'objet de la grille de l'axe est basé sur le service `com.sun.star.chart.ChartGrid`, qui prend en charge les propriétés de ligne du service de support `com.sun.star.drawing.LineStyle`. Pour plus d'informations, reportez-vous au Chapitre 8.

Propriétés du titre de l'axe

Les objets de formatage du titre de l'axe sont basés sur le service `com.sun.star.chart.ChartTitle`, également utilisé pour les titres des diagrammes.

Exemple

L'exemple suivant crée un diagramme linéaire. La couleur blanche est définie pour la paroi arrière du diagramme. Les axes X et Y possèdent une grille secondaire grise pour un meilleur repérage visuel. La valeur minimale de l'axe Y est fixée à 0 et sa valeur maximale à 100, de manière à conserver la résolution du diagramme même en cas de modification des valeurs.

```
Dim Doc As Object Dim Charts As Object Dim Chart as Object

Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress

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

Rect.X = 8000 Rect.Y = 1000 Rect.Width = 10000 Rect.Height = 7000

RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12

Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)

Chart = Charts.getByName('MyChart').embeddedObject
Chart.Diagram = Chart.createInstance('com.sun.star.chart.LineDiagram')

Chart.Diagram.Wall.FillColor = RGB(255, 255, 255)

Chart.Diagram.HasXAxisGrid = True
Chart.Diagram.XMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.HasYAxisGrid = True
Chart.Diagram.YMainGrid.LineColor = RGB(192, 192, 192)

Chart.Diagram.YAxis.Min = 0
Chart.Diagram.YAxis.Max = 100
```

Diagrammes 3D

La plupart des diagrammes de StarOffice peuvent également s'afficher avec des images 3D. Tous les types de diagrammes offrant cette option prennent en charge le service `com.sun.star.chart.Dim3DDiagram`. Ce service n'offre qu'une propriété :

- **Dim3D (Boolean)** : active l'affichage en 3D.

Diagrammes empilés

Les diagrammes empilés sont des diagrammes organisés avec plusieurs valeurs individuelles empilées pour produire une valeur totale. Cette vue montre non seulement les valeurs individuelles, mais également une présentation de toutes les valeurs.

Dans StarOffice, il est possible d'afficher plusieurs types de diagrammes sous forme empilée. Tous ces diagrammes prennent en charge le service `com.sun.star.chart.StackableDiagram`, qui fournit quant à lui les propriétés suivantes :

- **Stacked (Boolean)** : active le mode de visualisation des éléments empilés.
- **Percent (Boolean)** : affiche la répartition en pourcentage plutôt qu'en valeurs absolues.

Types de diagrammes

Diagrammes linéaires

Les diagrammes linéaires (service `com.sun.star.chart.LineDiagram`) prennent en charge un axe X, deux axes Y et un axe Z. Ils peuvent s'afficher sous forme d'images 2D ou 3D (service `com.sun.star.chart.Dim3Ddiagram`). Les lignes peuvent être empilées (`com.sun.star.chart.StackableDiagram`).

Les diagrammes linéaires offrent les propriétés suivantes :

- **SymbolType (const)** : symbole pour l'affichage des points de données (constante correspondant à `com.sun.star.chart.ChartSymbolType`).
- **SymbolSize (Long)** : taille du symbole d'affichage des points de données, exprimée en centièmes de millimètre.
- **SymbolBitmapURL (String)** : nom des fichiers d'images utilisées pour afficher les points de données.
- **Lines (Boolean)** : relie les points de données à l'aide de lignes.
- **SplineType (Long)** : fonction spline de lissage des lignes (0 : pas de fonction spline, 1 : splines cubiques, 2 : splines B).
- **SplineOrder (Long)** : poids polynomial des splines (pour les splines B uniquement).
- **SplineResolution (Long)** : nombre de points de support pour le calcul des splines.

Diagrammes de surface

Les diagrammes de surface (service `com.sun.star.chart.AreaDiagram`) prennent en charge un axe X, deux axes Y et un axe Z. Ils peuvent s'afficher sous forme d'images 2D ou 3D (service `com.sun.star.chart.Dim3Ddiagram`). Les surfaces peuvent être empilées (`com.sun.star.chart.StackableDiagram`).

Diagrammes à barres

Les diagrammes à barres (service `com.sun.star.chart.BarDiagram`) prennent en charge un axe X, deux axes Y et un axe Z. Ils peuvent s'afficher sous forme d'images 2D ou 3D (service `com.sun.star.chart.Dim3Ddiagram`). Les barres peuvent être empilées (`com.sun.star.chart.StackableDiagram`).

Ils offrent les propriétés suivantes :

- **Vertical (Boolean)** : affiche les barres verticalement.
- **Deep (Boolean)** : en mode visualisation 3D, positionne les barres les unes derrière les autres plutôt que les unes à côté des autres.
- **StackedBarsConnected (Boolean)** : relie les barres associées dans un diagramme empilé à l'aide de lignes (uniquement dans les diagrammes horizontaux).
- **NumberOfLines (Long)** : nombre de lignes à afficher dans un diagramme empilé sous forme de lignes plutôt que sous forme de barres.

Diagrammes à secteurs

Les diagrammes à secteurs (service `com.sun.star.chart.PieDiagram`) ne contiennent pas d'axes et ne peuvent pas être empilés. Ils peuvent s'afficher sous forme d'images 2D ou 3D (service `com.sun.star.chart.Dim3Ddiagram`).

CHAPITRE 10

Accès aux bases de données

StarOffice possède une interface de base de données intégrée (indépendante de tout système) nommée Star Database Connectivity (SDBC). Cette interface a été développée dans le but d'offrir l'accès au plus grand nombre possible de sources de données différentes.

Dans cet objectif, l'accès aux sources de données s'effectue via des pilotes. Les sources à partir desquelles les pilotes récupèrent leurs données n'ont pas d'importance pour un utilisateur SDBC. Certains pilotes accèdent aux bases de données de fichiers et y récupèrent directement les données. D'autres utilisent des interfaces standard telles que JDBC ou ODBC. Toutefois, il existe également des pilotes spéciaux qui utilisent le carnet d'adresses MAPI, les annuaires LDAP ou les feuilles de calcul StarOffice comme sources de données.

Les pilotes étant basés sur des composants UNO, il est possible de développer d'autres pilotes et, par conséquent, d'ouvrir de nouvelles sources de données. Le document StarOffice Developer's Guide contient de plus amples informations à ce sujet.

Remarque – En termes de concept, SDBC est comparable aux bibliothèques ADO et DAO disponibles dans VBA. Il permet l'accès de haut niveau aux bases de données, quels que soient les serveurs de base de données sous-jacents.

Remarque – L'interface de base de données de StarOffice s'est développée lors du lancement de StarOffice 8. Par le passé, on accédait aux bases de données principalement à l'aide d'un ensemble de méthodes de l'objet `Application`, mais l'interface de StarOffice 7 se subdivise en plusieurs objets. Un service `DatabaseContext` sert d'objet racine pour les fonctions de base de données.

SQL : un langage dédié aux requêtes

Le langage SQL permet aux utilisateurs de SDBC d'effectuer des requêtes. Pour comparer les différences entre les divers langages SQL, les composants SDBC de StarOffice ont leur propre analyseur SQL. La fenêtre de requête permet de vérifier les commandes SQL saisies et de corriger les erreurs de syntaxe simples, comme celles qui sont associées aux caractères majuscules et minuscules.

Si un pilote autorise l'accès à une source de données qui ne prend pas en charge SQL, il doit convertir indépendamment les commandes SQL transférées vers l'accès natif requis.

Remarque – L'implémentation SQL à partir de SDBC est orientée vers la norme SQL-ANSI. Les extensions spécifiques à Microsoft, telles que la construction `INNER JOIN` ne sont pas prises en charge. Elles doivent être remplacées par des commandes standard (ainsi `INNER JOIN` devra être remplacée par la clause `WHERE` correspondante).

Types d'accès aux bases de données

L'interface de base de données de StarOffice est disponible dans les applications StarOffice Writer et StarOffice Calc, ainsi que dans les formulaires de base de données.

Dans StarOffice Writer, les lettres standard peuvent être créées à l'aide de sources de données SDBC puis imprimées. Il existe également une option permettant de déplacer les données de la fenêtre de la base de données dans des documents texte à l'aide de la fonction glisser-déposer.

Si l'utilisateur insère une table de base de données dans une feuille de calcul, StarOffice crée une zone de table qui peut être mise à jour par un clic de souris si les données d'origine ont été modifiées. À l'inverse, il est possible de déplacer les données d'une feuille de calcul vers une table de base de données et d'importer une base de données.

Enfin, StarOffice offre un mécanisme pour les formulaires basés sur des bases de données. Pour cela, l'utilisateur doit d'abord créer un formulaire standard StarOffice Writer ou StarOffice Calc, puis lier les champs à une base de données.

Toutes les options indiquées ici sont basées sur l'interface utilisateur de StarOffice. Aucune connaissance en programmation n'est nécessaire pour utiliser les fonctions correspondantes.

Cependant, ce chapitre fournit peu d'informations concernant les fonctions spécifiées : il concerne principalement l'interface de programmation à partir de SDBC, qui permet l'automatisation des requêtes aux bases de données et offre de ce fait une plus grande variété d'applications.

Il est toutefois nécessaire de posséder une connaissance élémentaire du fonctionnement des bases de données et du langage de requête SQL pour bien comprendre les sections qui suivent.

Sources de données

Une base de données peut être intégrée à StarOffice en créant ce que l'on nomme couramment une *source de données*. L'interface utilisateur offre une option correspondante pour créer des sources de données dans le menu **Extras**. Vous pouvez également créer des sources de données pour les utiliser dans StarOffice Basic.

Un objet de contexte de base de données créé à l'aide de la fonction `createUnoService` sert de point de départ pour accéder à une source de données. Il est basé sur le service `com.sun.star.sdb.DatabaseContext` et sert d'objet racine pour toutes les opérations de la base de données.

L'exemple suivant montre comment un objet de contexte de base de données peut être créé, puis utilisé pour déterminer les noms de toutes les sources de données disponibles. Il affiche les noms dans une boîte de message.

```
Dim DatabaseContext As Object
Dim Names
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")

Names = DatabaseContext.getElementNames()
For I = 0 To UBound(Names())
   MsgBox Names(I)
Next I
```

Les sources de données individuelles sont basées sur le service `com.sun.star.sdb.DataSource` et peuvent être déterminées à partir du contexte de la base de données en utilisant la méthode `getByName` :

```
Dim DatabaseContext As Object
Dim DataSource As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
```

Cet exemple crée un objet `DataSource` pour une source de données nommée *Customers*.

Les sources de données offrent un ensemble de propriétés qui, à leur tour, fournissent des renseignements d'ordre général concernant l'origine des données et des informations concernant les méthodes d'accès. Ces propriétés sont les suivantes :

- **Name (String)** : nom de la source de données.
- **URL (String)** : URL de la source de données, au format *jdbc: subprotocol : subname* ou *sdbc: subprotocol : subname*.
- **Info (Array)** : matrice contenant des paires `PropertyValue` avec des paramètres de connexion (généralement un nom d'utilisateur et un mot de passe).
- **User (String)** : nom d'utilisateur.
- **Password (String)** : mot de passe de l'utilisateur (il n'est pas enregistré).
- **IsPasswordRequired (Boolean)** : un mot de passe doit obligatoirement être fourni par l'utilisateur.
- **IsReadOnly (Boolean)** : permet l'accès en lecture seule à la base de données.
- **NumberFormatsSupplier (Object)** : objet contenant les formats numériques pour la base de données (prend en charge l'interface `com.sun.star.util.XNumberFormatsSupplier`, voir section "Formats de nombre, de date et de texte" à la page 134).
- **TableFilter (Array)** : liste des noms de table à afficher.
- **TableTypeFilter (Array)** : liste des types de table à afficher. Les valeurs disponibles sont `TABLE`, `VIEW` et `SYSTEM TABLE`.
- **SuppressVersionColumns (Boolean)** : masque les colonnes utilisées pour la gestion des versions.

Remarque – Les sources de données de StarOffice ne sont pas comparables une à une avec les sources de données ODBC. Une source de données ODBC ne recouvre que les informations concernant l'origine des données, tandis qu'une source de données de StarOffice inclut également un ensemble d'informations relatives à l'affichage des données dans les fenêtres de base de données de StarOffice.

Requêtes

Il est possible d'assigner des requêtes prédéfinies à une source de données. StarOffice note les commandes SQL des requêtes afin qu'elles soient disponibles à tout moment. Les requêtes simplifient l'utilisation des bases de données car elles peuvent s'ouvrir d'un simple clic de souris. Elles permettent ainsi aux novices d'employer les commandes SQL.

Un objet prenant en charge le service `com.sun.star.sdb.QueryDefinition` se trouve derrière une requête. L'accès aux requêtes s'effectue grâce à la méthode `QueryDefinitions` de la source de données.

L'exemple suivant énumère les noms des requêtes à la source de données qui peuvent être établies dans un message.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

For I = 0 To QueryDefinitions.Count() - 1
   QueryDefinition = QueryDefinitions(I)
   MsgBox QueryDefinition.Name
Next I
```

Outre la propriété Name utilisée dans cet exemple, `com.sun.star.sdb.QueryDefinition` offre de nombreuses autres propriétés. Ces méthodes sont les suivantes :

- **Name (String)** : nom de la requête.
- **Command (String)** : commande SQL (en règle générale, une commande `SELECT`).
- **UpdateTableName (String)** : pour les requêtes basées sur plusieurs tables : nom de la table dans laquelle des modifications de valeur sont possibles.
- **UpdateCatalogName (String)** : nom des catalogues de tables actualisables.
- **UpdateSchemaName (String)** : nom des diagrammes de tables actualisables.

L'exemple qui suit montre comment un objet Query peut être créé par programmation et assigné à une source de données.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim QueryDefinitions As Object
Dim QueryDefinition As Object
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
QueryDefinitions = DataSource.getQueryDefinitions()

QueryDefinition = createUnoService("com.sun.star.sdb.QueryDefinition")
QueryDefinition.Command = "SELECT * FROM Customer"

QueryDefinitions.insertByName("NewQuery", QueryDefinition)
```

L'objet Query est d'abord créé en utilisant l'appel `createUnoService`, initialisé, puis inséré dans l'objet `QueryDefinitions` à l'aide de `insertByName`.

Liaisons avec des formulaires de base de données

Pour simplifier l'utilisation des sources de base de données, StarOffice fournit une option permettant de lier les sources de données et les formulaires de base de données. Ces liaisons sont disponibles par l'intermédiaire de la

méthode`getBookmarks()`. Celle-ci retourne un conteneur nommé (`com.sun.star.sdb.DefinitionContainer`) contenant toutes les liaisons de la source de données. Les repères de texte sont accessibles par l'intermédiaire de `Name` ou `Index`.

L'exemple suivant détermine l'URL du repère de texte *MyBookmark*.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Bookmarks As Object
Dim URL As String
Dim I As Integer

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")
Bookmarks = DataSource.Bookmarks()

URL = Bookmarks.getByName("MyBookmark")
MsgBox URL
```

Accès à la base de données

Pour accéder à une base de données, il faut y être connecté. Une voie de transfert est donc utilisée pour permettre la communication directe avec la base de données. Contrairement aux sources de données présentées dans la section précédente, la connexion à la base de données doit donc être rétablie à chaque fois que le programme est relancé.

StarOffice offre différentes manières d'établir des connexions à la base de données. La méthode basée sur une source de données existante est illustrée ci-dessous.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If
```

Le code utilisé dans cet exemple commence par vérifier si la base de données est protégée par mot de passe. Dans le cas contraire, il crée la connexion à la base de données requise à l'aide de l'appel `GetConnection`. Les deux chaînes vides de la ligne de commande correspondent au nom et au mot de passe de l'utilisateur.

Si la base de données est protégée par mot de passe, l'exemple crée un objet `InteractionHandler` et ouvre la connexion à la base de données à l'aide la méthode `ConnectWithCompletion`. L'InteractionHandler garantit que StarOffice demande les données de connexion requises à l'utilisateur.

Itération de tables

Dans StarOffice, pour accéder à une table, on utilise généralement l'objet `ResultSet`. Un objet `ResultSet` est un ensemble courant de données issues d'un volume de résultats, obtenu à l'aide de la commande `SELECT`.

L'exemple montre l'utilisation d'un objet `ResultSet` pour interroger les valeurs à partir d'une table de base de données.

```
Dim DatabaseContext As Object
Dim DataSource As Object
Dim Connection As Object
Dim InteractionHandler as Object
Dim Statement As Object
Dim ResultSet As Object

DatabaseContext = createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource = DatabaseContext.getByName("Customers")

If Not DataSource.IsPasswordRequired Then
   Connection = DataSource.GetConnection("","")
Else
   InteractionHandler = createUnoService("com.sun.star.sdb.InteractionHandler")
   Connection = DataSource.ConnectWithCompletion(InteractionHandler)
End If

Statement = Connection.createStatement()
ResultSet = Statement.executeQuery("SELECT CustomerNumber FROM Customer")

If Not IsNull(ResultSet) Then
   While ResultSet.next
      MsgBox ResultSet.getString(1)
   Wend
End If
```

Une fois la connexion à la base de données établie, l'exemple crée un objet `Statement` via l'appel `Connection.createStatement`. Cet objet `Statement` utilise ensuite l'appel `executeQuery` pour retourner l'objet `ResultSet` courant. Le programme vérifie alors si l'objet `ResultSet` existe et, le cas échéant, parcourt les enregistrements de données à l'aide d'une boucle. Les valeurs requises (dans notre exemple, celles du champ `CustomerNumber`) retournent l'objet `ResultSet` en utilisant la méthode `getString`, selon laquelle le paramètre 1 détermine que l'appel se relie aux valeurs de la première colonne.

Remarque – L'objet `ResultSet` de SDBC est comparable à l'objet `Recordset` de DAO et ADO, car il offre également l'accès itératif à une base de données.

Remarque – L'accès à la base de données s'effectue en réalité dans StarOffice 8 par l'intermédiaire d'un objet `ResultSet`. Celui-ci reflète le contenu d'une table ou le résultat d'une commande SQL-SELECT. Dans les versions précédentes, la navigation au sein des données s'effectuait grâce aux méthodes résidentes telles que `DataNextRecord` fournies par l'objet `ResultSet` dans l'objet `Application`.

Méthodes de récupération des valeurs en fonction du type

Comme le montre l'exemple de la section précédente, StarOffice offre une méthode `getString` pour accéder au contenu des tables. Cette méthode retourne le résultat sous la forme d'une chaîne. Les méthodes `get` suivantes sont disponibles :

- `getByte()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getShort()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getInt()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getLong()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getFloat()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getDouble()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getBoolean()` : prend en charge les types de données SQL dédiés aux nombres, aux caractères et aux chaînes.
- `getString()` : prend en charge tous les types de données SQL.
- `getBytes()` : prend en charge les types de données SQL dédiés aux valeurs binaires.
- `getDate()` : prend en charge les types de données SQL dédiés aux nombres, aux chaînes, à la date et à l'heure.
- `getTime()` : prend en charge les types de données SQL dédiés aux nombres, aux chaînes, à la date et à l'heure.
- `getTimestamp()` : prend en charge les types de données SQL dédiés aux nombres, aux chaînes, à la date et à l'heure.

- `getCharacterStream()` : prend en charge les types de données SQL dédiés aux nombres, aux chaînes et aux valeurs binaires.
- `getUnicodeStream()` : prend en charge les types de données SQL dédiés aux nombres, aux chaînes et aux valeurs binaires.
- `getBinaryStream()` : valeurs binaires.
- `getObject()` : prend en charge tous les types de données SQL.

Dans tous les cas, le nombre de colonnes doit apparaître comme un paramètre dont les valeurs doivent être interrogées.

Variantes de l'objet ResultSet

La rapidité d'accès aux bases de données est souvent critique. C'est la raison pour laquelle StarOffice offre plusieurs moyens pour optimiser les objets `ResultSet` et ainsi contrôler la vitesse d'accès. Plus un objet `ResultSet` offre de fonctions, plus son implémentation est généralement complexe et plus lentes sont les fonctions.

Un objet `ResultSet` simple, tel que celui présenté à la section 'Itération de tables', offre les fonctions minimales. Il ne permet l'application de l'itération que vers l'avant, et pour les valeurs à interroger. Par conséquent, d'autres options de navigation plus complexes, telles que la modification des valeurs, ne sont pas incluses.

L'objet Statement utilisé pour créer l'objet `ResultSet` offre certaines propriétés qui permettent d'influer sur les fonctions de celui-ci :

- **ResultSetConcurrency (const)** : spécifications selon lesquelles les données peuvent être modifiées ou non (spécifications correspondant à `com.sun.star.sdbc.ResultSetConcurrency`).
- **ResultSetType (const)** : spécifications relatives au type de `ResultSet` (spécifications correspondant à `com.sun.star.sdbc.ResultSetType`).

Les valeurs définies dans `com.sun.star.sdbc.ResultSetConcurrency` sont les suivantes :

- **UPDATABLE** : l'objet `ResultSet` permet la modification des valeurs.
- **READ_ONLY** : l'objet `ResultSet` ne permet pas les modifications.

Le groupe de constantes `com.sun.star.sdbc.ResultSetConcurrency` fournit les spécifications suivantes :

- **FORWARD_ONLY** : l'objet `ResultSet` permet uniquement la navigation vers l'avant.
- **SCROLL_INSENSITIVE** : l'objet `ResultSet` permet tout type de navigation ; les changements des données d'origine ne sont toutefois pas notés.
- **SCROLL_SENSITIVE** : l'objet `ResultSet` permet tout type de navigation ; les changements des données d'origine modifient l'objet `ResultSet`.

Remarque – Un objet `ResultSet` contenant les propriétés `READ_ONLY` et `SCROLL_INSENSITIVE` correspond à un jeu d'enregistrements du type `Snapshot` dans ADO et DAO.

Lorsque vous utilisez les propriétés `UPDATEABLE` et `SCROLL_SENSITIVE` de l'objet `ResultSet`, celui-ci possède des fonctions d'étendue comparable à un objet `Recordset` de type `Dynaset` dans ADO et DAO.

Méthodes de navigation dans les objets ResultSets

Si un objet `ResultSet` est de type `SCROLL_INSENSITIVE` ou `SCROLL_SENSITIVE`, il prend en charge un ensemble de méthodes de navigation dans le stock de données. Les méthodes centrales sont les suivantes :

- **next()** : va à l'enregistrement de données suivant.
- **previous()** : va à l'enregistrement de données précédent.
- **first()** : va au premier enregistrement de données.
- **last()** : va au dernier enregistrement de données.
- **beforeFirst()** : se place avant le premier enregistrement de données.
- **afterLast** : se place après le dernier enregistrement de données.

Toutes les méthodes retournent un paramètre de type Boolean qui spécifie si la navigation a réussi ou non.

Pour déterminer la position courante du curseur, les méthodes de test suivantes sont fournies et toutes retournent une valeur de type Boolean :

- **isBeforeFirst()** : l'objet `ResultSet` se situe avant le premier enregistrement de données.
- **isAfterLast()** : l'objet `ResultSet` se situe après le premier enregistrement de données.
- **isFirst()** : l'objet `ResultSet` est le premier enregistrement de données.
- **isLast()** : l'objet `ResultSet` est le dernier enregistrement de données.

Modification des enregistrements de données

Si un objet `ResultSet` a été créé avec la valeur `ResultSetConcurrency = UPDATEABLE`, son contenu est modifiable. Ceci s'applique seulement tant que la commande SQL permet la réécriture des données dans la base de données (dépend du principe). Par exemple, ceci n'est pas possible pour les commandes SQL complexes avec des colonnes liées ou des valeurs cumulées.

L'objet `ResultSet` fournit des méthodes `update` pour modifier les valeurs. Elles sont structurées de la même manière que les méthodes `get` qui permettent de récupérer des valeurs. La méthode `updateString`, par exemple, permet l'écriture d'une chaîne.

Après modification, les valeurs doivent être transférées dans la base de données à l'aide de la méthode`updateRow()`. L'appel doit intervenir avant la prochaine commande de navigation, sinon les valeurs sont perdues.

Une erreur faite pendant les modifications peut être annulée à l'aide de la méthode`cancelRowUpdates()`. Cet appel n'est possible que si les données n'ont pas été réécrites dans la base de données à l'aide de la méthode`updateRow()`.

CHAPITRE 11

Boîtes de dialogue

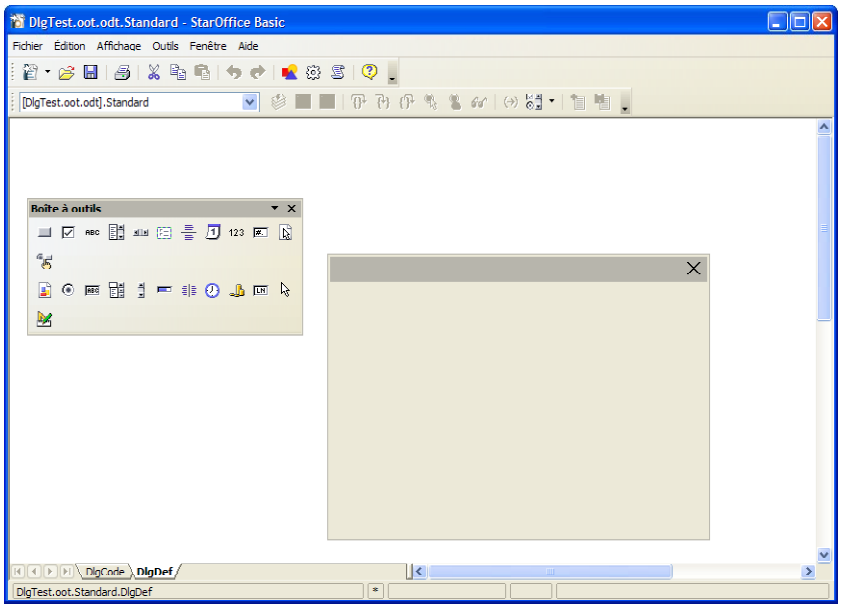
Vous pouvez ajouter des fenêtres de boîte de dialogue et des formulaires personnalisés aux documents StarOffice. Ceux-ci peuvent, à leur tour, être liés à des macros StarOffice Basic afin d'en étendre les possibilités d'utilisation de façon considérable. Les boîtes de dialogue peuvent, par exemple, afficher les informations d'une base de données ou guider les utilisateurs lors d'un processus étape par étape de création d'un nouveau document sous la forme d'un assistant.

Utilisation des boîtes de dialogue

Les boîtes de dialogue StarOffice Basic sont constituées d'une fenêtre comprenant des champs de texte, des zones de liste, des boutons radio et autres éléments de contrôle.

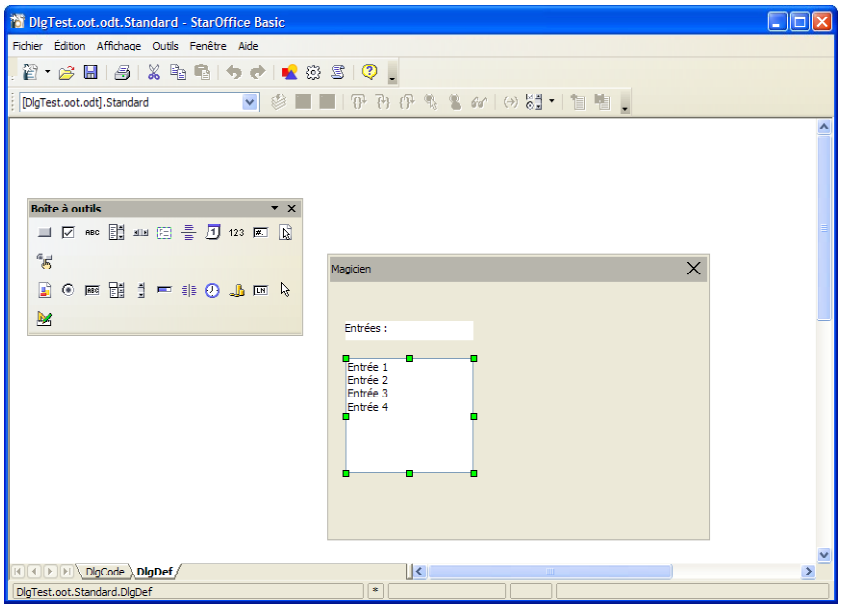
Création de boîtes de dialogue

Vous pouvez créer des boîtes de dialogue à l'aide de l'éditeur de boîte de dialogue StarOffice dont le fonctionnement est identique à celui l'application StarOffice Draw :



Faites glisser les éléments de contrôle de votre choix depuis la palette de conception (située à droite) vers la zone de la boîte de dialogue dans laquelle vous pourrez définir la position et la taille de ces éléments.

L'exemple illustre une boîte de dialogue contenant une étiquette et une zone de liste.



Vous pouvez ouvrir une boîte de dialogue à l'aide du code suivant :

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)

Dlg.Execute()
Dlg.dispose()
```

`CreateUnoDialog` crée un objet nommé `Dlg` qui fait référence à la boîte de dialogue associée. Avant de pouvoir créer une boîte de dialogue, vous devez vérifier que la bibliothèque qu'elle utilise (dans cet exemple, la bibliothèque `Standard`) est chargée. Dans le cas contraire, la méthode `LoadLibrary` accomplit cette tâche.

Une fois que l'objet de la boîte de dialogue `Dlg` a été initialisé, vous pouvez utiliser la méthode `Execute` pour afficher la boîte de dialogue. Les boîtes de dialogue telles que celle-ci sont décrites comme modales, car elles ne permettent aucune autre action du programme avant leur fermeture. Lorsque cette boîte de dialogue est ouverte, le programme reste dans l'appel `Execute`.

La méthode `dispose` à la fin du code approuve les ressources utilisées par la boîte de dialogue lorsque le programme se termine.

Fermeture des boîtes de dialogue

Fermeture par OK ou Annuler

Si une boîte de dialogue contient un bouton **OK** ou **Annuler**, elle se ferme automatiquement lorsque vous cliquez sur l'un d'entre eux. Vous trouverez de plus amples informations sur l'utilisation de ces boutons à la section Détails des éléments de contrôle des boîtes de dialogue, plus loin dans ce chapitre.

Si vous fermez une boîte de dialogue en cliquant sur le bouton **OK**, la méthode `Execute` retourne la valeur 1 ; sinon, c'est la valeur 0 qui est retournée.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
   MsgBox "Ok pressed"
Case 0
   MsgBox "Cancel pressed"
End Select
```

Fermeture par le bouton de fermeture de la barre de titre

Si vous le souhaitez, vous pouvez fermer une boîte de dialogue en cliquant sur le bouton de fermeture de la barre de titre de la fenêtre de la boîte de dialogue. Dans ce cas, la méthode `Execute` de la boîte de dialogue retourne la valeur 0, comme si vous aviez appuyé sur le bouton Annuler.

Fermeture avec un appel de programme explicite

Vous pouvez également fermer une fenêtre de boîte de dialogue avec la méthode `endExecute` :

```
Dlg.endExecute()
```

Accès à des éléments de contrôle individuels

Une boîte de dialogue peut contenir autant d'éléments de contrôle que nécessaire. Vous pouvez accéder à ces éléments par l'intermédiaire de la méthode `getControl` qui retourne le nom de l'élément de contrôle.

```
Dim Ctl As Object

Ctl = Dlg.getControl("MyButton")
```

```
Ctl.Label = "New Label"
```

Ce code détermine l'objet de l'élément de contrôle `MyButton`, puis initialise la variable d'objet `Ctl` avec une référence à l'élément. Enfin, le code définit la propriété `Label` de l'élément de contrôle sur la valeur `New Label`.

Remarque – StarOffice Basic fait la distinction entre les majuscules et les minuscules pour les noms des éléments de contrôle.

Utilisation du *modèle* des boîtes de dialogue et des éléments de contrôle

La séparation entre les éléments visibles du programme (*Vue*) et les données ou les documents sous-jacents (*Modèle*) se produit à de nombreux endroits dans l'API StarOffice. Outre les méthodes et les propriétés des éléments de contrôle, les objets des boîtes de dialogue et des éléments de contrôle ont un objet `Model` subordonné. Cet objet vous permet d'accéder directement au contenu d'une boîte de dialogue ou d'un élément de contrôle.

Dans les boîtes de dialogue, la distinction entre les données et leur représentation n'est pas toujours aussi claire que dans d'autres zones de l'API StarOffice. Les éléments de l'API sont disponibles via les objets View et Model.

La propriété `Model` offre un accès contrôlé par programme au modèle des objets de boîte de dialogue et d'élément de contrôle.

```
Dim cmdNext As Object

cmdNext = Dlg.getControl("cmdNext")
cmdNext.Model.Enabled = False
```

Cet exemple désactive le bouton `cmdNtext` de la boîte de dialogue `Dlg` à l'aide de l'objet du modèle à partir de `cmdNtext`.

Propriétés

Nom et titre

Chaque élément de contrôle possède son nom propre et peut être interrogé à l'aide de la propriété de modèle suivante :

- **Model.Name (String)** : nom de l'élément de contrôle.

Vous pouvez spécifier le titre apparaissant dans la barre de titre d'une boîte de dialogue à l'aide de la propriété de modèle suivante :

- **Model.Title (String)** : titre de la boîte de dialogue (s'applique uniquement aux boîtes de dialogue).

Position et taille

Vous pouvez interroger la taille et la position d'un élément de contrôle à l'aide des propriétés suivantes de l'objet Model :

- **Model.Height (long)** : hauteur de l'élément de contrôle (en unités ma).
- **Model.Width (long)** : largeur de l'élément de contrôle (en unités ma).
- **Model.PositionX (long)** : position X de l'élément de contrôle, mesurée à partir de l'arête intérieure gauche de la boîte de dialogue (en unités ma).
- **Model.PositionY (long)** : position Y mesurée à partir de l'arête intérieure supérieure de la boîte de dialogue (en unités ma).

Pour que les boîtes de dialogue conservent la même apparence d'une plate-forme à l'autre, StarOffice utilise l'unité interne *Map AppFont (ma)* qui permet de spécifier la position et la taille des éléments au sein des boîtes de dialogue. Une unité ma correspond au huitième de la hauteur moyenne d'un caractère de la police système définie dans le système d'exploitation et au quart de sa largeur. En utilisant les unités ma, StarOffice garantit qu'une boîte de dialogue aura la même apparence quels que soient le système et les paramètres du système.

Si vous souhaitez modifier la taille ou la position des éléments de contrôle au moment de l'exécution, déterminez la taille totale de la boîte de dialogue et réglez les valeurs des éléments de contrôle sur les rapports des parties correspondants.

Remarque – Map AppFont (ma) remplace l'unité Twips pour obtenir une meilleure indépendance vis-à-vis des plates-formes.

Focus et séquence de tabulation

Vous pouvez naviguer dans les éléments de contrôle des différentes boîtes de dialogue en appuyant sur la touche de tabulation. Les propriétés suivantes sont disponibles dans ce contexte dans le modèle des éléments de contrôle :

- **Model.Enabled (Boolean)** : active l'élément de contrôle.
- **Model.Tabstop (Boolean)** : permet d'atteindre l'élément de contrôle à l'aide de la touche de tabulation.

- **Model.TabIndex (Long)** : position de l'élément de contrôle dans l'ordre d'activation.

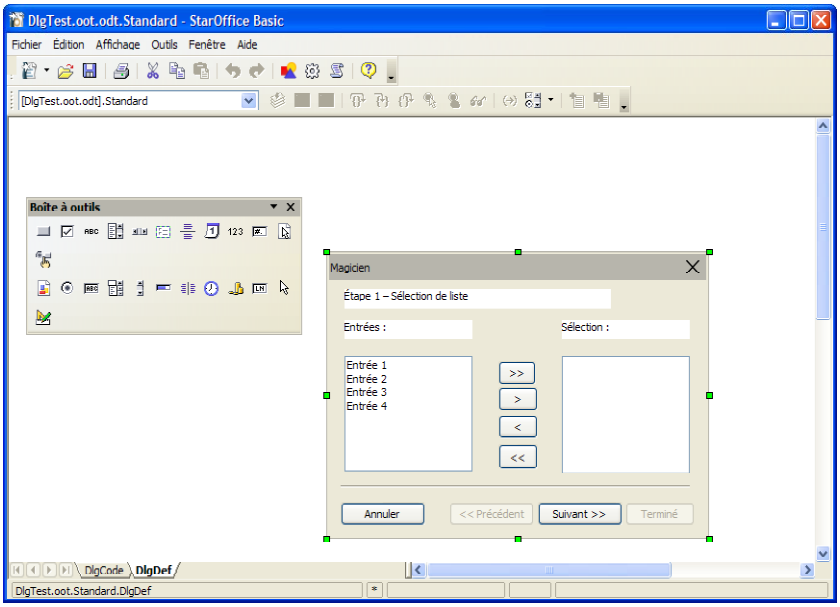
Enfin, l'élément de contrôle fournit une méthode `getFocus` permettant à l'élément de contrôle sous-jacent de recevoir le focus :

- **getFocus** : l'élément de contrôle reçoit le focus (uniquement pour les boîtes de dialogue).

Boîtes de dialogue à plusieurs onglets

Dans StarOffice, une boîte de dialogue peut posséder plusieurs onglets. La propriété `Step` d'une boîte de dialogue définit l'onglet actif, tandis que la propriété `Step` d'un élément de contrôle spécifie l'onglet dans laquelle il doit s'afficher.

La valeur `Step` (si elle définie sur 0) constitue un cas à part. Si vous définissez cette valeur sur zéro dans une boîte de dialogue, tous les éléments de contrôle sont visibles, quelle que soit leur valeur `Step`. De même, si vous définissez cette valeur sur zéro pour un élément de contrôle, celui-ci s'affiche sur tous les onglets d'une boîte de dialogue.



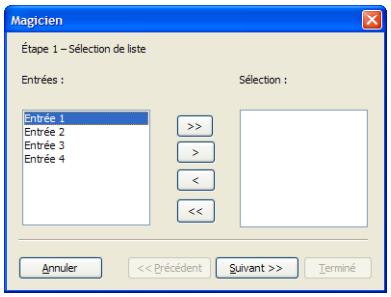
Dans l'exemple ci-dessus, vous pouvez également assigner la valeur `Step` (définie sur 0) à la ligne de séparation, ainsi que les boutons `Annuler`, `Précédent`, `Suivant` et `Terminé` pour afficher ces éléments sur tous les onglets. Vous pouvez également assigner ces éléments à un seul onglet (le premier, par exemple).

Le code suivant montre comment la valeur `Step` des gestionnaires d'événements des boutons `Suivant` et `Précédent` peut être augmentée ou réduite et modifie le statut de ces boutons.

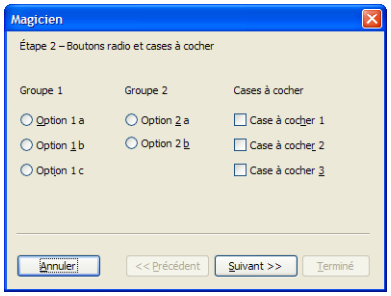
```
Sub cmdNext_Initiated
   Dim cmdNext As Object
   Dim cmdPrev As Object

   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
```

Une variable globale `Dlg` faisant référence à une boîte de dialogue ouverte doit être ajoutée pour que cet exemple fonctionne. La boîte de dialogue change ensuite d'apparence de la façon suivante :

Onglet 1 :



Onglet 2 :



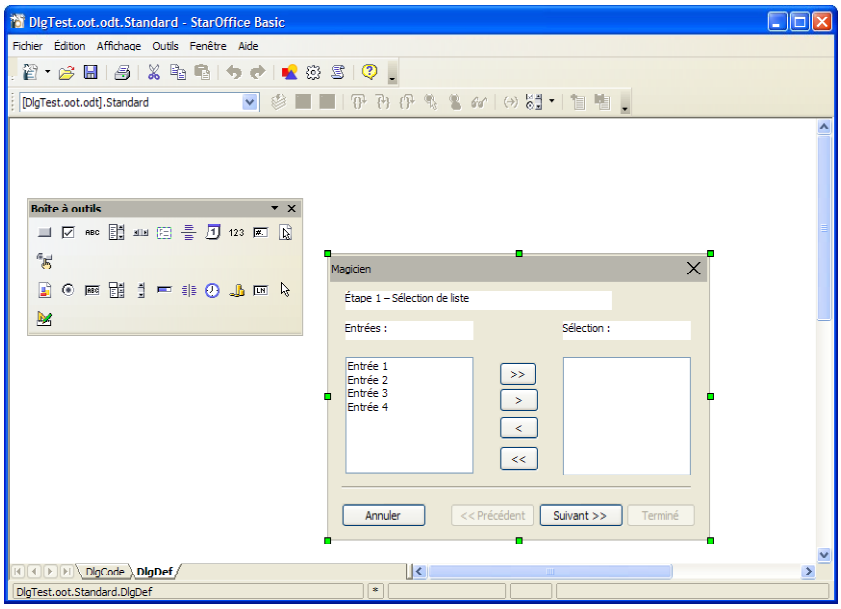
Événements

Les boîtes de dialogue et formulaires de StarOffice sont basés sur un modèle de programmation orienté événement dans lequel vous pouvez assigner des *gestionnaires d'événements* aux éléments de contrôle. Un gestionnaire d'événements exécute une procédure prédéfinie lorsqu'une action particulière se produit, même lorsque cette action est un autre événement. Grâce à la gestion d'événements, vous pouvez également éditer des documents ou des bases de données ouvertes, et accéder à d'autres éléments de contrôle.

Les éléments de contrôle de StarOffice reconnaissent divers types d'événements pouvant être déclenchés dans différentes situations. Ces types d'événements peuvent être répartis en quatre groupes :

- **Mouse control** : événements correspondant à des actions effectuées à l'aide de la souris (des mouvements simples ou un clic sur un emplacement particulier de l'écran, par exemple).
- **Keyboard control** : événements déclenchés par des séquences de touches.
- **Focus modification** : événements exécutés par StarOffice lorsque des éléments de contrôle sont activés ou désactivés.
- **Control element-specific events** : événements liés à certains éléments de contrôle.

Lorsque vous utilisez des événements, vérifiez que vous créez la boîte de dialogue associée dans l'environnement de développement de StarOffice et qu'elle contient les éléments de contrôle ou documents nécessaires (si les événements s'appliquent à un formulaire).



La figure ci-dessus illustre l'environnement de développement de StarOffice Basic avec une boîte de dialogue contenant deux zones de liste. Vous pouvez déplacer les données d'une liste à l'autre en utilisant les boutons entre les deux zones de liste.

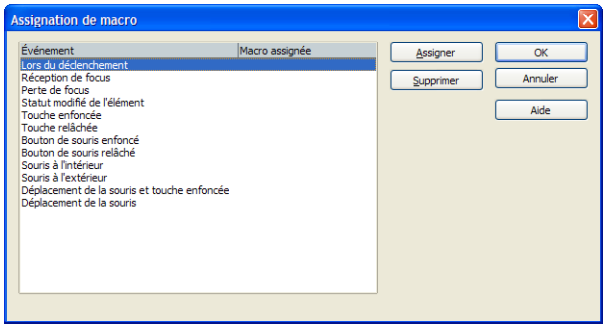
Pour afficher la mise en page à l'écran, vous devez créer les procédures StarOffice Basic associées afin qu'elles puissent être appelées par les gestionnaires d'événements. Bien que vous puissiez utiliser ces procédures dans tous les modules, il est préférable d'en limiter l'utilisation à deux modules. Pour simplifier la lecture de votre code, il est recommandé d'attribuer des noms significatifs à ces procédures. Le passage direct à une procédure générale du programme depuis une macro peut produire un code peu clair. Ainsi, afin de simplifier la maintenance et le dépannage des codes, créez plutôt une autre procédure servant de point d'entrée pour la gestion d'événements, même si elle n'exécute qu'un seul appel vers la procédure cible.

Le code de l'exemple suivant déplace une entrée de la zone de liste de gauche vers celle de droite dans une boîte de dialogue.

```
Sub cmdSelect_Initiated
   Dim objList As Object
   lstEntries = Dlg.getControl("lstEntries")
   lstSelection = Dlg.getControl("lstSelection")

   If lstEntries.SelectedItem > 0 Then
      lstSelection.AddItem(lstEntries.SelectedItem, 0)
      lstEntries.removeItems(lstEntries.SelectItemPos, 1)
   Else
      Beep
   End If
```

```
End Sub
```

Si cette procédure a été créée dans StarOffice Basic, vous pouvez l'assigner à un événement requis en utilisant la fenêtre des propriétés de l'éditeur de boîte de dialogue.



La boîte de dialogue d'assignation répertorie toutes les procédures de StarOffice Basic. Pour assigner une procédure à un événement, sélectionnez la procédure et cliquez sur **Assigner**.

Paramètres

L'occurrence d'un événement particulier ne suffit pas toujours pour obtenir une réponse appropriée. D'autres informations peuvent s'avérer nécessaires. Par exemple, pour traiter un clic de souris, vous aurez certainement besoin de connaître la position du curseur à l'écran lors de l'appui sur le bouton.

Dans StarOffice Basic, vous pouvez utiliser des paramètres d'objet pour fournir à une procédure des informations supplémentaires concernant un événement, par exemple :

```
Sub ProcessEvent(Event As Object)

End Sub
```

La précision avec laquelle l'objet et les propriétés `Event` sont structurés dépend du type d'événement déclenché par l'appel de procédure. Les sections suivantes décrivent les types d'événements de façon détaillée.

Quel que soit le type d'événement, tous les objets donnent accès à l'élément de contrôle correspondant et à son modèle. L'élément de contrôle peut être atteint en utilisant

```
Event.Source
```

et son modèle en utilisant

`Event.Source.Model`

Vous pouvez utiliser ces propriétés pour déclencher un événement au sein d'un gestionnaire d'événements.

Événements de la souris

StarOffice Basic reconnaît les événements de souris suivants :

- **Mouse moved** : l'utilisateur déplace la souris.
- **Mouse moved while key pressed** : l'utilisateur déplace la souris tout en maintenant une touche enfoncée.
- **Mouse button pressed** : l'utilisateur appuie sur un bouton de la souris.
- **Mouse button pressed** : l'utilisateur relâche un bouton de la souris.
- **Mouse outside** : l'utilisateur déplace la souris en dehors de la fenêtre active.

La structure des objets Event associés est définie dans la structure `com.sun.star.awt.MouseEvent` qui fournit les informations suivantes :

- **Buttons (short)** : le bouton est enfoncé (une ou plusieurs constantes correspondant à `com.sun.star.awt.MouseButton`).
- **X (long)** : coordonnée X de la souris, mesurée en pixels à partir de l'angle supérieur gauche de l'élément de contrôle.
- **Y (long)** : coordonnée Y de la souris, mesurée en pixels à partir de l'angle supérieur gauche de l'élément de contrôle.
- **ClickCount (long)** : nombre de clics associés à l'événement de la souris (si StarOffice peut répondre assez vite, `ClickCount` sera également égal 1 pour un double-clic, car un seul événement est déclenché).

Les constantes définies dans `com.sun.star.awt.MouseButton` pour les boutons de la souris sont :

- **LEFT** : bouton gauche de la souris.
- **RIGHT** : bouton droit de la souris.
- **MIDDLE** : bouton du milieu de la souris.

Le résultat de l'exemple suivant permet de déterminer la position de la souris, ainsi que le bouton de souris qui a été enfoncé :

```
Sub MouseUp(Event As Object)
   Dim Msg As String
   Msg = "Keys: "
   If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
      Msg = Msg & "LEFT "
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
      Msg = Msg & "RIGHT "
```

```
   End If
   If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
      Msg = Msg & "MIDDLE "
   End If
   Msg = Msg & Chr(13) & "Position: "
   Msg = Msg & Event.X & "/" & Event.Y
   MsgBox Msg
End Sub
```

Remarque – Les événements VBA `Click` et `Doubleclick` ne sont pas disponibles dans StarOffice Basic. Vous pouvez donc utiliser l'événement StarOffice Basic `MouseUp` à la place de l'événement `click` et imiter l'événement `Doubleclick` en modifiant la logique de l'application.

Événements du clavier

Les événements de clavier suivants sont disponibles dans StarOffice Basic :

- **Key pressed** : l'utilisateur appuie sur une touche.
- **Key released** : l'utilisateur relâche une touche.

Les deux événements sont associés à des actions de touches *logiques* et non à des actions *physiques*. Si l'utilisateur appuie sur plusieurs touches afin d'obtenir un seul caractère (par exemple, pour ajouter un accent à un caractère), StarOffice Basic ne crée qu'un événement.

Une action unique sur une touche de modification, comme la touche MAJ ou ALT, ne crée pas d'événement indépendant.

Les informations relatives à une touche enfoncée sont indiquées par l'objet Event que StarOffice Basic fournit à la procédure pour la gestion des événements. Il contient les propriétés suivantes :

- **KeyCode (short)** : code de la touche enfoncée (les valeurs par défaut correspondent à `com.sun.star.awt.Key`).
- **KeyChar (String)** : caractère saisi (en tenant compte des touches de modification).

L'exemple suivant utilise la propriété `KeyCode` pour établir si l'utilisateur a appuyé sur la touche Entrée, la touche de tabulation ou sur l'une des autres touches contrôle. Si l'une de ces touches a été enfoncée, son nom est retourné ; sinon, c'est le caractère saisi qui est retourné :

```
Sub KeyPressed(Event As Object)
   Dim Msg As String
   Select Case Event.KeyCode
   Case com.sun.star.awt.Key.RETURN
      Msg = "Return pressed"
   Case com.sun.star.awt.Key.TAB
```

```
      Msg = "Tab pressed"
   Case com.sun.star.awt.Key.DELETE
      Msg = "Delete pressed"
   Case com.sun.star.awt.Key.ESCAPE
      Msg = "Escape pressed"
   Case com.sun.star.awt.Key.DOWN
      Msg = "Down pressed"
   Case com.sun.star.awt.Key.UP
      Msg = "Up pressed"
   Case com.sun.star.awt.Key.LEFT
      Msg = "Left pressed"
   Case com.sun.star.awt.Key.RIGHT
      Msg = "Right pressed"
   Case Else
      Msg = "Character " & Event.KeyChar & " entered"
   End Select
   MsgBox Msg
End Sub
```

Vous trouverez de plus amples informations sur les autres constantes du clavier dans la référence de l'API, dans le groupe de constantes `com.sun.star.awt.Key`.

Événements du focus

Les événements du focus indiquent si un élément de contrôle reçoit ou perd le focus. Par exemple, vous pouvez utiliser ces événements pour déterminer si un utilisateur a terminé le traitement d'un élément de contrôle de telle sorte que vous puissiez mettre à jour d'autres éléments d'une boîte de dialogue. Les événements de focus suivants sont disponibles :

- **When receiving focus** : l'élément reçoit le focus.
- **When losing focus** : l'élément perd le focus.

Les objets `Event` des événements de focus sont structurés de la façon suivante :

- **FocusFlags (short)** : raison du changement de focus (la valeur par défaut correspond à `com.sun.star.awt.FocusChangeReason`).
- **NextFocus (Object)** : objet recevant le focus (seulement pour l'événement `When losing focus`).
- **Temporary (Boolean)** : le focus est temporairement perdu.

Événements propres à l'élément de contrôle

Outre les événements ci-dessus, qui sont pris en charge par tous les éléments de contrôle, il existe également certains événements spécifiques qui ne sont définis que pour certains éléments de contrôle. Les plus importants de ces événements sont :

- **When Item Changed** : la valeur d'un élément de contrôle change.

- **Item Status Changed** : le statut d'un élément de contrôle change.
- **Text modified** : le texte d'un élément de contrôle change.
- **When initiating** : action pouvant être exécutée lorsque l'élément de contrôle est déclenché (par exemple, lorsque l'utilisateur appuie sur un bouton).

Lorsque vous utilisez des événements, notez que certains d'entre eux, comme `When initiating`, peuvent être déclenchés chaque fois que vous cliquez avec la souris sur certains éléments de contrôle (par exemple, sur des boutons radio). Aucune action n'est exécutée pour vérifier si le statut de l'élément de contrôle a réellement été modifié. Pour éviter ce type d''événement aveugle', enregistrez l'ancienne valeur de l'élément de contrôle dans une variable globale et vérifiez si la valeur est modifiée au moment de l'exécution d'un événement.

Les propriétés de l'événement `Item Status Changed` sont :

- **Selected (long)** : entrée actuellement sélectionnée.
- **Highlighted (long)** : entrée actuellement mise en évidence.
- **ItemId (long)** : ID de l'entrée.

Détails des éléments de contrôle des boîtes de dialogue

StarOffice Basic reconnaît un ensemble d'éléments de contrôle qui peuvent être répartis dans les groupes suivants :

Champs de saisie :

- Champs de texte
- Champs de date
- Champs horaires
- Champs numériques
- Champs monétaires
- Champs adoptant tous les formats

Boutons :

- Boutons standard
- Cases à cocher
- Boutons radio

Listes de sélection :

- Zones de liste
- Boîtes combinées

Autres éléments de contrôle :

- Barres de défilement (horizontales et verticales)
- Champs de groupes
- Barres de progression
- Lignes de séparation (horizontales et verticales)
- Images
- Champs de sélection de fichier

Les éléments de contrôle les plus importants sont présentés ci-après.

Boutons

Un bouton exécute une action lorsque vous cliquez dessus.

Le scénario le plus simple est celui au cours duquel le bouton déclenche un événement `When Initiating` lorsqu'un utilisateur clique dessus. Vous pouvez également lier une autre action au bouton pour qu'une boîte de dialogue s'ouvre en utilisant la propriété `PushButtonType`. Lorsque vous cliquez sur un bouton pour lequel cette propriété est définie sur la valeur 0, rien ne se passe au niveau de la boîte de dialogue. Lorsque vous cliquez sur un bouton pour lequel cette propriété est définie sur la valeur 1, la boîte de dialogue est fermée et la méthode `Execute` de la boîte de dialogue retourne la valeur 1 (la séquence de la boîte de dialogue s'est terminée correctement). Si la propriété `PushButtonType` a la valeur 2, la boîte de dialogue est fermée et la méthode `Execute` de la boîte de dialogue retourne la valeur 0 (boîte de dialogue fermée).

Vous trouverez ci-après toutes les propriétés disponibles par l'intermédiaire du modèle de bouton :

- **Model.BackgroundColor (long)** : couleur d'arrière-plan.
- **Model.DefaultButton (Boolean)** : le bouton est utilisé comme valeur par défaut et répond à la touche Entrée en l'absence de focus.
- **Model.FontDescriptor (struct)** : structure spécifiant les détails de la police utilisée (correspondant à la structure `com.sun.star.awt.FontDescriptor`).
- **Model.Label (String)** : étiquette affichée sur le bouton.
- **Model.Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **Model.TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **Model.HelpText (String)** : texte d'aide s'affichant lorsque vous déplacez le curseur de la souris au-dessus de l'élément de contrôle.
- **Model.HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.
- **PushButtonType (short)** : action liée au bouton (0 : pas d'action, 1 : OK, 2 : Annuler).

Boutons radio

Ces boutons sont généralement utilisés en groupes et vous permettent de choisir parmi une ou plusieurs options. Lorsque vous sélectionnez une option, toutes les autres options du groupe sont désactivées. De cette façon, il n'est possible de définir q'un seul bouton radio à la fois.

Un élément de contrôle d'un bouton radio fournit deux propriétés :

- **State (Boolean)** : active le bouton.
- **Label (String)** : étiquette affichée sur le bouton.

Vous pouvez également utiliser les propriétés suivantes à partir du modèle des boutons radio :

- **Model.FontDescriptor (struct)** : structure contenant les détails de la police utilisée (correspondant à `com.sun.star.awt.FontDescriptor`).
- **Model.Label (String)** : étiquette affichée sur l'élément de contrôle.
- **Model.Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **Model.State (Short)** : si cette propriété est égale à 1, l'option est activée ; sinon, elle est désactivée.
- **Model.TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **Model.HelpText (String)** : texte d'aide s'affichant lorsque le curseur de la souris est placé au-dessus de l'élément de contrôle.
- **Model.HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Pour regrouper plusieurs boutons radio, vous devez les positionner les uns à la suite des autres dans la séquence d'activation sans aucun espace (propriété `Model.TabIndex` décrite en tant que Order dans l'éditeur de boîte de dialogue). Si la séquence d'activation est interrompue par un autre élément de contrôle, StarOffice démarre automatiquement avec un nouveau groupe d'éléments de contrôle pouvant être activé indépendamment du premier.

Remarque – Contrairement à VBA, vous ne pouvez pas insérer de boutons radio dans un groupe d'éléments de contrôle dans StarOffice Basic. Le groupement d'éléments de contrôle dans StarOffice Basic sert uniquement à créer une division visuelle en dessinant un cadre autour des éléments de contrôle.

Cases à cocher

Les cases à cocher servent à enregistrer une valeur Yes ou No et, suivant le mode, elles peuvent adopter deux ou trois états. En plus des états Yes et No, une case à cocher peut posséder un état intermédiaire si le statut correspondant pour Yes ou No a plusieurs significations ou n'est pas clair.

Les cases à cocher offrent les propriétés suivantes :

- **State (Short)** : état de la case à cocher (0 : non, 1 : oui, 2 : état intermédiaire).
- **Label (String)** : étiquette de l'élément de contrôle.
- **enableTriState (Boolean)** : outre les états actif et inactif, vous pouvez également utiliser l'état intermédiaire.

L'objet Model d'une case à cocher fournit les propriétés suivantes :

- **Model.FontDescriptor (struct)** : structure contenant les détails de la police utilisée (correspondant à la structure `com.sun.star.awt.FontDescriptor`).
- **Model.Label (String)** : étiquette de l'élément de contrôle.
- **Model.Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **Model.State (Short)** : état de la case à cocher (0 : non, 1 : oui, 2 : état intermédiaire).
- **Model.Tabstop (Boolean)** : permet d'atteindre l'élément de contrôle à l'aide de la touche de tabulation.
- **Model.TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **Model.HelpText (String)** : texte d'aide s'affichant lorsque vous placez le curseur de la souris au-dessus de l'élément de contrôle.
- **Model.HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Champs de texte

Les champs de texte permettent aux utilisateurs de saisir des chiffres et du texte. Le service `com.sun.star.awt.UnoControlEdit` constitue la base des champs de texte.

Un champ de texte peut contenir une ou plusieurs lignes et peut être édité ou bloqué par les utilisateurs. Les champs de texte peuvent également servir de champs numériques ou monétaires spéciaux, ainsi que de champs d'écran pour certaines tâches particulières. Ces éléments de contrôle étant basés sur le service Uno `UnoControlEdit`, leur gestion contrôlée par programme est semblable.

Les champs de texte offrent les propriétés suivantes :

- **Text (String)** : texte actuel.
- **SelectedText (String)** : texte actuellement mis en évidence.
- **Selection (Struct)** : mise en évidence en lecture seule des détails (structure correspondant à `com.sun.star.awt.Selection`, avec les propriétés `Min` et `Max` spécifiant le début et la fin de la mise en évidence).
- **MaxTextLen (short)** : nombre maximal de caractères autorisés dans le champ.
- **Editable (Boolean)** : `True` active l'option de saisie de texte et `False` bloque cette option de saisie (la propriété ne peut pas être appelée directement : vous devez utiliser `IsEditable`).

- **IsEditable (Boolean)** : le contenu de l'élément de contrôle peut être modifié, en lecture seule.

De plus, les propriétés suivantes sont fournies par l'intermédiaire de l'objet Model associé :

- **Model.Align (short)** : orientation du texte (0 : aligné à gauche, 1 : centré, 2 : aligné à droite).
- **Model.BackgroundColor (long)** : couleur d'arrière-plan de l'élément de contrôle.
- **Model.Border (short)** : type de bordure (0 : pas de bordure, 1 : bordure 3D, 2 : bordure simple).
- **Model.EchoChar (String)** : caractère écho pour le champ de mot de passe.
- **Model.FontDescriptor (struct)** : structure contenant les détails de la police utilisée (correspondant à la structure `com.sun.star.awt.FontDescriptor`).
- **Model.HardLineBreaks (Boolean)** : des retours à la ligne automatiques sont constamment insérés dans le texte de l'élément de contrôle.
- **Model.HScroll (Boolean)** : le texte possède une barre de défilement horizontale.
- **Model.MaxTextLen (Short)** : longueur maximale du texte. Si la valeur 0 est indiquée, il n'existe aucune limite.
- **Model.MultiLine (Boolean)** : l'entrée peut s'étendre sur plusieurs lignes.
- **Model.Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **Model.ReadOnly (Boolean)** : le contenu de l'élément de contrôle est en lecture seule.
- **Model.Tabstop (Boolean)** : permet d'atteindre l'élément de contrôle à l'aide de la touche de tabulation.
- **Model.Text (String)** : texte associé à l'élément de contrôle.
- **Model.TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **Model.VScroll (Boolean)** : le texte possède une barre de défilement verticale.
- **Model.HelpText (String)** : texte d'aide s'affichant lorsque le curseur de la souris est placé au-dessus de l'élément de contrôle.
- **Model.HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Zones de liste

Les zones de listes (service `com.sun.star.awt.UnoControlListBox`) prennent en charge les propriétés suivantes :

- **ItemCount (Short)** : nombre d'éléments, en lecture seule.
- **SelectedItem (String)** : texte de l'entrée mise en évidence, en lecture seule.
- **SelectedItems (Array Of Strings)** : champ de données contenant les entrées mises en évidence, en lecture seule.

- **SelectItemPos (Short)** : numéro de l'entrée actuellement mise en évidence, en lecture seule.
- **SelectItemsPos (Array of Short)** : champ de données contenant le nombre d'entrées mises en évidence (pour les listes acceptant les sélections multiples), en lecture seule.
- **MultipleMode (Boolean)** : `True` active l'option de sélection multiple des entrées et `False` bloque les sélections multiples (la propriété ne peut pas être appelée directement : vous devez utiliser `IsMultipleMode`).
- **IsMultipleMode (Boolean)** : permet la sélection multiple dans les listes, en lecture seule.

Les zones de liste offrent les méthodes suivantes :

- **addItem (Item, Pos)** : insère la chaîne spécifiée dans `Item` dans la liste, à la position `Pos`.
- **addItem (ItemArray, Pos)** : insère les entrées spécifiées dans `ItemArray` dans le champ de données, à la position `Pos`.
- **removeItems (Pos, Count)** : supprime les entrées `Count` à partir de la position `Pos`.
- **selectItem (Item, SelectMode)** : active ou désactive la mise en évidence de l'élément spécifié dans la chaîne `Item` en fonction de la variable booléenne `SelectMode`.
- **makeVisible (Pos)** : fait défiler les champs de la liste afin de rendre l'entrée spécifiée par `Pos` visible.

L'objet Model des zones de liste fournit les propriétés suivantes :

- **Model.BackgroundColor (Long)** : couleur d'arrière-plan de l'élément de contrôle.
- **Model.Border (short)** : type de bordure (0 : pas de bordure, 1 : bordure 3D, 2 : bordure simple).
- **Model.FontDescriptor (struct)** : structure contenant les détails de la police utilisée (correspondant à la structure `com.sun.star.awt.FontDescriptor`).
- **Model.LineCount (Short)** : nombre de lignes dans l'élément de contrôle.
- **Model.MultiSelection (Boolean)** : permet la sélection multiple d'entrées.
- **Model.SelectedItems (Array of Strings)** : liste des entrées mises en évidence.
- **Model.StringItemList (Array of Strings)** : liste de toutes les entrées.
- **Model.Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **Model.ReadOnly (Boolean)** : le contenu de l'élément de contrôle est en lecture seule.
- **Model.Tabstop (Boolean)** : permet d'atteindre l'élément de contrôle à l'aide de la touche de tabulation.
- **Model.TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **Model.HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.

- **Model.HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Remarque – L'option VBA permettant d'émettre des entrées de liste avec une valeur numérique supplémentaire (`ItemData`) n'existe pas dans StarOffice Basic. Si vous souhaitez gérer une valeur numérique (ID de base de données, par exemple) en plus du texte en langage naturel, vous devez créer un champ de données auxiliaire en plus de la zone de liste.

CHAPITRE **12**

Formulaires

À de nombreux égards, la structure des formulaires StarOffice correspond aux boîtes de dialogue abordées au chapitre précédent. Il existe cependant quelques différences importantes :

- Les boîtes de dialogue apparaissent sous la forme d'une seule fenêtre de dialogue, qui s'affiche sur le document et qui ne permet aucune autre action que le traitement de la boîte de dialogue jusqu'à sa fermeture. Les formulaires, en revanche, s'affichent directement dans le document, tout comme les éléments de dessin.
- L'éditeur de boîte de dialogue situé dans l'environnement de développement StarOffice Basic permet de créer des boîtes de dialogue. Les formulaires sont créés directement dans le document à l'aide de la **barre d'outils Fonctions de formulaire**.
- Alors que les fonctions de boîte de dialogue sont disponibles dans tous les documents StarOffice, l'ensemble complet des fonctions de formulaire n'est disponible que dans les textes et les feuilles de calcul.
- Les éléments de contrôle d'un formulaire peuvent être liés à une table de base de données externe. Cette fonction n'est pas disponible dans les boîtes de dialogue.
- Les éléments de contrôle des boîtes de dialogue et des formulaires diffèrent en plusieurs points.

Les utilisateurs souhaitant créer des formulaires avec leurs propres méthodes de gestion des événements doivent se reporter au chapitre 11, Boîtes de dialogue. Les mécanismes expliqués ici sont identiques à ceux des formulaires.

Utilisation des formulaires

Les formulaires StarOffice peuvent contenir des champs de texte, des zones de liste, des boutons radio, ainsi que plusieurs autres éléments de contrôle, qui sont insérés directement dans un texte ou une feuille de calcul. La **barre d'outils Fonctions de formulaire** permet d'éditer des formulaires.

Un formulaire StarOffice peut adopter l'un des deux modes suivants : le mode brouillon et le mode affichage. En mode brouillon, la position des éléments de contrôle peut être modifiée et leurs propriétés peuvent être éditées dans la fenêtre des propriétés.

La **barre d'outils Fonctions de formulaire** permet également de passer d'un mode à l'autre.

Détermination de formulaires d'objet

StarOffice positionne les éléments de contrôle d'un formulaire au niveau de l'objet de dessin. Le formulaire d'objet réel est accessible par l'intermédiaire de la liste des formulaires au niveau du dessin. L'accès aux objets s'effectue de la façon suivante dans les documents texte :

```
Dim Doc As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
DrawPage = Doc.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

La méthode `GetByIndex` retourne le formulaire avec le numéro d'index 0.

Lorsque vous utilisez des feuilles de calcul, une étape intermédiaire est nécessaire via la liste des feuilles, car les niveaux de dessin ne se trouvent pas directement dans le document mais dans les feuilles individuelles :

```
Dim Doc As Object
Dim Sheet As Object
Dim DrawPage As Object
Dim Form As Object

Doc = StarDesktop.CurrentComponent
Sheet = Doc.Sheets.GetByIndex(0)
DrawPage = Sheet.DrawPage
Form = DrawPage.Forms.GetByIndex(0)
```

Comme le suggère le nom de la méthode `GetByIndex`, un document peut contenir plusieurs formulaires. Cela est utile si, par exemple, le contenu de différentes bases de données s'affiche dans un document ou si une relation de la base de données 1:n s'affiche dans un formulaire. L'option permettant de créer des sous-formulaires peut également être utilisée à cet effet.

Les trois aspects d'un formulaire d'éléments de contrôle

Un élément de contrôle d'un formulaire a trois aspects :

- En premier lieu, il y a le *modèle* de l'élément de contrôle. Il s'agit pour le programmeur StarOffice Basic du principal objet lors de l'utilisation des formulaires d'éléments de contrôle.
- Son équivalent est la *vue* de l'élément de contrôle, qui permet de gérer les informations d'affichage.
- Dans la mesure où les formulaires d'éléments de contrôle des documents sont gérés comme des éléments spéciaux du dessin, il existe également un objet *Shape* qui représente les propriétés spécifiques de l'élément de dessin figurant dans l'élément de contrôle (en particulier sa position et sa taille).

Accès au modèle des formulaires d'éléments de contrôle

Les modèles des éléments de contrôle d'un formulaire sont accessibles via la méthode `GetByName` du formulaire d'objet :

```
Dim Doc As Object
Dim Form As Object
Dim Ctl As Object

Doc = StarDesktop.CurrentComponent
Form = Doc.DrawPage.Forms.GetByIndex(0)
Ctl = Form.getByName("MyListBox")
```

Cet exemple détermine le modèle de l'élément de contrôle `MyListBox`, situé dans le premier formulaire du document texte actuellement ouvert.

En cas de doute sur le formulaire d'un élément de contrôle, vous pouvez utiliser l'option de recherche dans tous les formulaires de l'élément de contrôle souhaité :

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
```

```
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      Exit Function
   End If
Next I
```

Cet exemple utilise la méthode `HasByName` pour vérifier tous les formulaires d'un document texte, afin de déterminer s'ils contiennent un modèle d'élément de contrôle nommé `MyListBox`. Si un modèle correspondant est trouvé, une référence est enregistrée dans la variable `Ctl` et la recherche est interrompue.

Accès à la vue des formulaires d'éléments de contrôle

Pour accéder à la vue d'un formulaire d'éléments de contrôle, vous devez d'abord indiquer le modèle associé. La vue du formulaire d'éléments de contrôle peut être déterminée grâce au modèle et au contrôleur de document.

```
Dim Doc As Object
Dim DocCrl As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim CtlView As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
DocCrl = Doc.getCurrentControler()
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyListBox") Then
      Ctl = Form.GetbyName("MyListBox")
      CtlView = DocCrl.GetControl(Ctl)
      Exit Function
   End If
Next I
```

Le code de cet exemple est très semblable à celui de l'exemple précédent permettant de déterminer un modèle d'élément de contrôle. Il utilise non seulement l'objet de document `Doc`, mais aussi l'objet du contrôleur de document `DocCrl` qui fait référence à la fenêtre du document actif. À l'aide de l'objet du contrôleur et du modèle de l'élément de contrôle, il utilise ensuite la méthode `GetControl` pour déterminer la vue (variable `CtlView`) du formulaire d'éléments de contrôle.

Accès à l'objet Shape des formulaires d'éléments de contrôle

La méthode permettant d'accéder aux objets Shape d'un élément de contrôle utilise également le niveau de dessin correspondant du document. Pour déterminer un élément de contrôle particulier, vous devez effectuer une recherche dans tous les éléments de dessin du niveau de dessin.

```
Dim Doc As Object
Dim Shape as Object
Dim I as integer

Doc = StarDesktop.CurrentComponent

For i = 0 to Doc.DrawPage.Count - 1
   Shape = Doc.DrawPage(i)

   If HasUnoInterfaces(Shape, _
      "com.sun.star.drawing.XControlShape") Then
      If Shape.Control.Name = "MyListBox" Then
            Exit Function
      End If
   End If
Next
```

Cet exemple vérifie tous les éléments de dessin pour déterminer s'ils prennent en charge l'interface `com.sun.star.drawing.XControlShape` requise pour les formulaires d'éléments de contrôle. Si c'est le cas, la propriété `Control.Name` vérifie ensuite si le nom de l'élément de contrôle est `MyListBox`. Si la valeur est True, la fonction met fin à la recherche.

Détermination de la taille et de la position des éléments de contrôle

Comme indiqué précédemment, la taille et la position des éléments de contrôle peuvent être déterminées à l'aide de l'objet `Shape` associé. À cet effet, la forme de l'élément de contrôle, comme tous les autres objets `Shape`, fournit les propriétés `Size` et `Position` :

- **Size (struct)** : taille de l'élément de contrôle (structure de données `com.sun.star.awt.Size`).
- **Position (struct)** : position de l'élément de contrôle (structure de données `com.sun.star.awt.Point`).

L'exemple suivant montre comment définir la position et la taille d'un élément de contrôle en utilisant l'objet Shape associé :

```
Dim Shape As Object

Point.x = 1000
```

```
Point.y = 1000
Size.Width = 10000
Size.Height = 10000

Shape.Size = Size
Shape.Position = Point
```

Pour que le code fonctionne, l'objet `Shape` de l'élément de contrôle doit déjà être connu. Si ce n'est pas le cas, il doit être déterminé en utilisant le code ci-dessus.

Détails des formulaires d'éléments de contrôle

Les éléments de contrôle disponibles dans les formulaires sont semblables à ceux des boîtes de dialogue. Vous pouvez choisir parmi plusieurs champs de texte simples, zones de liste, boîtes combinées et boutons.

Vous trouverez ci-dessous la liste des propriétés les plus importantes pour les formulaires des éléments de contrôle. Toutes les propriétés font partie des objets Model associés.

Outre les éléments de contrôle standard, un élément de contrôle de table est également disponible pour les formulaires : il permet l'intégration de tables de bases de données complètes. Pour plus d'informations à ce sujet, reportez-vous à la section "Formulaires de base de données" à la page 225 du Chapitre 12.

Boutons

L'objet Model d'un bouton de formulaire fournit les propriétés suivantes :

- **BackgroundColor (long)** : couleur d'arrière-plan.
- **DefaultButton (Boolean)** : le bouton sert de valeur par défaut. Dans ce cas, il répond également au bouton d'entrée s'il n'est pas activé.
- **Enabled (Boolean)** : l'élément de contrôle peut être activé.
- **Tabstop (Boolean)** : l'élément de contrôle peut être atteint avec le bouton de tabulation.
- **TabIndex (Long)** : position de l'élément de contrôle dans la séquence d'activation.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **Tag (String)** : chaîne contenant des informations supplémentaires, pouvant être enregistrées dans le bouton pour un accès contrôlé par le programme.

- **TargetURL (String)** : URL cible pour les boutons du type d'URL.
- **TargetFrame (String)** : nom de la fenêtre (ou du cadre) dans laquelle l'URL cible (`TargetURL`) doit être ouvert lors de l'activation du bouton (pour les boutons du type d'`URL`).
- **Label (String)** : nom du bouton.
- **TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.
- **HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.
- **ButtonType (Enum)** : action liée au bouton (valeur par défaut à partir de `com.sun.star.form.FormButtonType`).

La propriété `ButtonType` permet de définir une action s'exécutant automatiquement lorsque l'utilisateur appuie sur le bouton. Le groupe de constantes `com.sun.star.form.FormButtonType` associé fournit les valeurs suivantes :

- **PUSH** : bouton standard.
- **SUBMIT** : fin de l'entrée du formulaire (particulièrement utile pour les formulaires HTML).
- **RESET** : rétablit toutes les valeurs d'origine du formulaire.
- **URL** : appelle l'URL défini dans `TargetURL`. L'URL est ouvert dans la fenêtre spécifiée via `TargetFrame`.

Les types de bouton **OK** et **Annuler** figurant dans les boîtes de dialogue ne sont pas pris en charge dans les formulaires.

Boutons radio

Les propriétés suivantes d'un bouton radio sont disponibles via l'objet Model :

- **Enabled (Boolean)** : l'élément de contrôle peut être activé.
- **Tabstop (Boolean)** : l'élément de contrôle peut être atteint avec le bouton de tabulation.
- **TabIndex (Long)** : position de l'élément de contrôle dans la séquence d'activation.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **Tag (String)** : chaîne contenant des informations supplémentaires, pouvant être enregistrées dans le bouton pour un accès contrôlé par le programme.
- **Label (String)** : nom du bouton.
- **Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **State (Short)** : si la valeur est 1, l'option est activée ; sinon, elle est désactivée.

- **RefValue (String)** : chaîne permettant d'enregistrer des informations supplémentaires (pour gérer les ID des enregistrements de données, par exemple).
- **TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.
- **HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Le mécanisme de regroupement des boutons radio fait la distinction entre les éléments de contrôle destinés aux boîtes de dialogue et aux formulaires. Alors que les éléments de contrôle apparaissant successivement dans les boîtes de dialogue sont automatiquement regroupés, le regroupement des formulaires s'effectue sur la base des noms. Pour ce faire, tous les boutons radio d'un groupe doivent porter le même nom. StarOffice associe les éléments de contrôle groupés dans une matrice afin que les boutons individuels d'un programme StarOffice Basic puissent être atteints de la même manière que précédemment.

L'exemple suivant montre comment déterminer le modèle d'un groupe d'éléments de contrôle.

```
Dim Doc As Object
Dim Forms As Object
Dim Form As Object
Dim Ctl As Object
Dim I as Integer

Doc = StarDesktop.CurrentComponent
Forms = Doc.Drawpage.Forms

For I = 0 To Forms.Count - 1
   Form = Forms.GetbyIndex(I)
   If Form.HasByName("MyOptions") Then
      Ctl = Form. GetGroupbyName("MyOptions")
      Exit Function
   End If
Next I
```

Ce code correspond à l'exemple précédent permettant de déterminer un modèle simple d'élément de contrôle. Il effectue une recherche en boucle dans tous les formulaires du document texte actif et utilise la méthode `HasByName` pour vérifier si le formulaire correspondant contient un élément comportant le nom `MyOptions` recherché. Si tel est le cas, l'accès à la matrice du modèle s'effectue via la méthode `GetGroupByName`, plutôt que la méthode `GetByName` qui permet de déterminer des modèles simples.

Cases à cocher

L'objet Model d'un formulaire de case à cocher fournit les propriétés suivantes :

- **Enabled (Boolean)** : l'élément de contrôle peut être activé.

- **Tabstop (Boolean)** : l'élément de contrôle peut être atteint avec la touche de tabulation.
- **TabIndex (Long)** : position de l'élément de contrôle dans la séquence d'activation.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **Tag (String)** : chaîne contenant des informations supplémentaires, pouvant être enregistrées dans le bouton pour un accès contrôlé par le programme.
- **Label (String)** : nom du bouton.
- **Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **State (Short)** : si la valeur est 1, l'option est activée ; sinon, elle est désactivée.
- **RefValue (String)** : chaîne permettant d'enregistrer des informations supplémentaires (pour gérer les ID des enregistrements de données, par exemple).
- **TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.
- **HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Champs de texte

L'objet Model d'un formulaire de champ de texte fournit les propriétés suivantes :

- **Align (short)** : orientation du texte (0 : aligné à gauche, 1 : centré, 2 : aligné à droite).
- **BackgroundColor (long)** : couleur d'arrière-plan de l'élément de contrôle.
- **Border (short)** : type de bordure (0 : aucune bordure, 1 : bordure 3D, 2 : bordure simple).
- **EchoChar (String)** : caractère écho pour le champ de mot de passe.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **HardLineBreaks (Boolean)** : des retours à la ligne automatiques sont constamment insérés dans le texte de l'élément de contrôle.
- **HScroll (Boolean)** : le texte possède une barre de défilement horizontale.
- **MaxTextLen (Short)** : longueur maximale du texte. Si la valeur 0 est indiquée, il n'existe aucune limite.
- **MultiLine (Boolean)** : autorise les entrées sur plusieurs lignes.
- **Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **ReadOnly (Boolean)** : le contenu de l'élément de contrôle est en lecture seule.
- **Enabled (Boolean)** : l'élément de contrôle peut être activé.

- **Tabstop (Boolean)** : l'élément de contrôle peut être atteint avec la touche de tabulation.
- **TabIndex (Long)** : position de l'élément de contrôle dans la séquence d'activation.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **Text (String)** : texte de l'élément de contrôle.
- **TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **VScroll (Boolean)** : le texte possède une barre de défilement verticale.
- **HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.
- **HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Zones de liste

L'objet Model d'un formulaire de zone de liste fournit les propriétés suivantes :

- **BackgroundColor (long)** : couleur d'arrière-plan de l'élément de contrôle.
- **Border (short)** : type de bordure (0 : aucune bordure, 1 : bordure 3D, 2 : bordure simple).
- **FontDescriptor (struct)** : structure correspondant à `com.sun.star.awt.FontDescriptor`, indiquant les polices à utiliser.
- **LineCount (Short)** : nombre de lignes dans l'élément de contrôle.
- **MultiSelection (Boolean)** : autorise la sélection multiple d'entrées.
- **SelectedItems (Array of Strings)** : liste des entrées mises en évidence.
- **StringItemList (Array of Strings)** : liste de toutes les entrées.
- **ValueItemList (Array of Variant)** : liste contenant des informations supplémentaires pour chaque entrée (pour gérer les ID des enregistrements de données, par exemple).
- **Printable (Boolean)** : l'élément de contrôle peut être imprimé.
- **ReadOnly (Boolean)** : le contenu de l'élément de contrôle est en lecture seule.
- **Enabled (Boolean)** : l'élément de contrôle peut être activé.
- **Tabstop (Boolean)** : l'élément de contrôle peut être atteint avec la touche de tabulation.
- **TabIndex (Long)** : position de l'élément de contrôle dans la séquence d'activation.
- **FontName (String)** : nom du type de police.
- **FontHeight (Single)** : hauteur de caractère en points (pt).
- **Tag (String)** : chaîne contenant des informations supplémentaires, pouvant être enregistrées dans le bouton pour un accès contrôlé par le programme.

- **TextColor (Long)** : couleur du texte de l'élément de contrôle.
- **HelpText (String)** : texte d'aide s'affichant automatiquement lorsque le curseur de la souris se trouve au-dessus de l'élément de contrôle.
- **HelpURL (String)** : URL de l'aide en ligne pour l'élément de contrôle correspondant.

Remarque – La propriété `ValueItemList` des formulaires de zones de liste s'apparente à la propriété VBA `ItemData`, qui permet de gérer des informations supplémentaires pour les entrées de liste individuelles.

De plus, les méthodes suivantes sont fournies via l'objet View de la zone de liste :

- **addItem (Item, Pos)** : insère dans la liste la chaîne spécifiée dans `Item` à la position `Pos`.
- **addItems (ItemArray, Pos)** : insère dans la liste les entrées répertoriées dans le champ de données de la chaîne `ItemArray` à la position `Pos`.
- **removeItems (Pos, Count)** : supprime les entrées `Count` positionnées sur `Pos`.
- **selectItem (Item, SelectMode)** : active ou désactive la mise en évidence de l'élément spécifié dans la chaîne `Item`, en fonction de la variable `SelectMode`.
- **makeVisible (Pos)** : fait défiler les champs de la liste afin de rendre l'entrée spécifiée par `Pos` visible.

Formulaires de base de données

Les formulaires StarOffice peuvent être directement liés à une base de données. Les formulaires créés de cette manière offrent toutes les fonctions d'une base de données frontale complète et ne nécessitent pas de programmation indépendante.

L'utilisateur peut parcourir les tables et requêtes sélectionnées, y effectuer des recherches, modifier des enregistrements de données et en insérer de nouveaux. StarOffice vérifie automatiquement la pertinence des données récupérées à partir de la base de données et s'assure que toutes les modifications sont réécrites dans la base de données.

Un formulaire de base de données correspond en fait à un formulaire StarOffice standard. Outre les propriétés standard, les propriétés suivantes, propres aux bases de données, doivent également être définies dans le formulaire :

- **DataSourceName (String)** : nom de la source de données ; la source de données doit être créée de manière globale dans StarOffice. Reportez-vous au Chapitre 10
- **Command (String)** : nom de la table, requête ou commande Select SQL vers laquelle un lien doit être établi.

- **CommandType (Const)** : spécifie si la commande est une table, une requête ou une commande SQL (valeur obtenue à partir de l'énumération `com.sun.star.sdb.CommandType`).

L'énumération `com.sun.star.sdb.CommandType` reconnaît les valeurs suivantes :

- **TABLE** : table
- **QUERY** : requête
- **COMMAND** : commande SQL

Les champs de la base de données sont assignés aux éléments de contrôle individuels via cette propriété :

- **DataField (String)** : nom du champ de la base de données liée.

Tables

Un autre élément de contrôle est fourni pour utiliser les bases de données : il s'agit de l'élément de contrôle de table. Il représente le contenu d'une table ou d'une requête de base de données complète. Dans le scénario le plus simple, un élément de contrôle de table est lié à une base de données grâce au formulaire de l'assistant qui relie toutes les colonnes aux champs correspondants de la base de données, conformément aux spécifications de l'utilisateur. Du fait de la relative complexité de l'API, nous n'en fournirons pas une description complète pour l'instant.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U