StarOffice™ 7 Office Suite

A Sun™ ONE Software Offering

Manuel de programmation Basic

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

Part No. 817-3917-10
2003, Revision A

Copyrights and Trademarks

Table des matières

CHAPITRE **1**

Introduction

Ce manuel est une introduction à la programmation à l'aide de StarOffice Basic 7.0 et indique les applications possibles liées à l'utilisation de StarOffice Basic dans StarOffice. Pour tirer le meilleur parti de cet ouvrage, vous devez être familiarisé avec d'autres langages de programmation.

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

De nombreuses astuces de migration destinées aux programmeurs Microsoft Visual Basic ou à ceux ayant travaillé avec des versions antérieures de StarOffice Basic sont proposées tout au long de ce manuel. Celles-ci sont signalées par un petit symbole figurant sur le bord de la page. L'annexe de ce manuel contient un index de toutes les astuces de migration vous permettant d'accéder rapidement à celle que vous souhaitez consulter.

À propos de StarOffice Basic

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

Comme son nom l'indique, StarOffice Basic est un langage de programmation appartenant à la famille Basic. Quiconque ayant déjà travaillé avec d'autres langages Basic – en particulier avec Visual Basic ou Visual Basic pour applications (VBA) de Microsoft – se familiarisera 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** : définit les structures linguistiques élémentaires, par exemple, pour les déclarations de variables, les boucles et les fonctions.
- **La bibliothèque d'exécution** : fournit des fonctions standard qui n'ont pas de référence directe avec StarOffice, par exemple, des fonctions d'édition de nombres, de chaînes de caractères, de dates et de fichiers.
- **L'API (Application Programming Interface) StarOffice** : 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** : crée des fenêtres de boîtes de dialogue personnelles et permet l'ajout d'éléments de contrôle et de gestionnaires d'événements.

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

Utilisateurs cibles de StarOffice Basic

Le champ d'application de StarOffice Basic commence là où s'arrêtent les fonctions standard de StarOffice. Les tâches récurrentes peuvent ainsi être automatisées dans StarOffice Basic, des liens peuvent être établis vers d'autres programmes (vers un serveur de base de données, par exemple) et les activités complexes peuvent être exécutées par un clic sur un bouton grâce à des scripts prédéfinis.

StarOffice Basic offre un accès complet à toutes les fonctions StarOffice, supporte toutes les fonctions, modifie les types de document et fournit des options pour la création de fenêtres de boîtes de dialogue personnelles.

Utilisation de StarOffice Basic

Tous les utilisateurs de StarOffice peuvent utiliser StarOffice Basic sans aucun autre programme ou aide.
Même avec l'installation standard, StarOffice Basic comprend tous les composants nécessaires à la création de ses propres macros Basic, dont :

- **L'environnement de développement intégré** (IDE, Integrated Development Environment) qui met à disposition un éditeur pour la saisie et le test des macros.
- **L'interpréteur**, nécessaire à l'exécution des macros de StarOffice Basic.
- **Les interfaces** vers diverses applications StarOffice, permettant d'accéder directement aux documents Office.

Structure de ce manuel

Les trois premiers chapitres présentent StarOffice Basic aux lecteurs :

- Chapitre 2 : Le langage de StarOffice Basic
- Chapitre 3 : La 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

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/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 surtout prévu pour les programmeurs Java et C++. Les personnes déjà familiarisées avec la programmation StarOffice Basic peuvent trouver des informations complémentaires dans le Developer's Guide sur StarOffice Basic et la programmation dans StarOffice. Vous pouvez télécharger le Developer's Guide depuis Internet à l'adresse :

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

Les programmeurs souhaitant travailler directement avec Java ou C++ plutôt qu'avec StarOffice Basic doivent se reporter au Developer's Guide de StarOffice plutôt qu'à ce manuel. La programmation de StarOffice avec Java ou C++ est une tâche beaucoup plus complexe que la programmation à l'aide de StarOffice Basic.

CHAPITRE 2

Le langage de StarOffice Basic

StarOffice Basic appartient à la famille des langages Basic. De nombreuses parties de StarOffice Basic sont identiques à Microsoft Visual Basic pour applications (VBA) et à Microsoft Visual Basic. Toute personne ayant déjà travaillé avec ces langages se familiarisera rapidement avec StarOffice Basic.

Les programmeurs d'autres langages – comme Java, C++ ou Delphi – ne devraient pas non plus avoir de problème à se familiariser avec StarOffice Basic. StarOffice Basic 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 plus en détail plus loin.

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

Présentation d'un 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 à la recherche d'erreurs patentes, puis exécuté ligne par ligne.

Lignes de programme

Le fait que l'interpréteur Basic fonctionne ligne par ligne est une des différences majeures entre le Basic et les autres langages de programmation. Alors que l'emplacement des retours à la ligne dans le code source des programmes, par exemple en 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)
```

. 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 aux programmeurs d'employer des deux-points (:) pour diviser une ligne en différentes sections afin de pouvoir y placer plusieurs expressions. Les assignations

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

peuvent, par exemple, être écrites sous la forme suivante :

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

Commentaires

En plus du code à exécuter, un programme StarOffice Basic peut également contenir des commentaires expliquant le rôle de ses différentes parties et fournissant des informations précieuses qui pourront servir lorsque le programme sera repris ultérieurement, par exemple pour être débogué.

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 est relativement long
            ' et s'étend sur plusieurs lignes. Le
            ' caractère de commentaire doit donc être répété
            ' à chaque ligne.
```

Marqueur

Un programme StarOffice Basic peut contenir des dizaines, des centaines, voire des milliers de ***marqueurs*** (noms donnés aux variables, constantes, fonctions, etc.)
Lorsque vous choisissez un nom pour un marqueur, vous devez respecter les règles suivantes :

- 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. Le marqueur `UneVariableDeTest` , par exemple, renvoie à la même variable que `uneVariabledetest` et `UNEVARIABLEDETEST.`

  Il y a 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.

Les règles de construction des marqueurs dans StarOffice Basic ne sont pas les mêmes que dans VBA. Contrairement à VBA, par exemple, StarOffice Basic n'autorise pas les caractères spéciaux dans les marqueurs, car ils peuvent provoquer des problèmes dans les projets internationaux.

Voici quelques exemples de marqueurs corrects et incorrects :

```
Surname             ' Correct
Surname5            ' Correct (le chiffre 5 n'est pas le premier caractère)
First Name          ' Incorrect (les espaces sont interdits)
DéjàVu                    ' Incorrect (les lettres comme é et à sont interdites)
5Surnames           ' Incorrect (le premier caractère ne peut pas être un chiffre)
First,Name          ' Incorrect (les virgules et les points sont interdits)
```

Utilisation des variables

Déclaration de variable implicite

Les langages Basic sont conçus pour être simples d'emploi. Par conséquent, 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 pratique et 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 de variable explicite

Afin d'éviter les erreurs dues à la déclaration implicite des variables, StarOffice Basic propose une option nommée :

```
Option Explicit
```

Celle-ci doit être indiquée dans la première ligne de code de chaque module et assure qu'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
```

Cet exemple déclare une variable portant le nom `MyVar` et de type variant. 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 logiques. Voici quelques exemples de variables de type Variant :

```
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 logique
```

Les variables déclarées 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 définie 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 du 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 enregistrer plusieurs déclarations de variable :

```
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

Les chaînes 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.

Depuis un jeu de caractères ASCII vers Unicode

Les jeux de caractères établissent une correspondance entre les différents caractères d'une chaîne et des codes (des chiffres et des caractères) dans une table, décrivant la façon dont l'ordinateur doit afficher la chaîne sur un écran ou une imprimante.

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 jeux de caractères ISO 8859 constituent le standard international longtemps attendu. 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. Cela a cependant pour résultat 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 supportée par de nombreux programmes, dont StarOffice et StarOffice Basic.

Variables chaînes

StarOffice Basic enregistre les chaînes sous forme de variables chaînes de caractères Unicode. Une variable 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 chaîne de caractères dépend de la longueur de celle-ci.

Exemple de déclaration d'une variable chaîne de caractères :

```
Dim Variable As String
```

Vous pouvez également écrire cette déclaration sous la forme :

```
Dim Variable$
```

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

Spécification de chaînes explicites

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

```
Dim MyString As String
MyString = " Ceci est un test"
```

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

```
Dim MyString As String
MyString =    "Cette chaîne est si longue qu'elle" + _
              "a été répartie sur deux lignes."
```

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

```
Dim MyString As String
MyString = "un ""-quotation mark."  ' produit un "-guillemet
```

Nombres

StarOffice Basic supporte cinq types de base pour traiter les nombres :

- Nombre entier
- Nombre entier long
- Nombre à virgule flottante
- Double
- Monnaie

Variables de type entier

Les variables de type entier (integer) peuvent stocker n'importe quel nombre entier compris entre -32 768 et 32 767. Un entier peut occuper deux octets de mémoire au maximum. Le symbole de déclaration de type pour une variable de type entier est 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 long, le nombre est arrondi par excès ou par défaut au nombre entier le plus proche.

Exemples de déclaration pour des variables entières :

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

Variables de type entier 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 quatre octets de mémoire au maximum. Le symbole de déclaration de type pour un entier long est 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, le nombre est arrondi par excès ou par défaut au nombre entier le plus proche.

Exemples de déclaration pour des variables de type entier long :

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

Variables de type simple

Les variables de type simple (single) peuvent stocker n'importe quel nombre à virgule flottante positif ou négatif, compris entre 3,402823 x 10^{38} et 1,401298 x 10^{-45}. Une variable de type simple peut occuper quatre octets de mémoire au maximum. Le symbole de déclaration de type pour une variable de type simple est !.

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 de type double

Les variables de type double peuvent stocker n'importe quel nombre à virgule flottante positif ou négatif et compris entre 1,79769313486232 x 10^{308} et 4,94065645841247 x 10^{-324}. Une variable de type double peut occuper huit octets de mémoire au maximum. Les variables doubles peuvent être utilisées pour des calculs précis. Le symbole de déclaration de type est le signe #.

Exemples de déclarations de variables doubles :

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

Variables monétaires

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 n'importe quelle valeur comprise entre –922 337 203 685 477,5808 et +922 337 203 685 477,5807 et peut occuper huit octets de mémoire au maximum. Le symbole de déclaration de type pour une variable monétaire est @.

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 entre) :

```
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 (.). Cette règle permet de garantir que 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 (+) ou moins (-) 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 (un espace entre le signe plus et le nombre de base)
A = - 1.43E2         ' Correct (un espace entre le signe moins 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 le premier et le 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. Cependant, le résultat attendu était $1,43 * 10^{-2}$ (soit 0,0143). Dans 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

Le système hexadécimal (base 16) présente l'avantage de pouvoir faire correspondre un octet à deux chiffres exactement. 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 vous suffit de les faire précéder par &H.

```
Dim A As Longer

A = &HFF      ' La valeur hexadécimale FF correspond à la valeur décimale 255
A = &H10      ' La valeur hexadécimale 10 correspond à la valeur décimale 16
```

Valeurs octales

StarOffice Basic comprend également le système octal (base 8), qui utilise les chiffres de 0 à 7 et dont les nombres sont précédés par &O.

```
Dim A As Longer

A = &O77      ' La valeur octale 77 correspond à la valeur décimale 63
A = &O10      ' La valeur octale 10 correspond à la valeur décimale 8
```

Vrai et faux

Variables logiques

Les variables logiques ne peuvent prendre qu'une des deux valeurs suivantes : `True (vrai)` et `False (faux)`. Elles sont bien adaptées aux spécifications binaires qui ne peuvent avoir que deux états. Une valeur logique 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 logiques. La déclaration ne peut se faire qu'en utilisant la mention supplémentaire ***As Boolean***.

Exemple de déclaration d'une variable logique :

```
Dim Variable As Boolean
```

Détails sur la date et l'heure

Variables de date

Les variables de date peuvent contenir des valeurs de date et d'heure. Lorsqu'il enregistre des 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
```

Champs de données

En plus des variables simples (***scalaires***), StarOffice Basic supporte également les champs de données (***matrices***). Un champ de données contient plusieurs variables adressées au moyen d'un indice.

Matrices simples

Une déclaration de matrice est similaire à celle d'une variable simple, mais le nom de la matrice est suivi de parenthèses contenant une indication du nombre de ses 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 déclarer des variables d'un type spécifique dans une matrice ; par exemple, la ligne suivante déclare une matrice contenant quatre variables entières :

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

Dans les exemples précédents, l'indice de la matrice commence toujours par la valeur initiale standard de zéro. Il est également possible de spécifier une plage de validité avec des valeurs initiale et finale pour 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'indice de 5 à 10:

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

Les valeurs d'indice ne doivent pas forcément être positives. L'exemple suivant montre une déclaration également 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'indice allant de -10 à -5.

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

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

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

Valeur spécifique pour l'indice de début

Généralement l'indice 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'indice commence ***toujours*** à 0. Pour plus de clarté, évitez d'utiliser Option Base 1.

Option Base 1 se contente de modifier la valeur d'indice 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)`.

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'indice de début est modifié. Alors que la déclaration `MyInteger(3)` crée trois valeurs entières dans VBA avec des valeurs d'indice de 1 à 3, la même déclaration dans StarOffice Basic crée quatre valeurs entières avec des valeurs d'indice de 1 à 4.

Champs de données multidimensionnels

En plus des champs de données unidimensionnels, StarOffice Basic supporte également les champs de données multidimensionnels. 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 valeurs (pouvant être adressées par les valeurs d'indice de 0 à 5). La matrice peut stocker un total de 6 × 6 = 36 valeurs entières.

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

Modifications dynamiques des dimensions des champs de données

Les exemples ci-dessus utilisent des champs de données ayant une dimension définie. Vous pouvez également 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 suivant :

```
ReDim MyArray(10)
```

Contrairement à VBA, où vous ne pouvez dimensionner que les matrices dynamiques avec Dim MyArray(), StarOffice Basic vous permet de modifier les matrices statiques comme les 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 pour 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'indice 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                          ' Définit les dimensions
                                                    ' initiales

' ...

ReDim Preserve MyArray(20)   As Integer             ' Augmente la taille du
                                                    ' champ de données, tout en
                                                    ' préservant le contenu
```

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

Contrairement à VBA, où seule la limite supérieure de la dernière dimension peut être modifiée lorsque vous utilisez `Preserve`, StarOffice Basic vous permet de modifier également les autres dimensions.

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

Portée et durée de vie des variables

Une variable dans StarOffice Basic est pourvue d'une durée de vie limitée ainsi que d'une portée limitée, qui détermine les autres parties de programme pouvant la lire et l'utiliser. 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, vous devez définir la variable comme `statique` :

```
Sub Test
      Static MyInteger As Integer

      ' ...

End Sub
```

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. Lorsque vous portez une application VBA vers StarOffice Basic, vous devez 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 est augmentée de un dans la fonction `Flip` et diminuée de un dans la fonction `Flop`. Ces deux modifications de la variable 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. Utilisez le mot-clé Private pour définir la variable :

```
Private MyInteger As Integer
```

Si plusieurs modules contiennent chacun une variable privée portant le même nom, StarOffice Basic crée une variable différente pour chaque occurrence du nom. Dans l'exemple suivant, les deux modules A et B possèdent une variable privée nommée C. La fonction Test commence par définir la variable privée dans le module A puis définit la variable privée 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 SetmoduleeA
      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
```

Constantes

Dans StarOffice Basic, utilisez le mot-clé `Const` pour déclarer une constante.

```
Const A = 10
```

Si vous le souhaitez, 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, alors que 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 vous permettent de lier des éléments selon les règles de l'algèbre de Boole. Si les opérateurs sont appliqués à des valeurs logiques, 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` Test d'équivalence (les deux parties valent `True` ou `False`)

`IMP` Implication (si la première expression est vraie, alors la seconde doit l'être également)

Opérateurs de comparaison

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

=	Égalité de nombres, de dates et de chaînes
<>	Inégalité de nombres, de dates et de chaînes
>	Supérieur à pour les nombres, les dates et les chaînes
>=	Supérieur ou égal pour les nombres, les dates et les chaînes
<	Inférieur à pour les nombres, les dates et les chaînes
<=	Inférieur ou égal pour les nombres, les dates et les chaînes

StarOffice Basic ne supporte pas l'opérateur de comparaison VBA`Like`.

Instructions conditionnelles

Utilisez des instructions conditionnelles pour 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 le montre 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 `A` est supérieur à 3, sinon `B` reçoit la valeur 0.

Pour les situations plus complexes, vous pouvez imbriquer plusieurs instructions `If` , par exemple :

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

Si la valeur de la variable `A` vaut zéro, `B` reçoit la valeur `0`. Si `A` est inférieur à `3` (mais différent de zéro), alors `B` reçoit la valeur `1`. Dans tous les autres cas (c'est-à-dire si `A` est supérieur ou égal à `3`), `B` reçoit la valeur `2`.

Select...Case

L'instruction `Select...Case` constitue une alternative à l'imbrication des instructions `If` et sert dans les cas où vous désirez employer une même variable pour plusieurs conditions :

```
Select Case DayOfWeek
Case 1:
      NameOfWeekday = "Dimanche"
Case 2:
      NameOfWeekday = "Lundi"
Case 3:
      NameOfWeekday = "Mardi"
Case 4:
      NameOfWeekday = "Mercredi"
Case 5:
      NameOfWeekday = "Jeudi"
Case 6:
      NameOfWeekday = "Vendredi"
Case 7:
      NameOfWeekday = "Samedi"
End Select
```

Dans cet exemple, chaque nom d'un jour de la semaine correspond à un nombre, de telle manière que la variable `DayOfWeek` reçoit la valeur 1 pour `Dimanche`, `2 pour Lundi`, et ainsi de suite.

La commande `Select` n'est pas limitée aux correspondances directes – vous pouvez également utiliser des opérateurs de comparaison ou des listes d'expressions dans un embranchement `Case`. L'exemple suivant dresse la liste des variantes les plus importantes de la syntaxe :

```
Select Case Var
Case 1 To 5

      ' ... Var est compris entre les nombres 1 et 5

Case 6, 7, 8

      ' ... Var vaut 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. Vous pouvez également avoir des boucles se répétant 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 chaque passage. Lorsque la variable I vaut 10, la boucle s'arrête.

Pour que le compteur de la boucle soit incrémenté d'une autre valeur que 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 ci-dessus, le compteur est incrémenté de 0,5 à la fin de chaque passage, et la boucle est exécutée 19 fois.

Vous pouvez utiliser également 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 d'une boucle `For` prématurément. 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
```

La variante `For Each...Next` de VBA n'est pas supportée par 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` (dans les exemples suivants, `A > 10` représente une condition quelconque) :

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 la boucle `For...Next` , la boucle `Do...Loop` dispose d'une commande de fin prématurée. La commande `Exit Do` peut 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

Il existe de nombreux types d'utilisation des boucles, par exemple pour rechercher dans des listes, renvoyer des valeurs ou exécuter des tâches mathématiques complexes. L'exemple suivant présente 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) = "Patrick"
      Entry(2) = "Kurt"
      Entry(3) = "Thomas"
      Entry(4) = "Michel"
      Entry(5) = "David"
      Entry(6) = "Catherine"
      Entry(7) = "Suzanne"
      Entry(8) = "Eddy"
      Entry(9) = "Christine"
      Entry(10) = "Jean"

      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. Pour cette raison, 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

```
Sub Test

      ' ... ici se trouve le code de la procédure à proprement parler

End Sub
```

L'exemple définit une procédure nommée `Test` contenant 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 ***fonction,*** 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

      ' ... ici se trouve le 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 ci-dessus peut être appelée à l'intérieur du programme de la façon suivante :

```
Dim A

A = Test
```

Le 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 une 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 n'importe quel type. Le type se déclare de la même manière que pour une variable :

```
Function Test As Integer

      ' ... ici se trouve le 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 et d'une fonction

Dans StarOffice Basic, vous pouvez utiliser les commandes `Exit Sub` et `Exit Function` pour terminer prématurément une procédure ou une fonction, par exemple pour gérer une erreur. Ces commandes interrompent la procédure ou la fonction et retournent 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 vaut à présent 20
End Sub
Sub ChangeValue(TheValue As Integer)
       TheValue = 20
End Sub
```

Dans cet exemple, la valeur A définie dans la fonction `Test` est transmise comme paramètre à la fonction `ChangeValue`. La valeur est ensuite changée à 20 et passée à `TheValue`, qui est 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 indiquer qu'un paramètre doit être passé par valeur, assurez-vous que le mot-clé `ByVal` précède la déclaration de la variable dans l'en-tête de la fonction.

Dans l'exemple précédent, si nous remplaçons la fonction `ChangeValue` par la fonction

```
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`.

La méthode pour passer les paramètres aux procédures et aux fonctions de StarOffice Basic est littéralement identique à celle de VBA. Par défaut, les paramètres sont passés par référence. Pour passer des paramètres par valeur, utilisez le mot-clé `ByVal`. Dans VBA, vous pouvez également utiliser le mot-clé `ByRef` pour forcer le passage d'un paramètre par référence. StarOffice Basic ne supporte pas ce mot-clé, car ce comportement est déjà celui par défaut.

En principe, les fonctions et les procédures dans StarOffice Basic sont `Public`. Les mots-clés `Public` et `Private` utilisés dans VBA ne sont pas supportés par StarOffice Basic.

Paramètres facultatifs

Les fonctions et les procédures ne peuvent être appelées que si tous les paramètres nécessaires sont passés lors de l'appel.

StarOffice Basic vous permet d'indiquer certains paramètres comme étant ***facultatifs***, c'est-à-dire que 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` contrôle si un paramètre a bien été passé ou s'il a été omis.

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

      ' Vérifie si le paramètre B est bien présent
      If Not IsMissing (B) Then
             B_Local = B           ' le paramètre B est présent
      Else
             B_Local = 0           ' le paramètre B a été omis -> valeur par défaut de 0
      End If

      ' ... Début de la fonction en 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 a été omis, une valeur par défaut (en l'occurrence la valeur `0`) est transmise à `B_ Local` à la place du paramètre.

L'option de VBA permettant de définir des valeurs par défaut pour les paramètres facultatifs n'est pas supportée par StarOffice Basic.

Le mot-clé `ParamArray` présent dans VBA n'est pas supporté par StarOffice Basic.

Récursivité

StarOffice Basic supporte désormais la récursivité. 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 "Nombre incorrect pour le calcul de
factorielle !"
   Msgbox CalculateFactorial( 3.14 )   ' Affiche "Nombre incorrect pour le calcul de
factorielle !"
End Sub

Function CalculateFactorial( Number )
   If Number < 0 Or Number <> Int( Number ) Then
       CalculateFactorial = "Nombre incorrect pour le calcul de factorielle !"
   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 de base `0! = 1`.

Notez que le niveau de récursivité dans StarOffice Basic est pour l'instant limité à 500.

Traitement des erreurs

Le traitement correct des situations d'erreur est une des tâches qui prennent le plus de temps lors de l'écriture d'un programme. StarOffice Basic propose un large éventail d'outils simplifiant le traitement des erreurs.

L'instruction On Error

L'instruction `On Error` est l'élément clé de tout traitement d'erreur :

```
Sub Test
      On Error Goto ErrorHandler

      ' ... lance la 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` : .

La commande Resume

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

```
ErrorHandler:

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

      Resume Next
```

Utilisez la commande `Resume Proceed` pour indiquer le point particulier de reprise du 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 le format suivant :

```
Sub Test
      On Error Resume Next

      ' ... accomplit la tâche au cours de laquelle une erreur est susceptible de se
produire

End Sub
```

Utilisez la commande `On Error Resume Next` avec précaution, car elle a un effet global. Pour plus d'informations, voir ***Astuces pour le traitement d'erreur structuré***.

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

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

Alors que VBA synthétise les messages d'erreur dans un objet statistique nommé `Err`, 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.

En 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` ou `Resume`.

Astuces pour le traitement d'erreur structuré

La commande de définition `On Error`, tout comme la commande de retour `Resume`, sont des variantes de la construction `Goto`.

Si vous souhaitez structurer proprement votre code pour éviter de générer des 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 en 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

      ' ... Ici se trouve le code du programme en lui-même

      ' Désactive le traitement des erreurs
      On Error Goto 0
```

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

' Début du traitement des erreurs
ErrorHandler:

	' Contrôle si l'erreur était attendue
	If Err = ExpectedErrorNo Then
		' ... Traitement de l'erreur
	Else
		' ... Avertissement d'erreur imprévue
	End If

	On Error Goto 0				' Désactive le traitement des erreurs
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`) de manière qu'une erreur ultérieure puisse être clairement identifiée.

CHAPITRE 3

La bibliothèque d'exécution de StarOffice Basic

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

Fonctions de conversion

Dans de nombreuses situations, une variable d'un certain type doit être convertie dans un autre type.

Conversions de type implicites et explicites

La manière la plus simple de modifier 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 de caractères, et la variable B un 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 (comme nous l'avons déjà recommandé).

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 une chaîne de caractères.
- **`CInt(Var)`** – convertit tout type de données en une valeur entière.
- **`CLng(Var)`** – convertit tout type de données en une valeur entière longue.
- **`CSng(Var)`** – convertit tout type de données en une valeur simple.
- **`CDbl(Var)`** – convertit tout type de données en une valeur double.
- **`CBool(Var)`** – convertit tout type de données en une valeur logique.
- **`CDate(Var)`** – convertit tout type de données en une valeur de 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 = CInt(B + C)             ' B et C sont d'abord ajoutés l'un à l'autre, puis convertis

                              (cela donne le nombre 2)
A = CStr(B) + Cstr(C)       ' B et C sont convertis en chaîne puis
                            ' combinés (cela donne 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. A reçoit la chaîne 2. 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 reçoit donc 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 nationaux correspondants. Inversement, les méthodes CStr utilisent les paramètres nationaux 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 quels que soient les paramètres nationaux
```

Vérification du contenu de variables

Dans certains cas, la date ne peut être convertie :

```
Dim A As String
Dim B As Date

A = "test"
B = A                   ' Génère un message d'erreur
```

Dans cet exemple, l'assignation de la chaîne test à une variable de date 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                   ' Génère 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. StarOffice Basic propose des fonctions de test pour accomplir ce type de tâche :

- **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 "Message d'erreur."
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 un nombre correct, `ValidInput` reçoit la valeur `0` et le programme émet un message d'erreur.

Alors qu'il existe en Basic des fonctions de test pour vérifier les nombres, les dates et les tableaux, la fonction correspondante pour les valeurs logiques manque. Il est possible cependant 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` du type Boolean et tente de l'assigner à la valeur de transfert. 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.

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 des jeux de caractères

Lorsqu'il traite des chaînes de caractères, StarOffice Basic emploie 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("л")             ' Lettre cyrillique л (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) + "Ceci est un test" + Chr(13)
```

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

Accès aux parties d'une chaîne

StarOffice Basic propose quatre fonctions renvoyant des parties de chaîne :

- **`Left(MyString, Length)`** – renvoie les `Length` premiers caractères de `MyString`.
- **`Right(MyString, Length)`** – renvoie les `Length` derniers caractères de `MyString`.
- **`Mid(MyString, Start, Length)`** – renvoie les `Length` premiers caractères de `MyString` à partir de la position `Start`.
- **`Len(MyString)`** – renvoie le nombre de caractères de `MyString`.

Voici quelques exemples de ces fonctions :

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

MyString = " Ceci est un petit test"

MyResult = Left(MyString,5)         ' Renvoie la chaîne " Ceci"
MyResult = Right(MyString, 5)       ' Renvoie la chaîne " test"
```

```
MyResult = Mid(MyString, 8, 5)      ' Renvoie la chaîne "t un "
MyLength = 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, SearchString, MyString)
```

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(SearchString, MyString, 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, Source, Search)
            If CurrentPos <> 0 Then
                Result = Result + Mid(Source, StartPos, _
                        CurrentPos - StartPos)
                Result = Result + NewPart
                StartPos = CurrentPos + Len(Search)
            Else
                Result = Result + Mid(Source, StartPos, Len(Source))
            End If ' Position <> 0
        Loop
    End If
```

```
    Replace = Result
End Function
```

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  = "Ceci fut mon texte"
Mid(MyString, 6, 3, "est")
```

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 €"
```

Les instructions de formatage utilisées dans VBA pour date et l'heure ne sont pas supportées 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 à l'intérieur du code du 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 des informations de date et d'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 l'heure de `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 = "Dimanche"
case 2
      MyWeekday = "Lundi"
case 3
      MyWeekday = "Mardi"
case 4
      MyWeekday = "Mercredi"
case 5
      MyWeekday = "Jeudi"
case 6
      MyWeekday = "Vendredi"
case 7
      MyWeekday = "Samedi"
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.

Certaines fonctions spécifiques à DOS pour les fichiers et les répertoires ne sont plus proposées par StarOffice 7, ou seulement
avec des fonctionnalités limitées. Par exemple, les fonctions `ChDir`, `ChDrive` et `CurDir` ne sont plus supportées.
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
```

Lorsqu'elle est appelée dans StarOffice Basic, contrairement à ce qui se passe dans VBA, 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).

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.

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.

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.

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.

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 "Le fichier existe."
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 leur date de 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 & " lecture seule "
End IF

If (FileMask AND 16) > 0 Then
       FileDescription = FileDescription & " répertoire "
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".

Les drapeaux utilisés dans VBA pour obtenir les propriétés de fichier *caché*, *fichier système, archive* et *nom de volume* ne sont pas supportés par StarOffice Basic car ils sont propres à Windows et ne sont pas, ou pas complètement, disponibles dans les autres systèmes d'exploitation.

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 nationaux 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, "Ceci est une ligne de test."
```

`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, "Ceci est une ligne de texte"             ' Enregistre une ligne
Print #FileNo, "Ceci est une autre ligne de texte" ' 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)

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

Loop

' Ferme le fichier
Close #FileNo

Msgbox Msg
```

Les différentes lignes sont extraites dans `Do While` une boucle, 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 de messages

`MsgBox` affiche une simple boîte d'information, qui peut posséder un ou plusieurs boutons. Dans sa version la plus simple

```
MsgBox "Voici une 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 permettant 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 "Souhaitez-vous continuer ?",  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'information contient plusieurs boutons, il faut utiliser une valeur de retour pour déterminer celui 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 ("Souhaitez-vous continuer ?",  292) = 6 Then
      ' Clic sur le bouton Oui
Else
      ' Clic sur le bouton Non
End IF
```

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

```
MsgBox "Souhaitez-vous continuer ?",  292, "Titre de la fenêtre"
```

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("Saisissez une valeur :", "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 possibles sont les suivantes :

- 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 pour accéder à StarOffice. Vous pouvez utiliser l'API StarOffice pour créer, ouvrir, modifier et imprimer des documents StarOffice. Elle permet d'étendre la portée fonctionnelle de StarOffice grâce à des macros personnelles et d'écrire des boîtes de dialogue personnalisées.

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

Ce chapitre se concentre sur la manière d'utiliser StarOffice dans StarOffice Basic avec 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 détails sur la façon de travailler avec les différentes parties de l'API StarOffice.

Universal Network Objects (UNO)

StarOffice propose une interface de programmation sous la forme d'Universal Network Objects (UNO, objets réseau universels). Il s'agit d'une interface de programmation orientée objet que StarOffice subdivise en différents objets qui assurent un accès programmable au paquetage Office.

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

Pour pouvoir utiliser un Universal Network Object dans StarOffice Basic, vous devez déclarer une variable pour l'objet associé. Cette déclaration se fait avec l'instruction `Dim` (voir le chapitre 2). Il faut employer le type `Object` pour déclarer une variable objet :

```
Dim Obj As Object
```

Cet appel déclare une variable objet nommée `Obj`.

La variable objet ainsi créée doit ensuite être initialisée pour pouvoir être utilisée. Cela peut être accompli avec 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 qui supporte le service*** `com.sun.star.frame.Desktop`. Le terme "service" employé dans StarOffice Basic correspond donc aux termes de ***type*** et de ***classe*** employés dans d'autres langages de programmation.

Il existe cependant une différence principale : un Universal Network Object peut supporter plusieurs services en même temps. Certains services UNO supportent à leur tour 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 terminer le programme.

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 supporte. Un objet VBA est toujours assigné à une classe unique. Un objet StarOffice Basic peut, lui, supporter plusieurs services.

Propriétés et méthodes

Un objet dans StarOffice Basic dispose d'une gamme de propriétés et de méthodes qui peuvent être appelées par son intermédiaire.

Propriétés

Les propriétés sont comme les propriétés d'un objet ; par exemple `Filename` et `Title` pour un objet `Document`.

Les propriétés sont définies grâce à une simple assignation :

```
Document.Title = "Manuel de programmation StarOffice 7.0"
Document.Filename = "progman.sxv"
```

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

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 (méthodes `get` et `set`). La propriété a été virtuellement imitée à partir de deux méthodes. Les objets 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 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 et qui permettent de l'appeler. L'objet `Document` précédent, par exemple, propose une méthode `Save`, pouvant être appelée de la manière suivante :

```
Document.Save()
```

Les méthodes, tout comme les 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)
```

indique 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 dispose de 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, comme `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 de module et de service, UNO utilise celui d' ***"interface".*** Ce terme est sans doute familier aux programmeurs Java, mais n'apparaît pas en Basic.

Une interface combine plusieurs méthodes. Au sens le plus strict du mot, un service dans UNO ne supporte pas des méthodes, mais des interfaces, qui, elles, proposent 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++, car dans ces langages, 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 de 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 ont été indiquées à nouveau 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 – supportent quelles propriétés, méthodes et interfaces et comment le 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 ainsi que le Developer's Guide et la référence de l'API.

La méthode supportsService

Un grand nombre d'objets UNO supportent la méthode `supportsService`, grâce à laquelle il est possible de déterminer si un objet supporte un service donné ou non. L'appel

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

par exemple, détermine si l'objet `TextElement` supporte le service `com.sun.star.text.Paragraph`.

Propriétés de débogage

Chaque objet UNO de StarOffice Basic connaît les propriétés, les méthodes et les interfaces qu'il contient déjà. 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 supportant un objet.

Le code 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 supportées 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`, faites attention au fait que la fonction renvoie l'ensemble des propriétés qu'un service donné peut supporter en théorie. Vous n'avez cependant aucune assurance que l'objet concerné peut 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 plus d'informations sur les services disponibles ainsi que sur leurs interfaces, méthodes et propriétés dans la référence pour l'API StarOffice. Elle se trouve sur le site www.openoffice.org :

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

Présentation de quelques interfaces centrales

Certaines interfaces de StarOffice se retrouvent dans de nombreuses parties 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 plus courantes d'entre elles.

L'origine des objets est expliquée plus loin dans ce manuel. Pour l'instant, nous nous contentons 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 pour créer des 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. De tels objets et services sont également connus sous le nom de ***services indépendants du contexte***.

À côté de ces services indépendants du contexte, il existe également des ***services contextuels*** dont les objets sont destinés à être utilisés avec d'autres objets. Un objet de dessin d'un 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 l'objet dépend. La méthode `createInstance`, définie dans l'interface `XMultiServiceFactory`, est utilisée en particulier 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")
```

Un modèle de paragraphe dans un document texte se crée de la même manière :

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

Accès par nom aux objets subordonnés

Les interfaces `XNameAccess` et `XNameContainer` sont utilisées dans les objets contenant des objets subordonnés, qui peuvent être adressés avec 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.

L'interface com.sun.star.container.XNameAccess

L'objet Sheet d'un classeur fournit un bon exemple d'utilisation de `XNameAccess`. Il combine 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("Feuille1")
```

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 avec 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("Feuille1") Then
      MsgBox " Feuille 1 existe"
Else
      MsgBox "Feuille1 n'existe pas"
End If
```

L'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`.

L'exemple suivant met cela en pratique. Il appelle un document texte contenant un objet `StyleFamilies` qu'il utilise 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 indice aux objets subordonnés

Les interfaces `XIndexAccess` et `XIndexContainer` sont utilisées dans les objets contenant des objets subordonnés et qui peuvent être adressés avec un indice.

`XIndexAccess` fournit les méthodes pour accéder aux différents objets. `XIndexContainer` fournit les méthodes pour insérer et supprimer des éléments.

L'interfacecom.sun.star.container.XIndexAccess

`XIndexAccess` propose les méthodes `getByIndex` et `getCount` pour appeler les objets subordonnés. `getByIndex` renvoie un objet avec un indice donné. `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)
      ' Édition de la feuille
Next I
```

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

L'interface com.sun.star.container.XIndexContainer

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

Accès itératif aux objets subordonnés

Dans certains cas, il se peut qu'un objet contienne une liste d'objets subordonnés ne pouvant être adressés ni par nom ni par index. C'est à ce type de situations que sont destinées les interfaces `XEnumeration` et `XenumerationAccess`. Elles proposent un mécanisme permettant d'atteindre tous les différents éléments subordonnés d'un objet, l'un après l'autre, sans recourir à une méthode d'adressage direct.

Les 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`. Celles-ci 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 un par un, 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

L'API StarOffice a été structurée de manière qu'un maximum de ses parties puissent être utilisées universellement pour différentes tâches. Cela concerne les interfaces et les services pour créer, ouvrir, enregistrer, convertir et imprimer des documents ainsi que pour l'administration des modèles. Comme les fonctions de ce type sont disponibles dans tous les types de documents, elles sont expliquées en premier dans ce chapitre.

StarDesktop

Il existe deux services très fréquemment appelés pour travailler sur des documents :

- Le service `com.sun.star.frame.Desktop`, similaire au service principal de StarOffice. Il fournit les fonctions pour l'objet cadre de StarOffice, sous lequel sont classées toutes les fenêtres de document. Ce service permet également de créer, d'ouvrir et d'importer des documents.
- Les fonctions de base pour les objets Document individuels sont fournis par le service `com.sun.star.document.OfficeDocument`. Il propose les méthodes pour l'enregistrement, l'export et l'impression des documents.

Le service `com.sun.star.frame.Desktop` s'ouvre automatiquement au lancement de StarOffice. Pour ce faire, StarOffice crée un objet accessible par le nom global `StarDesktop`.

L'interface la plus importante de `StarDesktop` est `com.sun.star.frame.XComponentLoader`. Elle comprend principalement la méthode `loadComponentFromURL`, qui sert à créer, importer et ouvrir des documents.

Le nom de l'objet `StarDesktop` remonte à StarOffice 5, où toutes les fenêtres de document étaient incluses dans une application commune nommée StarDesktop. Dans la version actuelle de StarOffice, StarDesktop ne correspond à plus aucun objet visible. Le nom StarDesktop a cependant été conservé pour l'objet cadre de StarOffice car il indique clairement qu'il s'agit d'un objet de base pour l'ensemble de l'application.

L'objet `StarDesktop` fait office de successeur de l'objet `Application` de StarOffice 5 qui servait auparavant d'objet racine. Mais contrairement à l'ancien objet `Application`, il est surtout destiné à l'ouverture de nouveaux 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.

Alors qu'on accède au document actif dans Word via `Application.ActiveDocument` et dans Excel via `Application.ActiveWorkbook`, 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 de base sur les documents dans 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 fichiers sont structurés pour les documents StarOffice, ainsi que le format dans lequel les fichiers sont enregistrés.

Noms de fichier en notation URL

Comme StarOffice a été conçue pour être 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 le standard Internet RFC 1738 pour les noms de fichier. Les noms de fichier standard utilisant ce système commencent par le préfixe

`file:///`

suivi du chemin local. Si le nom du fichier contient des sous-répertoires, ceux-ci sont indiqués par une *barre oblique* unique, 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")
            ' donne file:///C:/doc/test.sxw

MsgBox ConvertFromUrl("file:///C:/doc/test.sxw")
            '  donne (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.

Le standard Internet RFC 1738, utilisé ici, autorise l'emploi des caractères `0-9`, `a-z` et `A-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 pour cent. 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 dispose d'un format de fichier XML. Grâce à l'emploi de XML, l'utilisateur a la possibilité d'ouvrir et d'éditer les fichiers dans d'autres programmes.

Compression des fichiers

Comme XML utilise des fichiers texte standard, les fichiers obtenus sont généralement très volumineux. Pour cette raison, StarOffice compresse les fichiers et les enregistre sous forme de fichier ZIP.

Grâce à une option de la méthode `storeAsURL`, l'utilisateur peut enregistrer directement les fichiers XML d'origine. Voir Options de la méthode storeAsURL page 85.

Création, ouverture et import de documents

Les documents peuvent être ouverts, importés et créés avec la méthode

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

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

Comme second paramètre, `loadComponentFromURL` attend un nom pour l'objet cadre de la fenêtre que StarOffice crée en interne pour son administration. Le nom prédéfini `_blank` est généralement utilisé ici, et indique à StarOffice de créer une nouvelle fenêtre. Il est également possible de spécifier `_hidden`, qui permet de charger le document correspondant mais de le garder invisible.

L'utilisateur peut ouvrir un document StarOffice avec ces seuls paramètres, les deux derniers pouvant 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 test.sxw et l'affiche dans une nouvelle fenêtre.

Il est possible d'ouvrir autant de documents que vous le voulez avec cette méthode et de les modifier en utilisant les objets Document retournés.

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

Remplacement du contenu de la fenêtre de document

Les valeurs citées plus haut `_blank` et `_hidden` du paramètre `Frame` garantissent la création par StarOffice d'une nouvelle fenêtre pour chaque appel de `loadComponentFromURL`. Dans certaines situations, il peut être utile de remplacer le contenu d'une fenêtre existante. Dans ce cas, l'objet cadre de la fenêtre doit contenir un nom explicite. Notez que ce nom ne doit pas commencer par un caractère de soulignage. De plus, le paramètre `SearchFlags` doit être activé pour que le framework correspondant soit créé au cas où il n'existerait pas. La constante correspondante pour `SearchFlags` est :

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

L'exemple suivant montre comment remplacer le contenu d'une fenêtre ouverte grâce au paramètre Frame et à `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 second 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` fournissant à StarOffice diverses options pour l'ouverture et la création de documents. Le champ de données doit contenir une structure PropertyValue pour chaque option, et le nom de l'option y est enregistré sous forme de chaîne, tout comme sa valeur associée.

`loadComponentFromURL` supporte les options suivantes :

- **`AsTemplate (Boolean)`** – s'il vaut true, la méthode charge un nouveau document sans titre depuis l'URL donné. S'il vaut false, la méthode charge les fichiers de modèle pour édition.

- **`CharacterSet (String)`** – définit le jeu de caractères utilisé par un document.
- **`FilterName (String)`** – spécifie un filtre spécial pour la fonction `loadComponentFromURL`. Les noms des filtres disponibles sont définis dans le fichier `\share\config\registry\instance\org\openoffice\office\TypeDetection.xml`.
- **`FilterOptions (String)`** – définit des options supplémentaires pour les filtres.
- **`JumpMark (String)`** – après avoir ouvert un document, le programme passe à la position définie dans JumpMark.
- **`Password (String)`** – transmet un mot de passe pour un fichier protégé.
- **`ReadOnly (Boolean)`** – charge un document en lecture seule.

L'exemple suivant montre comment un fichier texte séparé par des virgules dans StarOffice Calc peut être ouvert avec l'option `Filtername`.

```
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` couvre une seule valeur car il contient une seule option. La propriété `Filtername` définit si StarOffice utilise un filtre de texte StarOffice Calc pour ouvrir les fichiers.

Création de nouveaux documents

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

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

```
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 supporte le service `com.sun.star.document.OfficeDocument`, qui fournit à son tour deux interfaces centrales :

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

En passant à StarOffice 7, 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 ce cas, 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())
```

En plus des méthodes ci-dessus, `com.sun.star.frame.XStorable` contient également certaines méthodes d'aide utiles lors de l'enregistrement des documents. Ces méthodes sont les suivantes :

- **`hasLocation()`** – spécifie si un URL a déjà été assigné au document.
- **`isReadonly()`** - spécifie si un document est doté d'une protection en lecture seule.
- **`isModified()`** - spécifie si un document a été modifié depuis son dernier enregistrement.

Le code permettant d'enregistrer un document peut être complété par ces options afin que celui-ci ne soit enregistré que si l'objet a vraiment été modifié et que le nom de fichier ne soit demandé que si cela 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 si tel est le 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 n'a pas d'URL ou s'il a été ouvert en lecture seule, il est enregistré sous un nouvel URL.

Options de la méthode storeAsURL

Comme pour la méthode `loadComponentFromURL`, certaines options peuvent être spécifiées sous la forme d'un champ de données `PropertyValue` à l'aide de la méthode `storeAsURL`. Elles déterminent la procédure utilisée par StarOffice pour enregistrer un document. `storeAsURL` comporte les options suivantes :

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

L'exemple suivant montre comment l'option `Overwrite` peut être associée à `storeAsURL` :

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

' ... Initialisation de Doc

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

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

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

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

Impression de documents

Comme pour l'enregistrement, les documents sont imprimés directement via 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 de StarOffice :

- **`CopyCount (Integer)`** – spécifie le nombre d'exemplaires à imprimer.
- **`FileName (String)`** – imprime le document dans le fichier spécifié.
- **`Collate (Boolean)`** – indique à l'imprimante de rassembler les pages de chaque exemplaire.
- **`Sort (Boolean)`** – tri les pages lors de l'impression de plusieurs exemplaires (`CopyCount` > 1).
- **`Pages (String)`** – comporte la liste des pages à imprimer (selon la syntaxe indiquée dans la boîte de dialogue d'impression).

L'exemple suivant montre comment plusieurs pages d'un document peuvent être imprimées 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 sélectionne l'imprimante. Cette propriété reçoit un champ de données `PropertyValue` avec les paramètres suivants :

- **`Name (String)`** – spécifie le nom de l'imprimante.
- **`PaperOrientation (Enum)`** – spécifie 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)`** – spécifie 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)`** – spécifie la taille du papier en centièmes de millimètre.

L'exemple suivant montre 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="Ma Laserjet HP"

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 contribuent largement à simplifier 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 par une modification centrale dans le document. Selon le type de documents concerné, StarOffice reconnaît toute une gamme de types de modèles différents.

StarOffice Writer supporte

- des modèles de caractère,
- des modèles de paragraphe,
- des modèles de cadre,
- des modèles de page
- des modèles de numérotation

StarOffice Calc supporte

- des modèles de cellule
- des modèles de page

StarOffice Impress supporte

- des modèles d'élément de caractères
- des 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 par 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 par un indice :

```
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 par rapport à l'exemple précédent affiche successivement les noms de tous les modèles de cellule dans une boîte de message.

détails sur diverses options de formatage

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

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

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

Sur l'utilisation des modèles, vous trouverez davantage d'informations au paragraphe ***Valeurs par défaut des propriétés de caractère et de paragraphe*** du chapitre 6, Documents texte.

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 ne comportent pas 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. 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 texte, des 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 essentiellement 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.

> La conception de l'API de StarOffice 7 pour StarOffice Writer diffère de celle de la version précédente. L'ancienne version de l'API était axée sur l'utilisation de l'objet `Selection`, très orientée sur l'idée d'interface utilisateur pour l'utilisateur final, axée sur la mise en évidence à l'aide de la souris.
>
> L'API de StarOffice 7 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'ancien 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 ne sont ni nommés ni indexés, et il n'est donc pas possible d'accéder ***directement*** à des paragraphes individuels. 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.

Dans l'utilisation de l'objet `Enumeration`, une particularité doit néanmoins être notée : il ne renvoie pas uniquement des paragraphes, mais également des tableaux (à strictement parler, dans StarOffice Writer, un tableau est un type spécial de paragraphe). C'est pourquoi, avant d'accéder à un objet renvoyé, vous devez vérifier qu'il supporte le service `com.sun.star.text.Paragraph` pour les paragraphes ou le service `com.sun.star.text.TextTable` pour les tableaux.

L'exemple qui suit parcourt le contenu d'un document texte dans une boucle et utilise un message dans chaque cas pour informer 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

' Création d'un objet Document
Doc = StarDesktop.CurrentComponent

' Création d'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 "Le bloc actif contient un tableau."
	End If

	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		MsgBox "Le bloc actif contient un paragraphe."
	End If
Wend
```

Cet exemple crée un objet Document `Doc` qui référence le document StarOffice actif. À l'aide de `Doc`, l'exemple crée ensuite un objet `Enumeration` qui parcourt chaque partie du texte (paragraphes et tableaux) et assigne l'élément actif à l'objet `TextElement`. L'exemple utilise la méthode `supportsService` pour vérifier si l'objet `TextElement` est un paragraphe ou un tableau et afficher le message correspondant.

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` respectivement 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***.

Le contenu de la procédure décrite ici pour accéder aux paragraphes d'un texte est comparable à la liste `Paragraphs` utilisée dans VBA et située dans les objets `Range` et `Document` de VBA. Mais contrairement à VBA, où les paragraphes sont accessibles par leur indice (par l'appel `Paragraph(1)`, par exemple), il est nécessaire, dans StarOffice Basic, 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 parfois détruire le formatage.

En effet, un paragraphe est constitué de sous-objets individuels. Chacun de ces sous-objets contient ses propres informations de formatage. Si un paragraphe contient en son milieu un mot en gras, par exemple, il est représenté dans StarOffice par trois portions de paragraphe : la portion précédant le gras, le mot en gras et la portion située après le gras, 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. Pour cela, 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 paragraphes
While Enum1.hasMoreElements
	TextElement = Enum1.nextElement
	If TextElement.supportsService("com.sun.star.text.Paragraph") Then
		Enum2 = TextElement.createEnumeration

		' Boucle parcourant tous les sous-paragraphes
		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. L'exemple de code modifie le contenu de chacune de ces portions de paragraphe 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, leurs informations de formatage sont conservées lors du remplacement de la chaîne.

Formatage

Le texte peut être formaté de différentes manières . 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.

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.

Dans l'ancienne API de StarOffice, un texte était essentiellement formaté à l'aide de l'objet `Selection` et ses 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. Une liste des propriétés de caractère et de paragraphe disponibles figure dans les paragraphes suivants.

Propriétés de caractère

Les propriétés de format se rapportant individuellement aux caractères 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 supporter le service `com.sun.star.style.CharacterProperties`. StarOffice reconnaît une gamme étendue de services supportant 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 de StarOffice. La liste suivante décrit les plus importantes :

- **`CharFontName (String)`** – le nom du type de police sélectionné.
- **`CharColor (Long)`** – la couleur du texte.
- **`CharHeight (Float)`** – CharHeight **`(Float)`** – la hauteur du caractère en points (pt).
- **`CharUnderline (Constant group)`** – le type de soulignage (constantes selon `com.sun.star.awt.FontUnderline`).
- **`CharWeight (Constant group)`** – la graisse de la police (constantes selon `com.sun.star.awt.FontWeight`).
- **`CharBackColor (Long)`** – la couleur d'arrière-plan.
- **`CharKeepTogether (Boolean)`** – suppression des retours à la ligne automatiques.
- **`CharStyleName (String)`** – le 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 la distance entre le paragraphe et le bord de la page ou l'interligne. Les propriétés de paragraphe sont accessibles par 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` supportent également 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 de StarOffice. Les plus courantes sont les suivantes :

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

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

	End If
Wend

' Écriture des balises HTML de fin
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, déjà abordée 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 via un modèle (formatage indirect)
- **`com.sun.star.beans.PropertyState.AMBIGUOUS_VALUE` –** l'origine du formatage est inconnue.
  Cet état apparaît, par exemple, lorsque la méthode interroge la propriété graisse d'un paragraphe comportant des mots en gras et d'autres en maigre.

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 trouve une portion de paragraphe correspondante, il supprime le formatage direct à l'aide de la méthode `setPropertyToDefault` et assigne un modèle de caractère `MyBold` à la portion de paragraphe correspondante.

```
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 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 paragraphe
		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 permettant 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. En revanche, ils ne suffisent pas pour résoudre de nombreux problèmes. 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`.

TextCursor

Un objet `TextCursor` dans l'API de 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.

Attention ! 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'un 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 supporte 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. Par exemple, dans l'exemple suivant, l'objet `TextCursor`

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

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 par l'objet `TextCursor` commence donc après le septième caractère du texte et se termine après le dixième.

Les méthodes centrales 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 `True` si le `TextCursor` est au début d'un mot.
- **`isEndOfWord ()`** - renvoie `True` si le `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 au début de la phrase suivante.
- **`gotoPreviousSentence (Expand)`** – déplacement au début de la phrase précédente.
- **`isStartOfSentence ()`** - renvoie `True` si le `TextCursor` est au début d'une phrase.
- **`isEndOfSentence ()`** - renvoie `True` si le `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 `True` si le `TextCursor` est au début d'un paragraphe.

- **`isEndOfParagraph ()`** – renvoie `True` si le `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 supportant également le service `com.sun.star.text.TextCursor` :

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

Formatage du texte avec TextCursor

Le service `com.sun.star.text.TextCursor` 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 `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`** – un saut de paragraphe.
- **`LINE_BREAK`** – un retour à la ligne à l'intérieur d'un paragraphe.
- **`SOFT_HYPHEN`** – un point possible de coupure de mot.
- **`HARD_HYPHEN`** – un point obligatoire de coupure de mot.
- **`HARD_SPACE`** – un 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 20e 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'opération d'insertion. Si le paramètre `True` est transmis 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. Un objet `SearchDescriptor` supporte le service `com.sun.star.util.SearchDescriptor` et 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="tout texte"
```

Du points 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 contenant des caractères génériques.
- **`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)`** - le nombre de caractères pouvant être ajoutés dans une recherche de similarité.
- **`SearchSimilarityExchange (Short)`** - le nombre de caractères pouvant être différents dans une recherche de similarité.
- **`SearchSimilarityRemove (Short)`** - le nombre de caractères pouvant être absents dans une recherche de similarité.
- **`SearchSimilarityRelax (Boolean)`** - tient compte simultanément de toutes les règles de déviation pour l'expression de recherche.

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
      ' Traitement du résultat de la recherche
      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 le terme "chiffre d'affaires" peut être recherché dans un texte et comment le résultat peut être mis en gras. Une recherche de similarité est utilisée pour que le terme "chiffre d'affaires", mais également sa forme plurielle "chiffres d'affaires" et ses variantes comme "chiffre daffaires" soient recherchées. Les expressions trouvées diffèrent au maximum de deux lettres de l'expression recherchée :

```
Dim SearchDesc As Object
Dim Doc As Object

Doc = StarDesktop.CurrentComponent

SearchDesc = Doc.createSearchDescriptor
SearchDesc.SearchString="chiffre d'affaires"
SearchDesc.SearchSimilarity = True
SearchDesc.SearchSimilarityAdd = 2
SearchDesc.SearchSimilarityExchange = 2
SearchDesc.SearchSimilarityRemove = 2
SearchDesc.SearchSimilarityRelax = False

Found = Doc.findFirst (SearchDesc)

Do While Found
Found.CharWeight = com.sun.star.awt.FontWeight.BOLD
Found = Doc.findNext( Found.End, Search)
Loop
```

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. Alors que, dans VBA, l'objet auxiliaire chargé de cela est accessible par la propriété `Find` de l'objet `Range`, il est créé dans StarOffice Basic par un appel `createSearchDescriptor` ou `createReplaceDescriptor` de l'objet Document. Même les propriétés et méthodes de recherche disponibles sont différentes.

Comme dans l'ancienne API de StarOffice, la recherche et le remplacement de texte dans la nouvelle API sont également effectués à l'aide de l'objet Document. Mais alors qu'il existait précédemment un objet `SearchSettings` destiné spécialement à définir les options de recherche, 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 également 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 supporte le service `com.sun.star.util.ReplaceDescriptor`. Toutes les propriétés de l'objet `SearchDescriptor` décrites dans le paragraphe précédent sont également supportées 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 I = 0 To 5
      Replace.SearchString = BritishWords(I)
      Replace.ReplaceString = USWords(I)
      Doc.replaceAll(Replace)
Next I
```

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 caractères génériques

La fonction de remplacement de StarOffice est particulièrement efficace lorsqu'elle est associée à des caractères génériques. Ces derniers permettent de définir une expression de recherche variable avec des substituants et des caractères spéciaux au lieu d'une valeur fixe.

Les caractères génériques supportés par StarOffice sont décrits en détail dans la section de 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 `*` 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 de caractères génériques `^$` :

```
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 que du texte

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 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 supportent 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)`** – l'énumération de tous les `AnchorTypes` qui supportent 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** à l'aide de la méthode `insertTextContent` de l'objet Text.
- `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`)

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. Alors que la méthode `Document.Tables.Add` gère la création et le paramétrage du tableau dans VBA, 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 document texte :

```
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)

      ' Édition du tableau
Next I
```

Les tableaux de texte sont disponibles dans StarOffice 7 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)`** – la couleur d'arrière-plan du tableau.
- **`BottomMargin (Long)`** – la marge inférieure en centièmes de millimètre.
- **`LeftMargin (Long)`** – la marge gauche en centièmes de millimètre.
- **`RightMargin (Long)`** – la marge droite en centièmes de millimètre.
- **`TopMargin (Long)`** – la marge supérieure en centièmes de millimètre.
- **`RepeatHeadline (Boolean)`** – répétition ou non de l'en-tête du tableau sur chaque page.
- **`Width (Long)`** – la largeur absolue du tableau en centièmes de millimètre.

Lignes

Un tableau est constitué d'une liste de lignes. L'exemple suivant montre comment il est possible d'accéder aux lignes d'un tableau et de 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, supportant 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)`** – la couleur d'arrière-plan de la ligne.
- **`Height (Long)`** – la 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

Comme pour les lignes, on accède aux colonnes à en appliquant les méthodes `getByIndex`, `getCount`, `insertByIndex` et `removeByIndex` à l'objet `Column`, accessible 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 = "ligne : " & CStr(RowIndex) + ", column: " & CStr(ColIndex)
      Next
Next
```

Une cellule de tableau est comparable à du texte standard. Elle supporte 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 associées 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 modifiant 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 peuvent être placés à un emplacement quelconque sur la page et ne sont pas inclus dans l'enchaînement.

Comme pour tous les objets `TextContent`, la création d'un cadre texte est distincte de son insertion dans le document.

```
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.

Les cadres texte sont l'équivalent pour StarOffice des cadres de position utilisés dans Word. Alors que VBA utilise la méthode `Document.Frames.Add` à cet effet, la création dans StarOffice Basic est effectuée par la procédure ci-dessus à l'aide d'un objet TextCursor ainsi que la méthode `createInstance` de l'objet Document.

Les objets cadres texte 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`, supporté également par chaque service TextFrame. Les propriétés centrales sont les suivantes :

- **`BackColor (Long)`** – la couleur d'arrière-plan du cadre texte.
- **`BottomMargin (Long)`** – la marge inférieure en centièmes de millimètre.
- **`LeftMargin (Long)`** – la marge gauche en centièmes de millimètre.
- **`RightMargin (Long)`** – la marge droite en centièmes de millimètre.
- **`TopMargin (Long)`** – la marge supérieure en centièmes de millimètre.
- **`Height (Long)`** – la hauteur du cadre texte en centièmes de millimètre.
- **`Width (Long)`** – la largeur du cadre texte en centièmes de millimètre.
- **`HoriOrient (const)`** – l'orientation horizontale du cadre texte (selon `com.sun.star.text.HoriOrientation`).
- **`VertOrient (const)`** – l'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 cadres texte 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 = "Ceci est un petit 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. La chaîne "Ceci est un petit test !" est enfin assignée au cadre texte.

Champs texte

Les champs texte sont des objets `TextContent`, car ils ajoutent une extension logique au-delà du texte pur. Des champs texte peuvent être insérés dans un document texte par les mêmes méthodes que celles utilisées 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 texte correspondant à la date du jour au début du document texte actif. La valeur `True` de la propriété `IsDate` entraîne l'affichage de la date uniquement et non de l'heure. La valeur `False` pour `IsFixed` garantit que la date est actualisée automatiquement à l'ouverture du document.

Alors que le type d'un champ est spécifié dans VBA par un paramètre de la méthode `Document.Fields.Add`, il est défini dans StarOffice Basic par le nom du service responsable de ce

type de champ.

Autrefois, on accédait aux champs texte par tout un ensemble de méthodes que StarOffice rendait disponibles dans l'ancien objet `Selection` (`InsertField`, `DeleteUserField`, `SetCurField`, par exemple).

Dans StarOffice 7, les champs sont administrés à l'aide d'un concept orienté objet. Pour créer un champ texte, il faut d'abord créer un champ texte du type requis et l'initialiser avec les propriétés adéquates. Ce champ 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 types de champ les plus importants et leurs propriétés sont décrits dans les sections suivantes.

Outre l'insertion de champs texte, la recherche des champs dans un document peut également être une tâche importante. L'exemple suivant montre comment tous les champs texte d'un document texte peuvent être parcourus dans une boucle et comment leur type peut être contrôlé.

```
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/heure"
	ElseIf TextField.supportsService("com.sun.star.text.TextField.Annotation") Then
		MsgBox "Annotation"
	Else
		MsgBox "inconnu"
	End If

Wend
```

Le point de départ dans la détermination des champs texte présents est la liste `TextFields` de l'objet Document. Cet exemple crée un objet `Enumeration` basé sur cette liste, par lequel tous les champs texte peuvent être interrogés successivement dans une boucle. Pour chaque champ texte trouvé, le service supporté est consulté à l'aide de la méthode `supportsService`. 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 "inconnu".

La liste ci-dessous répertorie les champs texte les plus importants et leurs propriétés associées. Une liste complète de tous les champs texte est fournie dans la référence de l'API dans le module `com.sun.star.text.TextField`. (Lorsque vous répertoriez le nom du service d'un champ

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 texte

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

renvoient le nombre de pages, de mots ou de caractères d'un texte. Ils supportent la propriété suivante :

- **`NumberingType (const)`** - le 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 texte `com.sun.star.text.TextField.PageNumber`. Vous pouvez spécifier les propriétés suivantes :

- **`NumberingType (const)`** - le format de numérotation (instructions selon les constantes de `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 texte supportant le service `com.sun.star.text.TextField.PageNumber`. Les lignes de l'en-tête et du 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` est définie à `True`. Le champ 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. Un clic sur ce symbole ouvre un champ texte dans lequel il est possible d'enregistrer un commentaire à cet emplacement du texte. Un champ d'annotation comporte les propriétés suivantes.

- **`Author (String)`** - le nom de l'auteur.
- **`Content (String)`** - un texte de commentaire.
- **`Date (Date)`** - la 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 supporte les propriétés suivantes :

- **`IsFixed (Boolean)`** – s'il est défini à `True`, l'heure de l'insertion reste inchangée ; s'il est défini à False, elle est actualisée à chaque ouverture du document.
- **`IsDate (Boolean)`** – si elle est définie à `True`, le champ affiche la date du jour, sinon il affiche l'heure actuelle.
- **`DateTimeValue (struct)`** – le contenu actif du champ (structure `com.sun.star.util.DateTime`)
- **`NumberFormat (const)`** – le format dans lequel la date ou l'heure sont décrites.

Nom / Numéro de chapitre

Le nom du chapitre actif est accessible par un champ 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 = "Mes repères de texte"
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 dans le document par la méthode `insertTextContent` à l'emplacement du curseur.

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("Mes repères de texte")

Cursor = Doc.Text.createTextCursorByRange(Bookmark.Anchor)
Cursor.String = "Emplacement du repère de texte"
```

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 des classeurs commandées par un programme. Ce chapitre décrit comment piloter les services, méthodes et propriétés concernées des classeurs.

La première section aborde la structure de base des classeurs et indique comment accéder au contenu de chaque cellule et l'éditer.

La seconde section, axée sur les zones de cellules et sur les options de recherche et de remplacement du contenu des cellules, se concentre sur la manière d'éditer efficacement les classeurs.

L'objet Range, qui permet d'accéder à toutes les zones d'une feuille, a été étendu dans la nouvelle API.

Structure des documents à base de tables (classeurs)

L'objet Document d'un classeur est basé sur le service `com.sun.star.sheet.SpreadsheetDocument`. Chacun de ces documents peut comporter plusieurs feuilles de calcul. Dans ce manuel, un ***document à base de 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.

La terminologie des feuilles de calcul et de 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 *Worksheet*, ces objets sont respectivement appelés *SpreadsheetDocument* et *Sheet* dans StarOffice Basic.

Classeurs

On peut accéder à chaque feuille de calcul d'un classeur par 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("Feuille 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` supporte le service `com.sun.star.sheet.Spreadsheet`. Outre les différentes interfaces qu'il propose pour l'édition du contenu, ce service dispose des propriétés suivantes :

- **`IsVisible (Boolean)`** – la feuille de calcul est visible.
- **`PageStyle (String)`** – le nom du modèle de page pour la feuille de calcul.

Création, suppression et attribution d'un nouveau à des feuilles

La liste `Sheets` d'un classeur permet également de créer, de supprimer et de renommer des feuilles individuelles. L'exemple qui suit utilise la méthode `hasByName` pour vérifier si une feuille nommée ***MaFeuille*** 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 dans `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("MaFeuille") Then
      Sheet = Doc.Sheets.getByName("MaFeuille")
Else
      Sheet = Doc.createInstance("com.sun.star.sheet.Spreadsheet")
      Doc.Sheets.insertByName("MaFeuille", Sheet)
End If
```

Les méthodes `getByName` et `insertByName` proviennent de l'interface `com.sun.star.container.XnameContainer` décrite au chapitre 4.

Lignes et colonnes

Chaque feuille contient une liste de ses lignes et de ses colonnes. Ces listes sont accessibles par les propriétés `Rows` et `Columns` de l'objet Spreadsheet et supportent les services `com.sun.star.table.TableColumns` et / ou `com.sun.star.table.TableRows`.

L'exemple qui suit crée deux objets qui renvoient à la première ligne et à la première colonne d'une feuille et stocke les références dans les variables objet `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 supportent le service `com.sun.star.table.TableColumn`, dont les propriétés sont les suivantes :

- **`Width (long)`** – la largeur de la colonne en centièmes de millimètre.
- **`OptimalWidth (Boolean)`** – définit la largeur d'une colonne à sa valeur optimale.
- **`IsVisible (Boolean)`** – affiche une colonne.
- **`IsStartOfNewPage (Boolean)`** – à l'impression, crée un saut de page avant une colonne.

La largeur de la colonne n'est optimisée que lorsque la propriété `OptimalWidth` est réglée sur `True`. Si la largeur d'une cellule individuelle est modifiée, celle de la colonne qui la contient ne change pas. 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)`** – la hauteur de la ligne en centièmes de millimètre.
- **`OptimalHeight (Boolean)`** – définit la hauteur de la ligne à sa valeur optimale.
- **`IsVisible (Boolean)`** – affiche la ligne.
- **`IsStartOfNewPage (Boolean)`** – à l'impression, crée un saut de page avant la ligne.

Si la propriété `OptimalHeight` d'une ligne est définie à `True`, sa hauteur change automatiquement lorsque celle d'une cellule qu'elle contient est modifiée. L'optimisation automatique continue 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
```

Les listes `Rows` et `Columns` sont accessibles par un indice dans StarOffice Basic. Contrairement à VBA, la première colonne a l'indice 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 (indice 3, la numérotation commençant à 0). Le second paramètre spécifie le nombre de colonnes à insérer (dans cet exemple : un).

La méthode `removeByIndex` supprime la sixième colonne (indice 5). Là aussi, le second paramètre spécifie le nombre de colonnes à supprimer.

Les méthodes d'insertion et de suppression des lignes utilisent l'objet `Rows` de la même manière que l'édition des colonnes utilisait l'objet `Columns`.

Cellules

Une feuille de calcul est constituée d'une liste bidimensionnelle de cellules. 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 qui suit 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"
```

En plus de ses coordonnées numériques, chaque cellule d'une feuille porte un nom : par exemple, la cellule supérieure gauche d'une feuille est nommée `A1`. La lettre `A` représente la colonne et le numéro `1` la ligne. Il est important de ne pas confondre le ***nom*** et la ***position*** d'une cellule, car la numérotation de la ligne commence à 1 pour le nom et à 0 pour la position.

Dans StarOffice, une cellule peut 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, un texte et une formule dans les champs A1 à A3.

Les propriétés `Value`, `String` et `Formula` prennent le pas sur la méthode `PutCell` pour définir les valeurs d'une cellule de tableau.

StarOffice traite le contenu saisi par la propriété `String` dans les cellules comme du texte, même si ce contenu est un nombre. Les nombres saisis ainsi sont alignés à gauche dans la cellule au lieu

d'être alignés à droite. Il convient également de différencier le texte et les nombres dans l'utilisation des formules :

```
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 1000, la formule A1+A2 renvoie la valeur 100. En effet, le contenu de la cellule A2 a été saisi comme chaîne et non comme un nombre.

Pour vérifier si le contenu d'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 "Contenu : Vide"
Case com.sun.star.table.CellContentType.VALUE
      MsgBox "Contenu : Valeur"
Case com.sun.star.table.CellContentType.TEXT
      MsgBox "Contenu : Texte"
Case com.sun.star.table.CellContentType.FORMULA
      MsgBox "Contenu : Formule"
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** – pas de valeur
- **VALUE** – un nombre
- **TEXT** – une chaîne
- **FORMULA** – une 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 de deux lignes par deux colonnes dans la seconde 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 en dessous de 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)`** – le numéro de la feuille (la numérotation commençant à 0).
- **`StartColumn (long)`** – la première colonne de la plage de cellules (la numérotation commençant à 0).
- **`StartRow (long)`** – la première ligne de la plage de cellules (la numérotation commençant à 0).
- **`EndColumn (long)`** – la dernière colonne de la plage de cellules (la numérotation commençant à 0).
- **`EndRow (long)`** – la dernière ligne de la plage de cellules (la numérotation commençant à 0).

La structure `CellRangeAddress` complétée doit être passée comme premier paramètre de 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 l'emplacement d'insertion. L'énumération `CellInsertMode` reconnaît les valeurs suivantes :

- **NONE** – les valeurs actives restent à leur emplacement actuel.
- **DOWN** – les cellules situées à l'emplacement de l'insertion et en dessous sont déplacées vers le bas.
- **RIGHT** – les cellules situées à l'emplacement de l'insertion et à droite de celui-ci sont déplacées vers la droite.
- **ROWS** – les lignes suivant l'emplacement de l'insertion sont déplacées vers le bas.
- **COLUMNS** – les colonnes situées après l'emplacement 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 de l'énumération `com.sun.star.sheet.CellDeleteMode` :

- **NONE** – les valeurs actives restent à leur emplacement actuel.
- **UP** – les cellules situées à l'emplacement de l'insertion et en dessous sont déplacées vers le haut.
- **LEFT** – les cellules situées à l'emplacement de l'insertion et à droite de celui-ci sont déplacées vers la gauche.
- **ROWS** – les lignes suivant l'emplacement de l'insertion sont déplacées vers le haut.
- **COLUMNS** – les colonnes situées après l'emplacement d'insertion sont déplacées vers la gauche.

L'interface `XRangeMovement` comporte deux méthodes supplémentaires pour déplacer (`moveRange`) ou copier (`copyRange`) des plages de cellules. L'exemple suivant déplace la plage B2:C3 de manière qu'elle commence à l'emplacement 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)
```

En plus de la structure `CellRangeAdress`, la méthode `moveRange` attend une structure `com.sun.star.table.CellAddress` pour définir l'origine de la région de destination du déplacement. La méthode `CellAddress` fournit les valeurs suivantes :

- **`Sheet (short)`** – le numéro de la feuille de calcul (la numérotation commençant à 0).
- **`Column (long)`** – le numéro de la colonne concernée (la numérotation commençant à 0).
- **`Row (long)`** – le 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`, `removeRange` ne fournit pas de paramètre permettant des déplacements automatiques.

La méthode `copyRange` fonctionne de la même manière que `moveRange`, à ceci près que `copyRange` insère une copie de la plage de cellules au lieu de déplacer cette dernière.

En ce qui concerne leur fonctionnement, les méthodes `insertCell`, `removeRange` et `copyRange` de StarOffice Basic sont comparables aux méthodes `Range.Insert`, `Range.Delete` et `Range.Copy` de VBA. Alors que dans VBA, les méthodes sont appliquées à l'objet `Range` correspondant, dans StarOffice Basic, elles sont appliquées à l'objet `Sheet` associé.

Formatage

Un classeur fournit des propriétés et des méthodes pour formater les cellules et les pages.

Propriétés de cellules

De nombreuses options existent pour formater les cellules, telles que la spécification du type de police et de la taille du texte. Chaque cellule supporte les services `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`, dont les principales propriétés sont décrites au chapitre 6 (***Documents texte***). 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 citées à des cellules individuelles et à des plages de cellules.

L'objet `CellProperties` de l'API de StarOffice est comparable à l'objet `Interior` de VBA, qui définit également des propriétés spécifiques 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)`** - la couleur d'arrière-plan de la cellule de tableau.
- **`IsCellBackgroundTransparent (Boolean)`** - définit la couleur d'arrière-plan comme transparente.
- **`ShadowFormat (struct)`** – spécifie l'ombre des cellules (structure selon `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 dans la structure `com.sun.star.table.ShadowLocation`).
- **`ShadowWidth (Short)`** - taille de l'ombre en centièmes de millimètre.
- **`IsTransparent (Boolean)`** - définit l'ombre comme transparente.
- **`Color (Long)`** - la couleur de l'ombre.

L'exemple suivant inscrit le nombre 1 000 dans la cellule B2, met en rouge la couleur d'arrière-plan à l'aide de la propriété `CellBackColor`, puis crée une ombre gris clair pour la cellule, qu'il déplace 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 dans `com.sun.star.table.CellHoriJustify`)
- **`VertJustify (enum)`** - justification verticale du texte (valeur dans `com.sun.star.table.CellVertJustify`)
- **`Orientation (enum)`** - l'orientation du texte (valeur selon `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 degrés

L'exemple suivant montre 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 fournit toute une gamme de formats de date et d'heure prédéfinis. Chacun de ces formats comporte un numéro interne utilisé pour assigner le format à des cellules à l'aide de la propriété `NumberFormat`. StarOffice fournit les méthodes `queryKey` et `addNew` pour l'accès à des formats numériques existants et la création de vos propres formats numériques. Ces méthodes sont accessibles par l'appel d'objet suivant :

```
NumberFormats = Doc.NumberFormats
```

Un format est spécifié à l'aide d'une chaîne de format structurée d'une manière similaire à la fonction de format de StarOffice Basic. Il existe toutefois une différence importante : alors que la fonction Format requiert des abréviations, points décimaux et caractères séparateurs de milliers anglais, les abréviations spécifiques au pays doivent être utilisées dans la structure d'une commande format de l'objet NumberFormats.

L'exemple suivant formate la cellule B2 pour que les nombres soient affichées 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 Calc présente un aperçu des différentes options de formatage des cellules.

Propriétés de page

Les propriétés de page sont les options de formatage permettant de positionner le contenu d'un document sur la 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 définissant les formats de page diffère des autres formes de formatage. Alors que les formats des cellules, des paragraphes et des 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`.

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)`** – la couleur de l'arrière-plan
- **`BackGraphicURL (String)`** – l'URL de l'image d'arrière-plan à utiliser
- **`BackGraphicFilter (String)`** – le nom du filtre pour interpréter les images d'arrière-plan
- **`BackGraphicLocation (Enum)`** – la position de l'image d'arrière-plan (valeur selon l'énumération `com.sun.star.style.GraphicLocation`)
- **`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)`** – la largeur de page en centièmes de millimètre

- **Height (long)** – la hauteur de page en centièmes de millimètre
- **PrinterPaperTray (String)** – le nom du bac d'alimentation d'imprimante à utiliser

L'exemple suivant règle le format de page du style de page "Standard" sur DIN A5 en orientation paysage (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 le réglage des marges, des bordures et des ombres :

- **LeftMargin (long)** – la largeur de la marge gauche en centièmes de millimètre
- **RightMargin (long)** – la largeur de la marge droite en centièmes de millimètre
- **TopMargin (long)** – la largeur de la marge supérieure en centièmes de millimètre
- **BottomMargin (long)** – la largeur de la marge inférieure en centièmes de millimètre
- **LeftBorder (struct)** – spécifications du trait gauche de la bordure de page (structure `com.sun.star.table.BorderLine`)
- **RightBorder (struct)** – Spécifications du trait droit de la bordure de page (structure `com.sun.star.table.BorderLine`)
- **TopBorder (struct)** – Spécifications du trait supérieur de la bordure de page (structure `com.sun.star.table.BorderLine`)
- **BottomBorder (struct)** – Spécifications du trait inférieur de la bordure de page (structure `com.sun.star.table.BorderLine`)
- **LeftBorderDistance (long)** – la distance entre la bordure gauche et le contenu de la page en centièmes de millimètre
- **RightBorderDistance (long)** – la distance entre la bordure droite et le contenu de la page en centièmes de millimètre
- **TopBorderDistance (long)** – la distance entre la bordure supérieure et le contenu de la page en centièmes de millimètre

- **BottomBorderDistance (long)** – la distance entre la bordure inférieure et le contenu de la page en centièmes de millimètre
- **ShadowFormat (struct)** – spécifications pour l'ombre du contenu de la page (structure `com.sun.star.table.ShadowFormat`)

L'exemple suivant définit les bordures gauche et droite du style de page "Standard" 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 permettant de formater les en-têtes sont les suivantes :

- **HeaderIsOn (Boolean)** – l'en-tête est activé
- **HeaderLeftMargin (long)** – la distance entre l'en-tête et la marge gauche en centièmes de millimètre.
- **HeaderRightMargin (long)** – la distance entre l'en-tête et la marge droite en centièmes de millimètre.
- **HeaderBodyDistance (long)** – la distance entre l'en-tête et le corps du document en centièmes de millimètre.
- **HeaderHeight (long)** – la 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 autour de 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 du cadre entourant l'en-tête (structure `com.sun.star.table.BorderLine`)

- **HeaderBottomBorder (struct)** - détails de la bordure inférieure du cadre entourant l'en-tête (structure `com.sun.star.table.BorderLine`)
- **HeaderLeftBorderDistance (long)** – la distance entre la bordure gauche et le contenu de l'en-tête en centièmes de millimètre
- **HeaderRightBorderDistance (long)** – la distance entre la bordure droite et le contenu de l'en-tête en centièmes de millimètre
- **HeaderTopBorderDistance (long)** – la distance entre la bordure supérieure et le contenu de l'en-tête en centièmes de millimètre
- **HeaderBottomBorderDistance (long)** – la distance entre la bordure inférieure et le contenu de l'en-tête en centièmes de millimètre
- **HeaderIsShared (Boolean)** – les en-têtes des pages paires et impaires ont un contenu identique (voir `HeaderText`, `HeaderTextLeft` et `HeaderTextRight`)
- **HeaderBackColor (long)** – la couleur d'arrière-plan de l'en-tête
- **HeaderBackGraphicURL (String)** – l'URL de l'image d'arrière-plan à utiliser
- **HeaderBackGraphicFilter (String)** – le nom du filtre pour interpréter les images d'arrière-plan pour l'en-tête
- **HeaderBackGraphicLocation (Enum)** – la position de l'image d'arrière-plan pour l'en-tête (valeur selon l'énumération `com.sun.star.style.GraphicLocation`)
- **HeaderBackTransparent (Boolean) –** transparence de l'arrière-plan de l'en-tête
- **HeaderShadowFormat (struct)** – détails de l'ombre de l'en-tête (structure `com.sun.star.table.ShadowFormat`)

Les propriétés permettant de formater les pieds de page sont les suivantes :

- **FooterIsOn (Boolean)** – le pied de page est activé
- **FooterLeftMargin (long)** – la distance entre le pied de page et la marge gauche en centièmes de millimètre.
- **FooterRightMargin (long)** – la distance entre le pied de page et la marge droite en centièmes de millimètre.
- **FooterBodyDistance (long)** – la distance entre le pied de page et le corps du document en centièmes de millimètre.
- **FooterHeight (long)** – la 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 du cadre 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)** – la distance entre la bordure gauche et le contenu du pied de page en centièmes de millimètre
- **FooterRightBorderDistance (long)** – la distance entre la bordure droite et le contenu du pied de page en centièmes de millimètre
- **FooterTopBorderDistance (long)** – la distance entre la bordure supérieure et le contenu du pied de page en centièmes de millimètre
- **FooterBottomBorderDistance (long)** – la distance entre la bordure inférieure et le contenu du pied de page en centièmes de millimètre
- **FooterIsShared (Boolean)** – les pieds de page des pages paires et impaires ont un contenu identique (voir `FooterText`, `FooterTextLeft` et `FooterTextRight`).
- **FooterBackColor (long)** – la couleur d'arrière-plan du pied de page
- **FooterBackGraphicURL (String)** – l'URL de l'image d'arrière-plan à utiliser
- **FooterBackGraphicFilter (String)** – le nom du filtre pour interpréter les images d'arrière-plan pour le pied de page
- **FooterBackGraphicLocation (Enum)** – la position de l'image d'arrière-plan pour le pied de page (valeur selon l'énumération `com.sun.star.style.GraphicLocation`)
- **FooterBackTransparent (Boolean) –** rend l'arrière-plan du pied de page 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 par les propriétés suivantes :

- **LeftPageHeaderContent (Objet)** – contenu des en-têtes des pages paires (service `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageHeaderContent (Objet)** – contenu des en-têtes des pages impaires (service `com.sun.star.sheet.HeaderFooterContent`)
- **LeftPageFooterContent (Objet)** – contenu des pieds de pages des pages paires (service `com.sun.star.sheet.HeaderFooterContent`)
- **RightPageHeaderContent (Objet)** – contenu des pieds de pages des pages impaires (service `com.sun.star.sheet.HeaderFooterContent`)

Si vous n'avez pas besoin de faire la différence entre les pages paires et impaires pour les en-têtes ou les pieds de page (la propriété `FooterIsShared` vaut `False`), définissez les propriétés des en-têtes et des pieds de page sur les pages impaires.

Tous les objets cités renvoient un objet supportant 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 qui suit écrit la valeur "Ce n'est qu'un test." dans le champ texte de gauche de l'en-tête à partir du modèle "Par défaut".

```
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("Par défaut")

DefPage.HeaderIsOn = True
HContent = DefPage.RightPageHeaderContent
HText = HContent.LeftText
HText.String = "Ce n'est qu'un test."
DefPage.RightPageHeaderContent = HContent
```

Notez bien la dernière ligne de l'exemple : lorsque le texte a été modifié, l'objet `TextContent` doit être attribué une nouvelle fois 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 ils 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 avec le contenu de l'en-tête (service `com.sun.star.text.XText`)
- **`HeaderTextLeft (Object)`** – objet Text avec le contenu des en-têtes des pages de gauche (service `com.sun.star.text.XText`)
- **`HeaderTextRight (Object)`** – objet Text avec le contenu des en-têtes des pages de droite (service `com.sun.star.text.XText`)
- **`FooterText (Object)`** – objet Text avec le contenu des pieds de page (service `com.sun.star.text.XText`)
- **`FooterTextLeft (Object)`** – objet Text avec le contenu des pieds de page des pages de gauche (service `com.sun.star.text.XText`)
- **`FooterTextRight (Object)`** – objet Text avec le contenu des pieds de page des pages de droite (service `com.sun.star.text.XText`)

L'exemple suivant crée un en-tête dans le style de page "Standard" des documents texte et ajoute le texte "Ce n'est qu'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("Standard")

DefPage.HeaderIsOn = True
HText = DefPage.HeaderText

HText.String = "Ce n'est qu'un 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 souhaitées. 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 sont 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 lignes et de colonnes
- **`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 la gauche vers la 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, celle-ci 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 pour les cellules individuelles (le service `com.sun.star.table.Cell`), StarOffice propose également des objets représentant des plages de cellules. Les objets comme `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("Feuille 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 l'ensemble des cellules entre les lignes 1 à 15 dans les 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 du coin supérieur 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("Feuille 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, consultez la section ***Formatage***.

Calculs avec des places 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 pour décrire 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** - la somme de toutes les valeurs numériques
- **COUNT** - nombre total de valeurs (y compris les valeurs qui ne sont pas numériques)
- **COUNTNUMS** - nombre total de valeurs numériques
- **AVERAGE** - la moyenne de toutes les valeurs numériques
- **MAX** - la valeur numérique la plus élevée
- **MIN** - la valeur numérique la plus basse
- **PRODUCT** - le produit de toutes les valeurs numériques
- **STDEV** - l'écart type
- **VAR** - la variance
- **STDEVP** - l'écart type calculé sur la population totale
- **VARP** - la variance calculée sur 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("Feuille 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 de cellules ou de 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` viennent de la liste de constantes `com.sun.star.sheet.CellFlags`. Cette liste fournit les éléments suivants :

- **`VALUE`** – les valeurs numériques qui ne sont pas formatées sous forme de date ou d'heure
- **`DATETIME`** – les valeurs numériques formatées sous forme de date ou d'heure
- **`STRING`** - les chaînes
- **`ANNOTATION`** – les commentaires liés aux cellules
- **`FORMULA`** – les formules
- **`HARDATTR`** – le formatage direct des cellules
- **`STYLES`** – le formatage indirect
- **`OBJECTS`** – les objets de dessin connectés aux cellules
- **`EDITATTR`** – le formatage des caractères ne s'appliquant qu'à des parties des cellules

Vous pouvez également combiner les 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 indique un processus de recherche et 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 = "est"
ReplaceDescriptor.ReplaceString = "était"

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 aux différentes 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 des documents de 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 situation est rendue légèrement plus complexe par la présence de ***couches***. Par défaut, chaque document contient les couches ***Mise en page***, ***Contrôles*** et ***Lignes de cote***, tous les éléments de dessin étant ajoutés à la couche ***Mise en page***. Vous avez également la possibilité d'ajouter de nouvelles couches. Consultez le Developer's Guide de StarOffice pour plus d'informations sur les couches de dessin.

Pages

Les pages d'un document de dessin sont accessibles par 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 ***Page 1***, les exemples suivants sont 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("Page 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 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("Lien") Then
            oSheet = oSheets.getByName("Lien")
      Else
            oSheet = oDocument.createInstance("com.sun.star.sheet.Spreadsheet")
            oSheets.insertByName("Lien", oSheet)
            oSheet.IsVisible = False
      End If
```

L'appel ci-dessus renvoie un objet Page supportant le service `com.sun.star.drawing.DrawPage`. Ce service reconnaît les propriétés suivantes :

- **`BorderLeft (Long)`** – la bordure gauche en centièmes de millimètre
- **`BorderRight (Long)`** – la bordure droite en centièmes de millimètre
- **`BorderTop (Long)`** – la bordure supérieure en centièmes de millimètre
- **`BorderBottom (Long)`** – la bordure inférieure en centièmes de millimètre
- **`Width (Long)`** – la largeur de page en centièmes de millimètre
- **`Height (Long)`** – la hauteur de page en centièmes de millimètre
- **`Number (Short)`** – le nombre de pages (la numérotation commençant à 1), lecture seule
- **`Orientation (Enum)`** – l'orientation de la page (selon l'énumération `com.sun.star.view.PaperOrientation`)

Si ces paramètres sont modifiés, c'est ***l'ensemble*** des pages du document qui sont affectées.

L'exemple suivant définit la taille de page d'un document de dessin qui vient d'être ouvert à 20 × 20 centimètres, avec une marge de 0,5 centimètres :

```
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 supportent 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 de telles propriétés comme le formatage ou d'appliquer des remplissages. Les options de formatage disponibles dépendent du type d'objet de dessin.

L'exemple suivant crée et insère un rectangle dans un document de 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 par l'appel `drawPages(0)`.

Les structures `Point` et `Size` sont ensuite initialisées avec le point d'origine (angle supérieur gauche) et la taille de l'objet de dessin. Les longueurs sont spécifiées en centièmes de millimètre.

Le code utilise ensuite l'appel `Doc.createInstance` pour créer l'objet de dessin rectangulaire comme spécifié par le service `com.sun.star.drawing.RectangleShape`. Enfin, l'objet de dessin est assigné à une page avec un appel `Page.add`.

Propriétés Fill

Cette section décrit quatre services, et pour chacun, le code d'exemple utilise un élément de forme rectangulaire combiné à différents types de formatage. Les propriétés de remplissage sont combinées dans le service `com.sun.star.drawing.FillProperties`.

StarOffice reconnaît quatre types de formatage principaux pour une zone à remplir. 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. La quatrième variante consiste à projeter des images existantes dans la zone à remplir.

Le mode de 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)`** – la couleur de remplissage de la zone.

Pour utiliser ce mode de 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 pouvez appliquer un dégradé de couleurs à n'importe quelle zone à remplir d'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` à assigner à la propriété `FillGradient`. Cette propriété propose les options suivantes :

- **`Style (Enum)`** - le type du dégradé, par exemple, linéaire ou radial (valeurs par défaut selon `com.sun.star.awt.GradientStyle`)
- **StartColor (Long)** - la couleur initiale du dégradé de couleurs
- **StartColor (Long)** - la couleur finale du dégradé de couleurs
- **Angle (Short)** - l'angle du dégradé de couleurs en dixièmes de degré
- **XOffset (Short)** - la coordonnée en X à laquelle le dégradé de couleur commence, en centièmes de millimètre
- **XOffset (Short)** - la coordonnée en Y à laquelle le dégradé de couleur commence, en centièmes de millimètre
- **StartIntensity (Short)** - l'intensité de `StartColor` en pourcentage (dans StarOffice Basic, vous pouvez également spécifier des valeurs supérieures à 100 pour cent)

- **EndIntensity (Short)** - l'intensité de `EndColor` en pourcentage (dans StarOffice Basic, vous pouvez également spécifier des valeurs supérieures à 100 pour cent)
- **StepCount (Short)** - le nombre de couleurs intermédiaires utilisées par StarOffice pour calculer les dégradés

L'exemple suivant montre l'utilisation de dégradés 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 initiale et finale est de 150 pour cent (`StartIntensity` et `EndIntensity`) ce qui donne des couleurs apparemment plus brillantes que celles 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 donner à la propriété `FillStyle` la valeur `HATCH`. Le code pour définir des hachures est très similaire à celui des dégradés de couleurs. Là

aussi, une structure auxiliaire, `com.sun.star.drawing.Hatch` en l'occurrence, sert à définir l'apparence des hachures. La structure pour les hachures a les propriétés suivantes :

- **`Style (Enum)`** - le type de hachures : simples, carrées ou carrées avec diagonales (valeurs par défaut selon `com.sun.star.awt.HatchStyle`)
- **`Color (Long)`** - la couleur des lignes
- **`Distance (Long)`** - la distance entre les lignes en centièmes de millimètre
- **`Angle (Short)`** - l'angle des hachures en dixièmes de degré

L'exemple suivant montre l'utilisation d'une structure pour 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`) dont les lignes sont pivotées de 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` à `BITMAP`. Si le bitmap est déjà disponible dans StarOffice, il vous suffit de spécifier son nom dans la propriété `FillBitMapName` style d'affichage (simple, carrelage ou étiré) dans la propriété `FillBitmapMode` (valeurs par défaut selon `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, dont il remplit la surface en répétant le bitmap Sky, disponible dans StarOffice.

```
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 également la possibilité d'ajuster 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 pour cent.

```
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` à 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 Line

Tous les objets de dessin susceptibles d'avoir une ligne de bordure supportent le service `com.sun.star.drawing.LineStyle`. Ce service propose notamment les propriétés suivantes :

- **`LineStyle (Enum)`** - le type de ligne (valeurs par défaut selon `com.sun.star.drawing.LineStyle`)
- **`LineColor (Long)`** - la couleur de la ligne
- **`LineTransparence (Short)`** - la transparence de la ligne
- **`LineWidth (Long)`** - l'épaisseur de la ligne en centièmes de millimètre
- **`LineJoint (Enum)`** - les transitions entre les points de connexion (valeurs par défaut selon `com.sun.star.drawing.LineJoint`)

L'exemple suivant crée un rectangle avec une bordure continue (`LineStyle` = `SOLID`) de 5 millimètres d'épaisseur (`LineWidth`) et une transparence de 50 pour cent. Les bords droit et gauche de la ligne se prolongent jusqu'à ce qu'ils se rencontrent (`LineJoint` = `MITER`) de manière à 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 pour dessiner des lignes pointillées. Pour plus d'informations, consultez la référence de l'API de StarOffice.

Propriétés Text (objets de dessin)

Les services `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties` peuvent formater du texte en objets de dessin. Ces services portent sur les caractères individuels comme sur les paragraphes et sont traités en détail dans le chapitre 6 (***Documents texte***).

L'exemple suivant insère du texte dans un rectangle et formate sa 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 = "Ceci est un 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 l'ajout de l'objet de dessin 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 à la hauteur de l'élément qu'il contient
- **`TextAutoGrowWidth (Boolean)`** - adapte la largeur de l'élément de dessin au texte qu'il contient
- **`TextHorizontalAdjust (Enum)`** - la position horizontale du texte à l'intérieur de l'élément de dessin (valeurs par défaut selon `com.sun.star.drawing.TextHorizontalAdjust`)
- **`TextVerticalAdjust (Enum)`** - la position verticale du texte à l'intérieur de l'élément de dessin (valeurs par défaut selon `com.sun.star.drawing.TextVerticalAdjust`)
- **`TextLeftDistance (Long)`** - écart à gauche entre l'élément de dessin et le texte, en centièmes de millimètre
- **`TextRightDistance (Long)`** - écart à droite entre l'élément de dessin et le texte en centièmes de millimètre
- **`TextUpperDistance (Long)`** - écart entre le haut du texte et l'élément de dessin en centièmes de millimètre

- **`TextLowerDistance (Long)`** - écart entre le bas du texte et l'élément de dessin en centièmes de millimètre

L'exemple suivant montre l'utilisation des propriétés citées.

```
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 = "Ceci est un 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 un texte dans le coin supérieur gauche de celui-ci grâce aux propriétés `TextVerticalAdjust` et `TextHorizontalAdjust`. L'écart minimum entre le texte et le bord de l'objet est défini à trois millimètres.

Propriétés Shadow

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)`** - la couleur de l'ombre
- **`ShadowTransparence (Short)`** - la transparence de l'ombre
- **`ShadowXDistance (Long)`** - l'écart vertical entre l'ombre et l'objet de dessin en centièmes de millimètres

- **ShadowYDistance (Long)** - l'écart horizontal entre l'ombre et l'objet de dessin en centièmes de millimètres

L'exemple suivant crée un rectangle avec une ombre décalée par rapport à celui-ci de 2 millimètres verticalement et horizontalement. L'ombre est rendue en gris foncé avec une transparence de 50 pour cent.

```
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`) supportent les services suivants pour leur formatage :

- **Propriétés Fill** – `com.sun.star.drawing.FillProperties`
- **Propriétés Line** – `com.sun.star.drawing.LineProperties`
- **Propriétés Text** – `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`)
- **Propriétés Shadow** – `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` est chargé des cercles et des ellipses, et prend en charge les services suivants :

- **Propriétés Fill** – `com.sun.star.drawing.FillProperties`
- **Propriétés Line** – `com.sun.star.drawing.LineProperties`
- **Propriétés Text** – `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`)
- **Propriétés Shadow** – `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 (les valeurs par défaut correspondent à `com.sun.star.drawing.CircleKind`)
- **`CircleStartAngle (Long)`** - angle de départ en dixièmes de degré (pour les segments de cercle ou d'ellipse seulement)
- **`CircleEndAngle (Long)`** - angle final en dixièmes de degré (pour les segments de cercle ou d'ellipse seulement)

La propriété `CircleKind` détermine si un objet est un cercle complet, une tranche circulaire ou une section d'un cercle. Les valeurs suivantes sont disponibles :

- **`com.sun.star.drawing.CircleKind.FULL`** – cercle entier ou ellipse entière
- **`com.sun.star.drawing.CircleKind.CUT`** 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 circulaire
- **`com.sun.star.drawing.CircleKind.ARC`** – angle (sans la ligne du cercle)

L'exemple suivant crée une tranche circulaire avec un angle de 70 degrés (produit par la différence entre l'angle de départ de 20 degrés et l'angle final de 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 offre le service `com.sun.star.drawing.LineShape` pour les objets Line. Les objets Line supportent tous les services de formatage généraux, à l'exception des zones. Vous trouverez ci-après toutes les propriétés associées au service `LineShape` :

- **Propriétés Line** – `com.sun.star.drawing.LineProperties`
- **Propriétés Text** – `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`)
- **Propriétés Shadow** – `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` spécifient le point d'arrivée 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 polygonales

StarOffice supporte également les formes polygonales complexes par l'intermédiaire du service `com.sun.star.drawing.PolyPolygonShape`. À proprement parler, un ***polypolygone*** n'est pas un simple polygone mais un polygone multiple. Par conséquent, il est 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 Fill** – `com.sun.star.drawing.FillProperties`
- **Propriétés Line** – `com.sun.star.drawing.LineProperties`
- **Propriétés Text** – `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`)
- **Propriétés Shadow** – `com.sun.star.drawing.ShadowProperties`

Le service `PolyPolygonShape` offre également une propriété vous permettant de définir les coordonnées d'un polygone :

- `PolyPolygon (Array)` – champ contenant les coordonnées du polygone (matrice de type double avec 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. Vous devez plutôt créer une matrice de points, l'intégrer à une seconde matrice (à l'aide de l'appel `Array(Coordinates() )` et assigner cette matrice au

polygone. Avant que l'appel correspondant puisse s'effectuer, le polygone doit être inséré dans le document.

La matrice double dans la définition permet de créer des formes complexes en fusionnant plusieurs polygones. Par exemple, vous pouvez créer un rectangle et y insérer un autre rectangle pour 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())
```

Par rapport aux zones pleines et aux zones qui sont des trous, StarOffice applique une règle simple : le bord de la forme extérieure est toujours la bordure extérieure du polypolygone. La ligne

suivante vers l'intérieur est 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 pleine.

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 n'importe quelle image de StarOffice dont l'apparence peut être adaptée au moyen d'un ensemble complet de propriétés.

Les objets graphiques supportent deux des propriétés de formatage générales :

- **Propriétés Text** – `com.sun.star.drawing.Text` (avec `com.sun.star.style.CharacterProperties` et `com.sun.star.style.ParagraphProperties`)
- **Propriétés Shadow** – `com.sun.star.drawing.ShadowProperties`

Les autres propriétés supportées par les objets graphiques sont les suivantes :

- **`GraphicURL (String)`** - URL de l'image
- **`AdjustLuminance (Short)`** - luminosité des couleurs, comme pourcentage (les valeurs négatives sont aussi autorisées)
- **`AdjustContrast (Short)`** - contraste comme pourcentage (les valeurs négatives sont aussi autorisées)
- **`AdjustRed (Short)`** - valeur de rouge comme pourcentage (les valeurs négatives sont aussi autorisées)
- **`AdjustGreen (Short)`** - valeur de vert comme pourcentage (les valeurs négatives sont aussi autorisées)
- **`AdjustBlue (Short)`** - valeur de bleu comme pourcentage (les valeurs négatives sont aussi autorisées)
- **`Gamma (Short)`** - valeur gamma d'une image
- **`Transparency (Short)`** - transparence d'une image comme pourcentage
- **`GraphicColorMode (enum)`** - mode de couleur, par exemple, standard, niveaux de gris, noir et blanc (valeur par défaut correspondant à `com.sun.star.drawing.ColorMode`)

L'exemple suivant montre comment insérer une page dans un objet graphique.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 sont liées
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 = "fichier:///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 insère l'image `test.jpg` et adapte son apparence en utilisant les propriétés `Adjust`. Dans cet exemple, les images sont représentées avec 40 pour cent de transparence et aucune autre conversion de couleur (`GraphicColorMode` = `STANDARD`).

Édition des objets de dessin

Regroupement des objets

Dans de nombreuses situations, il est utile de regrouper plusieurs objets de dessin individuels afin qu'ils se comportent comme un seul objet.

L'exemple suivant regroupe 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
' ' crée un élément de dessin carré
Square = Doc.createInstance("com.sun.star.drawing.RectangleShape")
Square.Size = Size
Square.Position = Point
Square.FillColor = RGB(255,128,128)
Page.add(Square)
' ' crée un élément de dessin cercle
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)
' ' regroupe les éléments de dessin carré et cercle
Shapes = createUnoService("com.sun.star.drawing.ShapeCollection")
Shapes.add(Square)
Shapes.add(Circle)
Group = Page.group(Shapes)
' ' centre les éléments de dessin regroupés
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 et les insère dans une page. Il crée ensuite un objet supportant le service `com.sun.star.drawing.ShapeCollection` et utilise la méthode `Add` pour ajouter le rectangle et le cercle à cet objet. Le service `ShapeCollection` est ajouté à la page en utilisant la méthode `Group` et retourne l'objet `Group` réel qui peut être édité comme une forme `Shape` individuelle.

Si vous souhaitez formater les différents objets d'un groupe, appliquez le formatage avant de les ajouter au groupe. Vous ne pouvez pas modifier les objets une fois qu'ils constituent un groupe.

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)
```

Le prochain exemple crée le même rectangle que dans le précédent, mais le cisaille 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 dans 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, décrite au chapitre 6, ***Documents texte***. Toutefois, dans les documents de dessin, les objets Descriptor de recherche et de remplacement ne sont pas créés directement par l'intermédiaire 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` et applique ensuite ce descripteur dans une boucle, à toutes les pages du document de dessin.

Présentations

Les présentations StarOffice sont basées sur des documents de dessin. 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, par la liste `DrawPages` de l'objet Document. Le service `com.sun.star.presentation.PresentationDocument`, chargé des documents de présentation, fournit également le service `com.sun.star.drawing.DrawingDocument` complet.

Utilisation des présentations

Outre les fonctions de dessin offertes par la propriété `Presentation`, le document de présentation contient un objet Presentation donnant accès aux propriétés principales et aux mécanismes de contrôle des présentations. Par exemple, cet objet offre une méthode `start` pouvant démarrer 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 au document de présentation actif et établit l'objet Presentation associé. La méthode `start()` de l'objet permet de démarrer l'exemple et d'exécuter la présentation à l'écran.

Les méthodes suivantes sont fournies comme objets Presentation :

- **`start`** - démarre la présentation
- **`end`** - met fin à la présentation
- **`rehearseTimings`**rehearseTimings - démarre la présentation depuis le début et établit son exécution

Les propriétés suivantes sont également disponibles :

- **`AllowAnimations (Boolean)`** – exécute les animations de la présentation
- **`CustomShow (String)`** – vous permet de spécifier le nom de la présentation afin de pouvoir y faire référence dans la présentation
- **`FirstPage (String)`** – nom de la diapo par laquelle la présentation doit commencer
- **`IsAlwaysOnTop (Boolean)`** – affiche toujours la fenêtre de la présentation au premier plan
- **`IsAutomatic (Boolean)`** – exécute automatiquement toute la présentation
- **`IsEndless (Boolean)`** – recommence la présentation depuis le début une fois terminée
- **`IsFullScreen (Boolean)`** – démarre 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 démarrage de la présentation
- **`UsePn (Boolean)`** – affiche le pointeur pendant la présentation

CHAPITRE 9

Diagrammes

StarOffice peut afficher les données dans un diagramme qui crée des liaisons graphiques entre les données sous la forme de barres, de secteurs, de lignes ou d'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 façon semblable au processus utilisé 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 visualisée immédiatement dans le diagramme associé. 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 feuilles de calcul

Les diagrammes ne sont pas traités comme des documents indépendants dans StarOffice, mais comme des objets incorporés dans 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("MonDiagramme", Rect, RangeAddress(), True, True)
```

Bien que le code utilisé dans l'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 un autre type de graphique, vous devez remplacer le diagramme à barres de façon explicite :

```
Chart = Charts.getByName("MonDiagramme").embeddedObject
Chart.Diagram = Chart.createInstance("com.sun.star.chart.LineDiagram")
```

La première ligne définit l'objet Chart correspondant. La seconde ligne remplace le diagramme actif par un nouveau – dans cet exemple, un diagramme linéaire.

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 feuille de calcul. 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 feuille de calcul. L'accès aux diagrammes s'effectue toujours en utilisant la liste `Charts` de l'objet `Sheet` associé.

La structure des diagrammes

La structure d'un diagramme – et donc la liste des services et interfaces qu'il prend en charge – dépend de son type. Par exemple, les méthodes et propriétés de l'axe Z sont disponibles uniquement dans les diagrammes 3D et non dans les diagrammes 2D. Dans les diagrammes à secteurs, il n'existe aucune interface pour utiliser des axes.

Les 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 offrent leurs propres objets pour chacun de ces éléments. L'objet `Chart` offre les propriétés suivantes pour administrer ces éléments :

- **`HasMainTitle (Boolean)`** – active le titre.

- **`Title (Object)`** – objet contenant des informations détaillées concernant le titre du diagramme (supporte le service `com.sun.star.chart.ChartTitle`).
- **`HasSubTitle(Boolean)`** – active le sous-titre.
- **`Subtitle (Object)`** – objet contenant des informations détaillées concernant le sous-titre du diagramme (supporte le service `com.sun.star.chart.ChartTitle`).
- **`HasLegend (Boolean)`** – active la légende.
- **Legend (Object)** – objet contenant des informations détaillées concernant la légende du diagramme (supporte le service `com.sun.star.chart.ChartLegendPosition`).

À de nombreux égards, les éléments spécifiés correspondent à un élément de dessin. Ceci 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 du programme technique des éléments de dessin.

De ce fait, les utilisateurs peuvent déterminer la position et la taille de l'élément à l'aide des propriétés `Size` et `Position`.

Les autres propriétés Fill et Line (services `com.sun.star.drawing.FillProperties` et `com.sun.star.drawing.LineStyle`) ainsi que les propriétés character (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 à laquelle est placée 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 a une couleur d'arrière-plan 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("MonDiagramme", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MonDiagramme").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 a un objet, accessible en utilisant les propriétés suivantes de l'objet Diagram :

- **`Area (Object)`** – zone d'arrière-plan du diagramme (supporte le service `com.sun.star.chart.ChartArea`).

L'arrière-plan d'un diagramme recouvre sa zone complète, y compris la zone sous le titre, le sous-titre et la légende du diagramme. Le service `com.sun.star.chart.ChartArea` associé prend en charge les propriétés Line et Fill et n'offre aucune autre propriété extensive.

Paroi et plancher 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 zone de données.

Il existe généralement deux parois dans les diagrammes 3D : l'une derrière la zone de données et l'autre servant 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, supporte le service `com.sun.star.chart.ChartArea`).
- **`Wall (Object)`** – parois du diagramme (uniquement pour les diagrammes 3D, supporte le service `com.sun.star.chart.ChartArea`).

Les objets spécifiés supportent le service `com.sun.star.chart.ChartArea` qui offre à son tour les propriétés Fill et Line (services `com.sun.star.drawing.FillProperties` et `com.sun.star.drawing.LineStyle`, 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 à son tour partie de l'objet `Chart` :

```
Chart.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("MonDiagramme", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MonDiagramme").EmbeddedObject

Chart.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, pour chacun des axes X, Y et Z primaires, il peut y avoir également un titre, une description, une grille et une grille auxiliaire. Il existe une option permettant d'afficher et de masquer tous ces éléments. L'objet Diagram offre les propriétés d'administration suivantes de ces fonctions (en prenant l'exemple d'un axe X ; les propriétés des axes Y et Z sont structurées de la même manière) :

- **`HasXAxis (Boolean)`** – active l'axe X.
- **`XAxis (Object)`** – objet contenant des informations détaillées concernant l'axe X (supporte 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 concernant la grille principale de l'axe X (supporte le service `com.sun.star.chart.ChartGrid`).
- **`HasXAxisHelpGrid (Boolean)`** – active la grille secondaire de l'axe X.
- **`XHelpGrid (Object)`** – objet contenant des informations détaillées concernant la grille secondaire de l'axe X (supporte le service `com.sun.star.chart.ChartGrid`).
- **`HasXAxisTitle (Boolean)`** – active le titre de l'axe X.
- **`XAxisTitle (Object)`** – objet contenant des informations détaillées concernant le titre de l'axe X (supporte 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 X secondaire) :

- **`HasSecondaryXAxis (Boolean)`** – active l'axe X secondaire.
- **`SecondaryXAxis (Object)`** – objet contenant des informations détaillées concernant l'axe X secondaire (supporte le service `com.sun.star.chart.ChartAxis`).
- **`HasSecondaryXAxisDescription (Boolean)`** – active la description de l'axe X.

Propriétés des axes

Les objets Axis d'un diagramme StarOffice supportent le service `com.sun.star.chart.ChartAxis`. Outre les propriétés des caractères (service `com.sun.star.style.CharacterProperties`, reportez-vous au chapitre 6) et des lignes (service `com.sun.star.drawing.LineStyle`, reportez-vous au chapitre 8), il offre les propriétés suivantes :

- **`Max (Double)`** - valeur maximum de l'axe.
- **`Min (Double)`** - valeur minimum 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 maximum de l'axe.
- **`AutoMin (Boolean)`** - détermine automatiquement la valeur minimum de l'axe.
- **`AutoOrigin (Boolean)`** - détermine automatiquement le point d'intersection des axes.
- **`AutoStepMain (Boolean)`** - détermine automatiquement la distance entre deux lignes principales de l'axe.
- **`AutoStepHelp (Boolean)`** - détermine automatiquement la distance entre deux lignes secondaires de l'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 en centièmes de degré.
- **`Marks (Const)`** - constante qui spécifie 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 spécifie si les lignes secondaires de l'axe doivent se trouver à l'intérieur et/ou à l'extérieur de la zone du diagramme (les valeurs par défaut correspondent à `com.sun.star.chart.ChartAxisMarks`)
- **`Overlap (Long)`** - pourcentage qui spécifie la marge de superposition des barres des différents jeux de données (à 100%, les barres s'affichent en superposition complète, à -100%, une distance de la largeur d'une barre les sépare).
- **`GapWidth (long)`** - pourcentage qui spécifie 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 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)`** - permet les retours à la ligne.
- **`TextCanOverlap (Boolean)`** - permet les chevauchements de texte.
- **`NumberFormat (Long)`** - format numérique (voir chapitre 7, section ***Formats de nombre, de date et de texte***)

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 à son tour les propriétés Line du service de support `com.sun.star.drawing.LineStyle` (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 de la paroi arrière du diagramme est définie sur blanc. Les axes X et Y ont une grille auxiliaire grise pour l'orientation visuelle. La valeur minimale de l'axe Y est fixée à 0 et la valeur maximale est fixée à 100 de manière à ce que la résolution du diagramme soit conservée même si les valeurs changent.

```
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("MonDiagramme", Rect, RangeAddress(), True, True)

Chart = Charts.getByName("MonDiagramme").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 graphiques 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 3D.

Diagrammes empilés

Les diagrammes empilés sont des diagrammes disposés 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 à son tour fournit les propriétés suivantes :

- **`Stacked (Boolean)`** – active l'affichage en mode empilé.
- **`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 de graphique 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 en fonction de `com.sun.star.chart.ChartSymbolType`).
- **`SymbolSize (Long)`** - taille du symbole d'affichage des points de données en centièmes de millimètre.
- **`SymbolBitmapURL (String)`** - nom de fichier des images d'affichage des points de données.
- **`Lines (Boolean)`** - relie les points de données par des 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 de graphique 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 de graphique 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 au lieu d'horizontalement.
- **`Deep (Boolean)`** - en mode d'affichage 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 au moyen de lignes (uniquement avec 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 être empilés. Ils peuvent s'afficher sous forme de graphique 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). L'objectif du développement de cette interface était d'offrir l'accès au plus grand nombre possible de sources de données différentes.

Pour rendre cela possible, l'accès aux sources de données s'effectue par 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 Guide du développeur de StarOffice contient de plus amples informations à ce sujet.

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.

L'interface de base de données de StarOffice s'est développée lors du lancement de StarOffice 6.0. Bien que par le passé, l'accès aux bases de données se soit principalement effectué en utilisant un ensemble de méthodes de l'objet `Application`, l'interface de StarOffice 6.0 se subdivise en plusieurs objets. Un service `DatabaseContext` sert d'objet racine pour les fonctions de la base de données.

SQL : un langage de requête

Le langage SQL est fourni comme langage de requête aux utilisateurs de SDBC. Pour comparer les différences entre les différents 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 permet 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.

L'implémentation SQL à partir de SDBC est orientée vers la norme SQL-ANSI. Les extensions spécifiques Microsoft, telles que la construction `INNER JOIN` ne sont pas supportées. Elles doivent être remplacées par des commandes standard (`INNER JOIN`, par exemple devra être remplacée par une clause `WHERE` correspondante).

Types d'accès aux bases de données

L'interface de base de données de StarOffice est disponible avec 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 avec l'assistance de sources de données SDBC et peuvent ensuite être 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 déplace 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'effectuer l'importation d'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 vers 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, mais s'intéresse principalement à 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 de base 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**. Mais vous pouvez également créer des sources de données et 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. Ceci 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 le contexte de la base de données peut être créé, puis utilisé pour déterminer les noms de toutes les sources de données disponibles. Il affiche les noms dans un 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 lesquelles, 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, sous la forme ***jdbc: subprotocol : subname*** ou ***sdbc: subprotocol : subname***.
- **`Info (Array)`** – array contenant des paires `PropertyValue` avec des paramètres de connexion (généralement un nom d'utilisateur et un mot de passe).
- **`User (String)`** – nom de l'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 disponibles pour la base de données (supporte l'interface `com.sun.star.util.XNumberFormatsSupplier`, voir chapitre 7, section ***Formats de nombre, de date et de texte***).
- **`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.

Les sources de données de StarOffice ne sont pas comparables une à une avec les sources de données ODBC. Si une source de données ODBC ne recouvre que les informations concernant l'origine des données, 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 permettent de simplifier l'utilisation des bases de données car elles peuvent s'ouvrir d'un simple clic de souris et permettent en outre aux utilisateurs sans aucune connaissance SQL d'émettre des commandes SQL.

Un objet prenant en charge le service `com.sun.star.sdb.QueryDefinition` est masqué 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 pouvant être établis 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 un ensemble complet d'autres propriétés. Il s'agit des propriétés :

- **`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 données, StarOffice offre une option de liaison des sources de données aux formulaires de base de données. Ces liaisons sont disponibles par l'intermédiaire de la méthode `getBookmarks()`. Ceci 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 d'`Index`.

L'exemple 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 aux bases de données

Une connexion à la base de données est nécessaire pour y accéder. Il s'agit d'une voie de transfert qui permet 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 redémarrage du programme.

StarOffice offre différentes manières d'établir des connexions à la base de données. Voici une explication de la méthode basée sur une source de données existante.

```
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 `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

L'accès à une table s'effectue généralement dans StarOffice par l'intermédiaire de l'objet `ResultSet`. Un `ResultSet` est un type de marqueur indiquant un ensemble de données courant au sein d'un volume de résultats obtenus à l'aide de la commande `SELECT`.

L'exemple montre l'utilisation d'un `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
```

Lorsque la connexion à la base de données a été établie, le code utilisé dans l'exemple utilise d'abord l'appel `Connection.createObject` pour créer un objet `Statement`. Cet objet `Statement` utilise ensuite l'appel `executeQuery` pour retourner le `ResultSet` réel. Le programme vérifie à présent si le `ResultSet` existe réellement et parcourt les enregistrements de données à l'aide d'une boucle. Les valeurs requises (dans notre exemple, celles du champ `CustomerNumber`) retournent le `ResultSet` en utilisant la méthode `getString`, selon laquelle le paramètre 1 détermine que l'appel se relie aux valeurs de la première colonne.

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.

L'accès à la base de données s'effectue en réalité dans StarOffice 7 par l'intermédiaire d'un objet `ResultSet`. Ceci représente le contenu d'une table ou le résultat d'une commande SQL-SELECT. Dans le passé, l'objet `ResultSet` fournissait les méthodes résidentes dans l'objet `Application` pour la navigation au sein des données (par exemple `DataNextRecord`).

Méthodes de récupération des valeurs en fonction du type

Comme le montre l'exemple de la section précédente, StarOffice offre une méthode `getString` d'accès au contenu de la table. La méthode fournit le résultat sous la forme d'une chaîne. Les méthodes `get` suivantes sont disponibles :

- `getByte()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getShort()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getInt()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getLong()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getFloat()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getDouble()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getBoolean()` – prend en charge les types de données SQL pour les nombres, les caractères et les chaînes.
- `getString()` – prend en charge tous les types de données SQL.
- `getBytes()` – prend en charge les types de données SQL pour les valeurs binaires.
- `getDate()` – prend en charge les types de données SQL pour les nombres, les chaînes, la date et l'heure.
- `getTime()` – prend en charge les types de données SQL pour les nombres, les chaînes, la date et l'heure.
- `getTimestamp()` – prend en charge les types de données SQL pour les nombres, les chaînes, la date et l'heure.
- `getCharacterStream()` – prend en charge les types de données SQL pour les nombres, les chaînes et les valeurs binaires.
- `getUnicodeStream()` – prend en charge les types de données SQL pour les nombres, les chaînes et les 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.

Les variantes ResultSet

L'accès aux bases de données est souvent une question de vitesse critique. C'est la raison pour laquelle StarOffice offre plusieurs moyens pour optimiser les `ResultSets` et ainsi contrôler la vitesse d'accès. Plus un `ResultSet` offre de fonctions, plus son implémentation est généralement complexe et plus lentes sont les fonctions.

Un `ResultSet` simple, tel que celui présenté à la section "Itération de tables", offre l'étendue de fonctions la plus réduite possible. 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 possibilité de modifier des valeurs, ne sont pas incluses.

L'objet `Statement` utilisé pour créer le `ResultSet` offre certaines propriétés qui permettent d'influer sur les fonctions du `ResultSet` :

- **`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 `ResultSets` (spécifications correspondant à `com.sun.star.sdbc.ResultSetType`).

Les valeurs définies dans `com.sun.star.sdbc.ResultSetConcurrency` sont les suivantes :

- **`UPDATABLE`** - `ResultSet` permet la modification des valeurs.
- **`READ_ONLY`** – `ResultSet` ne permet pas les modifications.

Le groupe de constantes `com.sun.star.sdbc.ResultSetConcurrency` fournit les spécifications suivantes :

- **`FORWARD_ONLY`** – `ResultSet` permet uniquement la navigation vers l'avant.
- **`SCROLL_INSENSITIVE`** – `ResultSet` permet tout type de navigation ; les changements des données d'origine ne sont toutefois pas notés.
- **`SCROLL_SENSITIVE`** – `ResultSet` permet tout type de navigation ; les changements des données d'origine modifient le `ResultSet`.

> Un `ResultSet` contenant des propriétés `READ_ONLY` et `SCROLL_INSENSITIVE` correspond à un jeu d'enregistrements du type `Snapshot` dans ADO et DAO.
>
> Lorsque vous utilisez les propriétés du `ResultSet` `UPDATEABLE` et `SCROLL_SENSITIVE` , l'étendue de fonctions d'un `ResultSet` est comparable à un `Recordset` de type `Dynaset` à partir d'ADO et de DAO.

Les méthodes de navigation dans les ResultSets

Si un `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()`** – navigation vers l'enregistrement de données suivant.

- **previous()** – navigation vers l'enregistrement de données précédent.
- **first()** – navigation vers le premier enregistrement de données.
- **last()** – navigation vers le dernier enregistrement de données.
- **beforeFirst()** – navigation avant le premier enregistrement de données.
- **afterLast()** – navigation après le dernier enregistrement de données.

Toutes les méthodes retournent un paramètre 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 Boolean :

- **isBeforeFirst()** – `ResultSet` est placé avant le premier enregistrement de données.
- **isAfterLast()** – `ResultSet` est placé après le dernier enregistrement de données.
- **isFirst()** – `ResultSet` est le premier enregistrement de données.
- **isLast()** – `ResultSet` est le dernier enregistrement de données.

Modification des enregistrements de données

Si un `ResultSet` a été créé avec la valeur `ResultSetConcurrency = UPDATEABLE`, son contenu est modifiable. Ceci ne s'applique que pendant le temps où 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, qui sont structurées de la même manière que les méthodes `get` pour 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 `updateRow()`.

CHAPITRE 11

Boîtes de dialogue

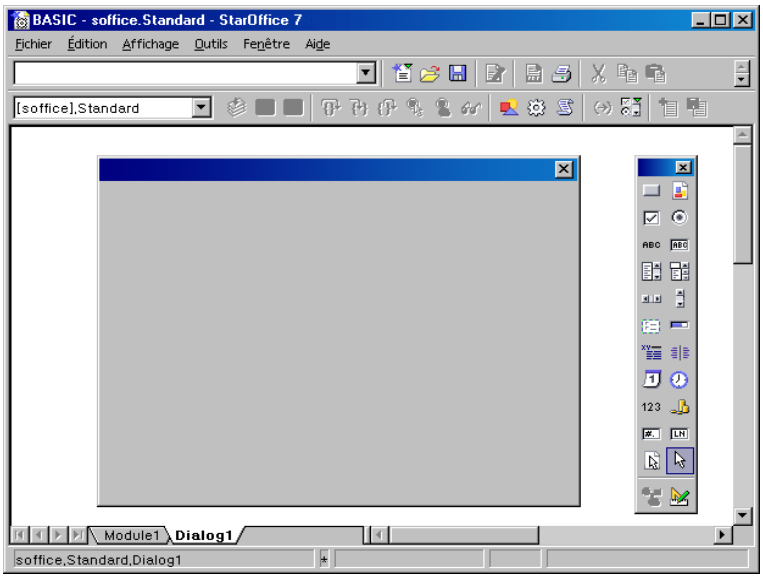
Vous pouvez ajouter des fenêtres de boîte de dialogue et des formulaires personnalisés aux documents StarOffice. Ceux-ci peuvent, à leur tour, être liés à des macros StarOffice Basic afin d'en étendre la plage d'utilisation de façon considérable. Les boîtes de dialogue peuvent, par exemple, afficher des informations d'une base de données ou guider les utilisateurs dans un processus étape par étape de création d'un nouveau document sous la forme d'un AutoPilote.

Utilisation des boîtes de dialogue

Les boîtes de dialogue de StarOffice Basic sont constituées d'une fenêtre de boîte de dialogue contenant des champs de texte, des zones de liste, des boutons radio, et d'autres éléments de contrôle.

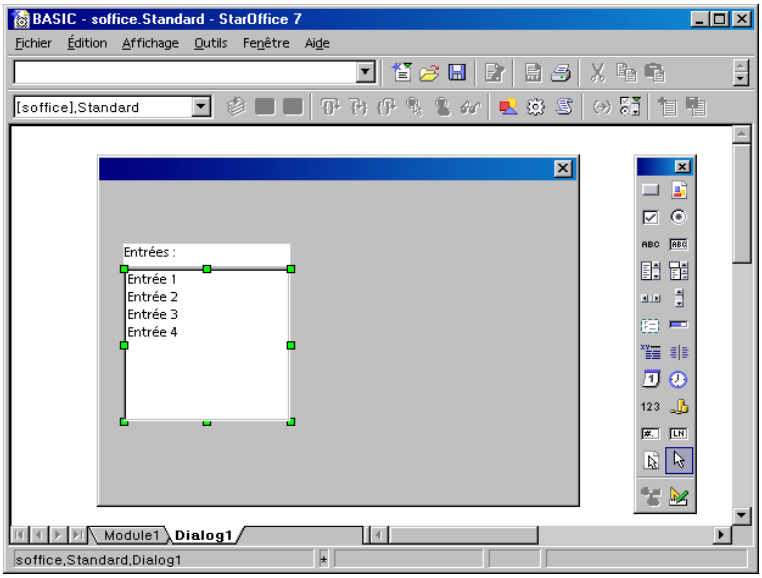
Création de boîtes de dialogue

Vous pouvez créer et structurer des boîtes de dialogue à l'aide de l'éditeur de boîte de dialogue de StarOffice que vous pouvez utiliser de la même manière qu'un dessin StarOffice :



Fondamentalement, vous faites glisser les éléments de contrôle que vous désirez depuis la palette de dessin (sur la droite) dans la zone de la boîte de dialogue dans laquelle vous pouvez en définir la position et la taille.

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 la 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 davantage d'informations sur l'utilisation de ces boutons à la section Détail 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 une valeur `1`, sinon, une valeur 0 est retournée.

```
Dim Dlg As Object

DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)

Select Case Dlg.Execute()
Case 1
      MsgBox "clic sur OK"
Case 0
      MsgBox "clic sur Annuler"
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 ouverte avec la méthode `endExecute` :

```
Dlg.endExecute()
```

Accès à des éléments de contrôle individuels

Une boîte de dialogue peut contenir un nombre illimité d'éléments de contrôle. 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`.

Veuillez noter que StarOffice Basic fait la différence entre les majuscules et les minuscules pour les noms des éléments de contrôle.

Utilisation du *modèle* de boîte de dialogue et d'élément 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 de 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 la représentation n'est pas toujours aussi claire que dans d'autres zones de l'API de StarOffice. Les éléments de l'API sont disponibles par l'intermédiaire de la vue et du modèle.

La propriété `Model` offre l'accès par programmation 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 a 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 du bord intérieur gauche de la boîte de dialogue (en unités ma)

- **`Model.PositionY (long)`** – position Y de l'élément de contrôle, mesurée à partir du bord intérieur supérieur de la boîte de dialogue (en unités ma)

Afin de garantir l'indépendance de l'apparence des boîtes de dialogue à l'égard des plates-formes, StarOffice utilise l'unité interne ***Map AppFont (ma)*** pour spécifier la position et la taille au sein des boîtes de dialogue. Une unité ma se définit comme le huitième de la hauteur moyenne d'un caractère de la police système définie dans le système d'exploitation et comme un 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 système.

Si vous souhaitez modifier la taille ou la position des éléments de contrôle pour 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.

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 de n'importe quelle boîte de dialogue en appuyant sur la touche Tab. 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 Tab
- **`Model.TabIndex (Long)`** – position de l'élément de contrôle dans l'ordre d'activation

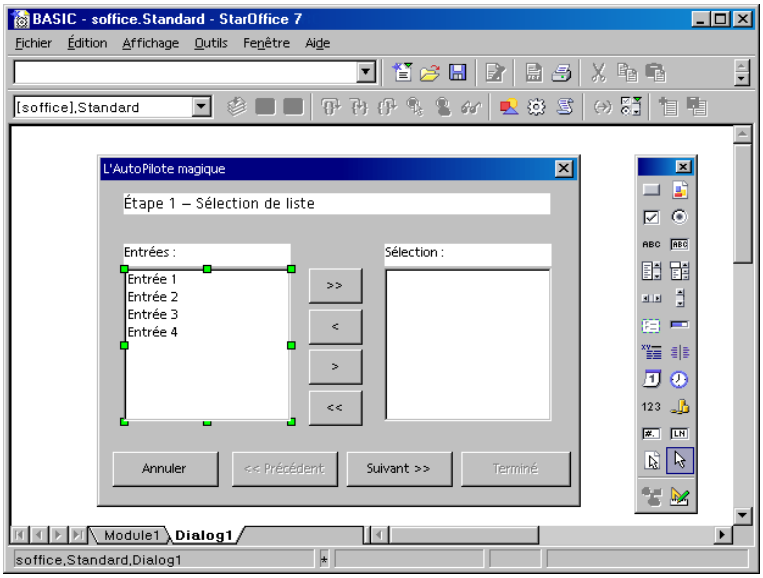
Enfin, l'élément de contrôle fournit une méthode `getFocus` garantissant que l'élément de contrôle sous-jacent reçoit le focus :

- **`getFocus`** – l'élément de contrôle reçoit le focus (uniquement pour les boîtes de dialogue)

Boîtes de dialogue à plusieurs pages

Une boîte de dialogue de StarOffice peut avoir plusieurs pages d'onglets. La propriété `Step` d'une boîte de dialogue définit sa page d'onglet courante tandis que la propriété `Step` d'un élément de contrôle spécifie la page d'onglet dans laquelle il doit s'afficher.

La valeur `Step` de `0` est un cas spécial. Si vous définissez cette valeur à 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 toutes les pages d'onglets d'une boîte de dialogue.



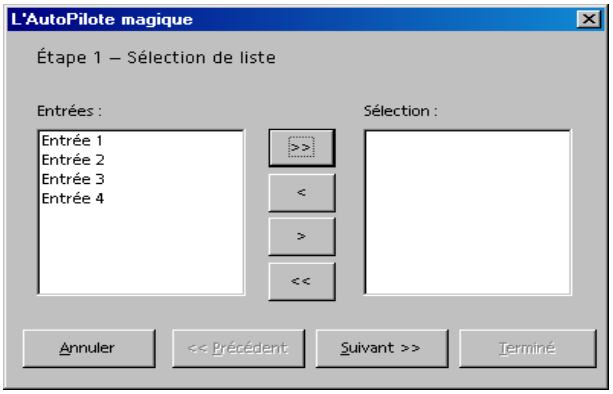
Dans l'exemple ci-dessus, vous pouvez également attribuer la valeur `Step` de `0` à la ligne de séparation ainsi que les boutons `Annuler`, `Précédent`, `Suivant` et `Terminé` pour afficher ces éléments sur toutes les pages. Vous pouvez également attribuer les éléments à une page d'onglet individuelle (par exemple à la page `1`).

Le code suivant montre comment la valeur de `Step` des gestionnaires d'événements des boutons `Suivant` et `Précédent` peut être augmentée ou réduite et modifie le statut de ces boutons.

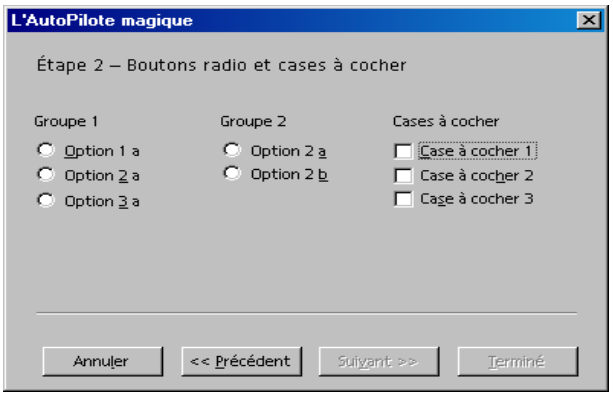
```
Sub cmdNext_Initiated
      Dim cmdNext As Object
      Dim cmdPrev As Object

      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 soit possible. La boîte de dialogue change ensuite d'apparence de la façon suivante :

Page 1 :



Page 2 :



Événements

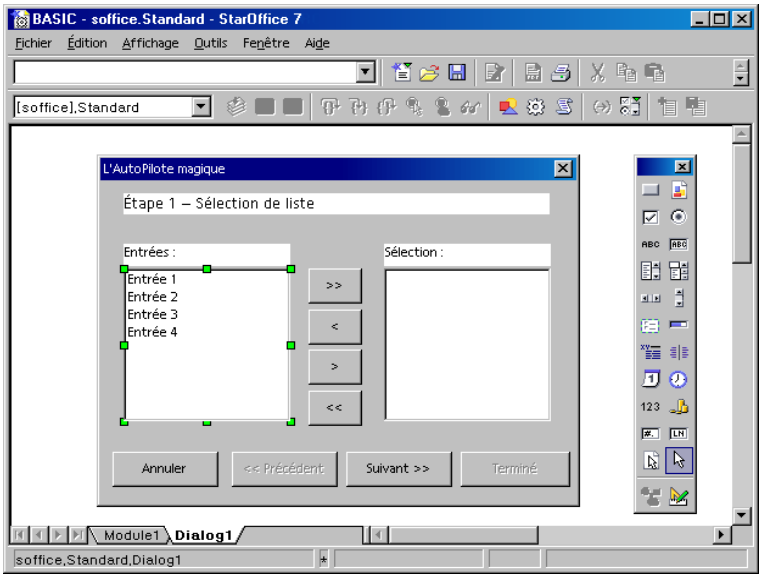
Les boîtes de dialogue et formulaires de StarOffice sont basés sur un modèle de programmation orienté objet dans lequel vous pouvez attribuer des ***gestionnaires d'événements*** aux éléments de contrôle. Un gestionnaire d'événements exécute une procédure prédéfinie lorsqu'une action particulière se produit, même lorsque cette action est un autre événement. Vous pouvez également éditer des documents ou des bases de données ouvertes avec le traitement des événements, et accéder à d'autres éléments de contrôle.

Les éléments de contrôle de StarOffice reconnaissent différents types d'événements pouvant être déclenchés dans différentes situations. Ces types d'événements peuvent être répartis en quatre groupes :

- **Contrôle par la souris** : événements correspondant à des actions de la souris (par exemple, des mouvements simples de la souris ou un clic sur un emplacement particulier à l'écran)
- **Contrôle par le clavier** : événements déclenchés par des séquences de touches

- **Modification du focus** : événements exécutés par StarOffice lorsque des éléments de contrôle sont activés ou désactivés
- **Événements spécifiques de l'élément de contrôle** : événements qui ne se produisent qu'en relation 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).



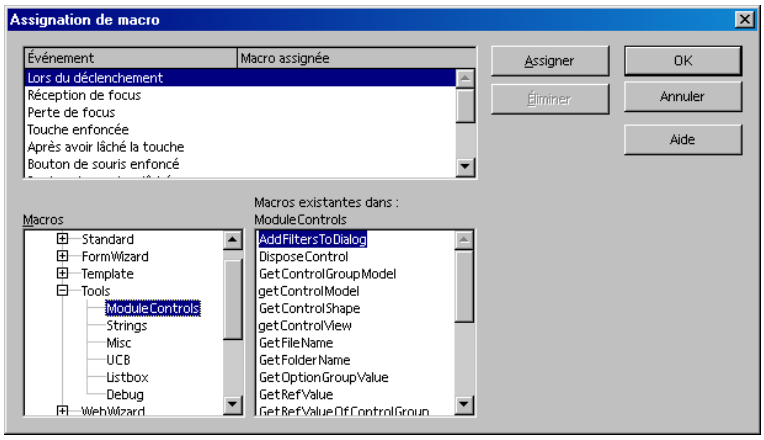
La figure ci-dessus illustre l'environnement de développement de StarOffice avec une boîte de dialogue contenant deux zones de liste. Vous pouvez déplacer les données d'une liste à l'autre en utilisant les boutons entre les deux zones de liste.

Si vous voulez afficher la présentation 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 n'importe quel module, 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. À la place, pour simplifier la gestion et le dépannage du code, créez plutôt une autre procédure servant de point d'entrée pour le traitement des événements, même si elle n'exécute qu'un seul appel à 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.SelectedItemPos, 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 de propriété 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 peut-être besoin de la position 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 davantage d'informations concernant un événement, par exemple :

```
Sub ProcessEvent(Event As Object)

End Sub
```

La précision avec laquelle l'objet `Event` est structuré et ses propriétés dépendent du type d'événement que l'appel à la procédure déclenche. Les sections suivantes décrivent en détail les types d'événements.

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 released`** – 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)`** – bouton enfoncé (une ou plusieurs constantes en fonction de `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 est également 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

L'exemple suivant sort 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 = "Touches : "
      If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
            Msg = Msg & "GAUCHE "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
            Msg = Msg & "DROIT "
      End If
      If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
            Msg = Msg & "MILIEU "
      End If
      Msg = Msg & Chr(13) & "Position : "
      Msg = Msg & Event.X & "/" & Event.Y
      MsgBox Msg
End Sub
```

Les événements VBA `Click` et `Doubleclick` ne sont pas disponibles dans StarOffice Basic. À la place, utilisez l'événement StarOffice Basic `MouseUp` pour l'événement `click` et imitez l'événement `Doubleclick` en changeant la logique de l'application.

Événements du clavier

Les événements suivants du clavier sont disponibles dans StarOffice Basic :

- **`Key pressed`** – l'utilisateur enfonce 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 pour sortir un seul caractère (par exemple, pour ajouter un accent à un caractère), StarOffice Basic ne crée qu'un événement.

Une action sur une seule touche ou une touche de modification, telle que la touche Maj ou Alt, ne crée pas un événement indépendant.

Les informations concernant une touche enfoncée sont indiquées par l'objet Event que StarOffice Basic fournit à la procédure pour le traitement 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 entré (prenant en considération les touches de modification)

L'exemple suivant utilise la propriété `KeyCode` pour établir si la touche Entrée, la touche Tab ou l'une des autres touches de contrôle a été enfoncée. Si l'une de ces touches a été enfoncée, son nom est retourné ; sinon le caractère saisi est retourné :

```
Sub KeyPressed(Event As Object)
      Dim Msg As String
      Select Case Event.KeyCode
      Case com.sun.star.awt.Key.RETURN
            Msg = "Touche Entrée enfoncée"
      Case com.sun.star.awt.Key.TAB
            Msg = "Touche Tabulation enfoncée"
      Case com.sun.star.awt.Key.DELETE
            Msg = "Touche Suppr enfoncée"
      Case com.sun.star.awt.Key.ESCAPE
            Msg = "Touche Échap enfoncée"
      Case com.sun.star.awt.Key.DOWN
            Msg = "Touche Bas enfoncée"
      Case com.sun.star.awt.Key.UP
            Msg = "Touche Haut enfoncée"
      Case com.sun.star.awt.Key.LEFT
            Msg = "Touche Gauche enfoncée"
      Case com.sun.star.awt.Key.RIGHT
            Msg = "Touche Droite enfoncée"
      Case Else
            Msg = "Caractère " & Event.KeyChar & " entered"
      End Select
      MsgBox Msg
End Sub
```

Des informations concernant d'autres constantes du clavier se trouvent dans la référence de l'API sous 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 spécifiques des éléments de contrôle

Outre les événements ci-dessus qui sont supportés 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`** – une action pouvant être exécutée lorsque l'élément de contrôle est déclenché (par exemple, lorsqu'un bouton est enfoncé)

Lorsque vous utilisez des événements, veuillez notez que certains événements, tels que `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 a changé lorsqu'un événement est en cours d'exécution.

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

Détail 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 d'heure
- Champs numériques
- Champs de devise
- Champs adoptant n'importe quel format

Boutons :

- Boutons standard
- Cases à cocher
- Boutons radio

Listes de sélection:

- Zones de liste
- Zones 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-dessous.

Boutons

Un bouton exécute une action lorsque vous cliquez dessus.

Le scénario le plus simple est celui dans lequel 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, la boîte de dialogue n'est pas modifiée. 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 de texte de l'élément de contrôle
- **`Model.HelpText (String)`** – texte d'aide qui s'affiche 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 d'option

Ces boutons sont généralement utilisés par groupes et vous permettent de sélectionner parmi une ou plusieurs options. Lorsque vous sélectionnez une option, toutes les autres options du groupe sont désactivées. Pour que ceci soit garanti à tout moment, seul un bouton d'option est défini.

Un élément de contrôle d'un bouton d'option 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 d'option :

- **`Model.FontDescriptor (struct)`** – structure avec les détails de la police utilisée (en fonction de `com.sun.star.awt.FontDescriptor`)
- **`Model.Label (String)`** - étiquette affichée sur l'élément de contrôle
- **`Model.Printable (Boolean)`** – élément de contrôle pouvant être imprimé
- **`Model.State (Short)`** – si cette propriété est égale à 1, l'option est activée, sinon elle est désactivée
- **`Model.TextColor (Long)`** – couleur de texte de l'élément de contrôle
- **`Model.HelpText (String)`** – texte d'aide qui s'affiche 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

Pour combiner plusieurs boutons d'option dans un groupe, 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 nouvel élément de contrôle pouvant être activé quel que soit le premier groupe d'éléments de contrôle.

Contrairement à VBA, vous ne pouvez pas insérer de boutons d'option dans un groupe d'éléments de contrôle dans StarOffice Basic. Le regroupement d'éléments de contrôle dans StarOffice Basic n'est utilisé que pour garantir une division visuelle en dessinant un cadre autour des éléments de contrôle.

Cases à cocher

Les cases à cocher servent à saisir une valeur Oui ou Non et, suivant le mode, elles peuvent adopter deux ou trois états. En plus des états Oui et Non, une case à cocher peut avoir un ***état intermédiaire*** si le statut correspondant Oui ou Non 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 pour 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 avec les détails de la police utilisée (correspond à la structure `com.sun.star.awt.FontDescriptor`)
- **Model.Label (String)** – étiquette pour 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)** – la touche Tab permet d'accéder à l'élément de contrôle
- **Model.TextColor (Long)** – couleur de texte de l'élément de contrôle
- **Model.HelpText (String)** – texte d'aide qui s'affiche lorsque vous laissez 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 nombres 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 non par les utilisateurs. Les champs de texte peuvent également servir de champs spéciaux numériques ou de devise, ainsi que de champs d'écran pour certaines tâches spéciales. Ces éléments de contrôle étant basés sur le service Uno `UnoControlEdit`, leur traitement contrôlé par programmation est semblable.

Les champs de texte offrent les propriétés suivantes :

- **Text (String)** – texte actif
- **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 active)
- **MaxTextLen (short)** – nombre maximum de caractères autorisés dans le champ
- **Editable (Boolean)** – `True` active l'option de saisie de texte, `False` bloque l'option de saisie (la propriété ne peut pas être appelée directement, mais seulement par l'intermédiaire de `IsEditable`)
- **IsEditable (Boolean)** – le contenu de l'élément de contrôle peut être changé, 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 pour 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 les champs de mot de passe
- **Model.FontDescriptor (struct)** – structure avec les détails de la police utilisée (correspond à la structure `com.sun.star.awt.FontDescriptor`)
- **Model.HardLineBreaks (Boolean)** – insertion automatique de retours à la ligne obligatoires dans le texte de l'élément de contrôle
- **Model.HScroll (Boolean)** –
- **Model.MaxTextLen (Short)** – longueur maximale du texte, où 0 correspond à aucune limite de longueur
- **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)** – la touche Tab permet d'accéder à l'élément de contrôle
- **Model.Text (String)** – texte de l'élément de contrôle
- **Model.TextColor (Long)** – couleur de texte de l'élément de contrôle
- **Model.VScroll (Boolean)** – le texte comporte une barre de défilement verticale
- **Model.HelpText (String)** – texte d'aide qui s'affiche 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

Zones de liste

Les zones de liste (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 avec les entrées mises en évidence, en lecture seule
- **SelectedItemPos (Short)** – numéro de l'entrée actuellement mise en évidence, en lecture seule

- **SelectedItemsPos (Array of Short)** – champ de données avec le nombre d'entrées mises en évidence (pour les listes supportant la sélection multiple), en lecture seule
- **MultipleMode (Boolean)** – `True` active l'option de sélection de plusieurs entrées, `False` bloque les sélections multiples (la propriété ne peut pas être appelée directement, mais seulement par l'intermédiaire de `IsMultipleMode`)
- **IsMultipleMode (Boolean)** – permet la sélection multiple dans les listes, en lecture seule

Les zones de liste offrent les méthodes suivantes :

- **addItem (Item, Pos)** – saisit la chaîne spécifiée dans `Item` dans la liste à la position `Pos`
- **addItems (ItemArray, Pos)** – saisit les entrées du champ de données `ItemArray` de la chaîne à la position `Pos` indiquée dans la liste
- **removeItems (Pos, Count)** – supprime les entrées `Count` qui se trouvent à 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 le champ de la liste pour rendre visible l'entrée spécifiée avec `Pos`

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 avec les détails de la police utilisée (correspond 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 des entrées
- **Model.SelectedItems (Array of Strings)** – liste des entrées mises en évidence
- **Model.StringItemList (Array of Strings)** – liste des 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)** – la touche Tab permet d'accéder à l'élément de contrôle
- **Model.TextColor (Long)** – couleur de 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

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 administrer une valeur numérique (par exemple une ID de base de données) en plus du texte en langage naturel, vous devez créer un champ de données auxiliaire qui effectue l'administration parallèlement à la zone de liste.

CHAPITRE **12**

Formulaires

À de nombreux égards, la structure des formulaires de 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.
- Un éditeur de boîte de dialogue permet de créer des boîtes de dialogue ; il se trouve dans l'environnement de développement de StarOffice Basic. Les formulaires sont créés en utilisant la barre d'outils des **fonctions de formulaire** directement dans le document.
- 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 sont différents sous plusieurs aspects.

Les utilisateurs souhaitant fournir leurs formulaires avec leurs propres méthodes de gestion d'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 de StarOffice peuvent contenir des champs de texte, des zones de liste, des boutons radio, ainsi qu'un ensemble d'autres éléments de contrôle, qui sont insérés directement dans un texte ou une feuille de calcul. La barre d'outils des **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 des **fonctions de formulaire** permet également de passer d'un mode à l'autre.

Détermination des 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 par l'intermédiaire de 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 déjà le nom de la méthode `GetByIndex`, un document peut contenir plusieurs formulaires. Ceci est pratique, par exemple, si 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 existe également dans ce but.

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. Pour le programmeur de StarOffice Basic, c'est le principal objet lorsqu'il travaille avec des formulaires d'éléments de contrôle.
- Son équivalent est la ***Vue*** de l'élément de contrôle, qui administre les informations d'affichage.
- Dans la mesure où les formulaires d'éléments de contrôle des documents sont administrés comme un élément spécial de dessin, il existe également un objet ***Shape*** qui représente les propriétés spécifiques de l'élément de dessin de 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 par l'intermédiaire de 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.

Si vous n'êtes pas sûr du formulaire d'un élément de contrôle, vous pouvez utiliser l'option de recherche dans tous les formulaires de l'élément de contrôle nécessaire :

```
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 terminée.

Accès à la vue des formulaires d'éléments de contrôle

Pour accéder à la vue d'un formulaire d'éléments de contrôle, il est nécessaire de disposer d'abord du modèle associé. La vue du formulaire d'éléments de contrôle peut être déterminée avec l'assistance du modèle et en utilisant le 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 Document `Doc` mais aussi l'objet de contrôleur de document `DocCrl` qui fait référence à la fenêtre de document courante. Avec l'aide de cet objet de 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 spécial, il est nécessaire de rechercher 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` nécessaire 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 ceci est vérifié, la fonction termine la recherche.

Détermination de la taille et de la position des éléments de contrôle

Comme cela a déjà été évoqué, la taille et la position des éléments de contrôle peuvent être déterminées à l'aide de l'objet `Shape` associé. La forme de l'élément de contrôle, comme tous les autres objets `Shape`, fournit les propriétés `Size` et `Position` à cette fin :

- **`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.x = 1000
Size.Width = 10000
Size.Width = 10000

Shape.Size = Size
```

```
Shape.Position = Point
```

L'objet Shape de l'élément de contrôle doit déjà être connu pour que le code fonctionne. 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. La sélection s'étend des champs de texte simples aux zones de liste, zones combinées et à différents 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, qui permet l'intégration de tables de bases de données complètes. Ceci est décrit à la section ***Formulaires de bases de données*** au chapitre 11.

Boutons

L'objet Model d'un bouton de formulaire fournit les propriétés suivantes :

- **BackgroundColor (long)** – couleur de l'arrière-plan.
- **DefaultButton (Boolean)** – le bouton sert de valeur par défaut. Dans ce cas, il répond également au bouton d'entrée s'il n'a pas de focus.
- **Enabled (Boolean)** – l'élément de contrôle peut être activé.
- **Tabstop (Boolean)** – l'élément de contrôle peut être atteint par l'intermédiaire de 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 l'accès par programmation.
- **TargetURL (String)** – URL cible pour les boutons du type de l'URL.
- **TargetFrame (String)** – nom de la fenêtre (ou du cadre) dans laquelle TargetURL doit être ouvert en activant le bouton (pour les boutons du type de l'URL).
- **Label (String)** – étiquette 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`).

Par l'intermédiaire de la propriété `ButtonType`, vous avez la possibilité de définir une action s'exécutant automatiquement lorsque le bouton est enfoncé. Le groupe `com.sun.star.form.FormButtonType` associé de constantes fournit les valeurs suivantes :

- **`PUSH`** – bouton standard.
- **`SUBMIT`** – fin de l'entrée du formulaire (particulièrement pertinent pour les formulaires HTML).
- **`RESET`** – réinitialise toutes les valeurs du formulaire à leurs valeurs d'origine.
- **`URL`** – appel de l'URL défini dans `TargetURL` (est ouvert dans la fenêtre spécifiée par l'intermédiaire de TargetFrame).

Les types de bouton **OK** et **Annuler** que l'on trouve dans les boîtes de dialogue ne sont pas supportés dans les formulaires.

Boutons d'option

Les propriétés suivantes d'un bouton d'option sont disponibles par l'intermédiaire de son objet model :

- **`Enabled (Boolean)`** – l'élément de contrôle peut être activé.
- **`Tabstop (Boolean)`** – l'élément de contrôle peut être atteint par l'intermédiaire de 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 l'accès par programmation.
- **`Label (String)`** – inscription du bouton.
- **`Printable (Boolean)`** – l'élément de contrôle peut être imprimé.
- **`State (Short)`** – si 1, l'option est activée, sinon, elle est désactivée.
- **`RefValue (String)`** – chaîne pour enregistrer des informations supplémentaires (par exemple, pour administrer les ID des enregistrements de données).
- **`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 d'option fait la distinction entre les éléments de contrôle pour les boîtes de dialogue et les formulaires. Tandis que les éléments de contrôle apparaissant l'un après l'autre dans les boîtes de dialogue sont automatiquement combinés pour constituer un groupe, le

regroupement des formulaires s'effectue sur la base des noms. Pour ce faire, tous les boutons d'option d'un groupe doivent contenir le même nom. StarOffice combine 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 simple modèle de contrôle d'élément. Il effectue une recherche en boucle dans tous les formulaires du document texte courant et utilise la méthode `HasByName` pour vérifier si le formulaire correspondant contient un élément comportant le nom `MyOptions` qu'il recherche. Si c'est le cas, l'accès à la matrice du modèle s'effectue en utilisant la méthode `GetGroupByName` (plutôt que la méthode `GetByName` pour déterminer des modèles simples).

Cases à cocher

L'objet Model d'un formulaire de cases à cocher fournit les propriétés suivantes :

- **`Enabled (Boolean)`** – l'élément de contrôle peut être activé.
- **`Tabstop (Boolean)`** – l'élément de contrôle peut être atteint par l'intermédiaire de 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 l'accès par programmation.
- **`Label (String)`** – étiquette du bouton.
- **`Printable (Boolean)`** – l'élément de contrôle peut être imprimé.
- **`State (Short)`** – si 1, l'option est activée, sinon, elle est désactivée.

- **RefValue (String)** – chaîne pour enregistrer des informations supplémentaires (par exemple, pour administrer les ID des enregistrements de données).
- **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

Les objets Model des formulaires de champs de texte fournissent 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 : pas de 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 automatiques à la ligne sont insérés en permanence dans le texte de l'élément de contrôle.
- **HScroll (Boolean)** – le texte a une barre de défilement horizontale.
- **MaxTextLen (Short)** – longueur maximale du texte ; si 0 est indiqué, il n'existe aucune limite.
- **MultiLine (Boolean)** – permet 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 par l'intermédiaire de 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)** – couleur du texte de l'élément de contrôle.
- **TextColor (Long)** – couleur du texte de l'élément de contrôle.
- **VScroll (Boolean)** – le texte a 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 des formulaires 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 : pas de bordure, 1 : cadre 3D, 2 : cadre simple).
- **FontDescriptor (struct)** – structure comportant les détails de la police utilisée (correspondant à la structure `com.sun.star.awt.FontDescriptor`).
- **LineCount (Short)** – nombre de lignes dans l'élément de contrôle.
- **MultiSelection (Boolean)** – permet la sélection de plusieurs 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 (par exemple, pour administrer les ID des enregistrements de données).
- **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 par l'intermédiaire de 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 l'accès par programmation.
- **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.

Par l'intermédiaire de leur propriété `ValueItemList`, les formulaires de zone de liste fournissent un équivalent de la propriété VBA, `ItemData`, par l'intermédiaire duquel vous pouvez administrer des informations supplémentaires pour des entrées de liste individuelles.

De plus, les méthodes suivantes sont fournies par l'intermédiaire de l'objet View de la zone de liste :

- **addItem (Item, Pos)** – insère la chaîne spécifiée dans `Item` dans la liste à la position `Pos`.
- **addItems (ItemArray, Pos)** – insère les entrées répertoriées dans le champ de saisie `ItemArray` de la chaîne dans la liste 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` suivant la variable `SelectMode`.
- **`makeVisible (Pos)`** – fait défiler le champ de la liste pour rendre visible l'entrée spécifiée avec `Pos`.

Formulaires de base de données

Il est possible de lier directement les formulaires StarOffice à une base de donnée. Les formulaires créés de cette manière offrent toutes les fonctions d'une base de données frontale complète, sans recours à la programmation indépendante.

L'utilisateur peut parcourir et rechercher les tables et requêtes sélectionnées, ainsi que modifier des enregistrements de données et en insérer de nouveaux. StarOffice vérifie automatiquement que les données pertinentes sont récupérées à partir de la base de données et que toute modification effectuée est récrite 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 spécifiques des bases de données suivantes doivent également être définies dans le formulaire :

- **`DataSourceName (String)`** – nom de la source de données (reportez-vous au chapitre 10, ***Accès aux bases de données***, Accès aux bases de données ; la source de données doit être globalement créée dans StarOffice).
- **`Command (String)`** – nom de la table, requête, ou commande SQL select vers laquelle un lien doit être établi.
- **`CommandType (Const)`** - spécifie si la `Command` est une table, requête, ou commande SQL (valeur à partir de l'énumération `com.sun.star.sdb.CommandType`).

L'énumération `com.sun.star.sdb.CommandType` recouvre les valeurs suivantes :

- **`TABLE`** – Table
- **`QUERY`** - Requête
- **`COMMAND`** – commande SQL

Les champs de la base de données sont assignés aux éléments de contrôle individuels par l'intermédiaire de cette propriété :

- **`DataField (String)`** – nom du champ de la base de données liée.

Tables

Un autre élément de contrôle est fourni pour utiliser des bases de données : l'élément de contrôle de table. Ceci représente le contenu d'une table ou 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 en utilisant le formulaire autopilote reliant toutes les colonnes aux champs correspondants de la base de données, conformément aux spécifications de l'utilisateur. Du fait que API associée est relativement complexe, nous n'en fournirons pas une description complète pour l'instant.

Annexe

Conseils de migration VBA

Conseils de migration StarOffice 5.x

Index

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X

Y

Z