

	Programmation Macro OpenOffice Formation OpenOffice	Diffusible à : Société Guidella
--	--	------------------------------------

et vous trouver...

TITRE DU DOCUMENT
Programmation Macro OpenOffice

RESUME
Formation sur le langage Basic OpenOffice Elément de la programmation basic pour écrire et modifier des macros d'OpenOffice Exemple de modification sur la macro expendedPDF de création d'un PDF.

PROJET	Numéro Affaire	CLIENT	Responsable GUIDELLA
Formation OpenOffice	NC		

HISTORIQUE			
Date	Révision	Auteur	Commentaires
11/12/04	Rev 1.1	Philippe Laneres	

Documents de références		
Auteur	Titre	Commentaires
Sun	Programmation Basic	« ProgrammationBasique.pdf »
Vincent Mormot	Comment faire une Macro pour choisir un Modèle via une Boite de Dialogue	Inséré dans le document
Bernard Marcelly	L'API OpenOffice.org (presque) sans peine	Inséré dans le document

Confidentiel		
	Imprimé le 28 décembre 2004	1/186



Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Table des matières

<u>I Le langage de StarOffice Basic</u>	10
<u>I.1 Lignes de programme</u>	10
<u>I.1.1 Lignes liées</u>	10
<u>I.1.2 Plusieurs sections pour une ligne</u>	10
<u>I.1.3 Commentaires</u>	10
<u>I.1.4 Marqueur</u>	10
<u>I.2 Utilisation des variables</u>	11
<u>I.2.1 Déclaration de variable implicite</u>	11
<u>I.2.2 Déclaration de variable explicite</u>	11
<u>a) Première ligne de code interdisant les variables implicites</u>	11
<u>b) Variables de type variant</u>	11
<u>c) Déclaration d'une variable associée à un type particulier</u>	11
<u>d) Variables chaînes de caractères Unicode</u>	11
<u>e) Spécification de chaînes explicites</u>	11
<u>f) Variables entières</u>	11
<u>g) Variables de type entier long</u>	12
<u>h) Variables de type simple</u>	12
<u>i) Variables de type double</u>	12
<u>j) Variables monétaires</u>	12
<u>I.2.3 Spécification de nombres explicites</u>	12
<u>a) Nombres décimaux</u>	12
<u>b) Notation exponentielle</u>	13
<u>c) Valeurs hexadécimales</u>	13
<u>d) Valeurs octales</u>	13
<u>e) Variables logiques</u>	13
<u>f) Variables de date</u>	13
<u>I.2.4 Champs de données</u>	13
<u>a) Matrices simples</u>	13
<u>b) Champs de données multidimensionnels</u>	13
<u>c) Modifications dynamiques des dimensions des champs de données</u>	14
<u>I.2.5 Portée et durée de vie des variables</u>	14
<u>a) Variables locales</u>	14
<u>b) Variables du domaine public</u>	14
<u>c) Variables globales</u>	15
<u>d) Variables privées</u>	15
<u>e) Constantes</u>	15
<u>I.3 Opérateurs</u>	16
<u>I.3.1 Opérateurs mathématiques</u>	16
<u>I.3.2 Opérateurs logiques</u>	16
<u>I.3.3 Opérateurs de comparaison</u>	16

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

2/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.4 Instructions conditionnelles	17
I.4.1 If...Then...Else	17
I.4.2 Select...Case	17
I.5 Boucles	18
I.5.1 For...Next	18
I.5.2 Do...Loop	18
I.5.3 While...Wend	19
I.5.4 Exemple de programme : tri à l'aide de boucles imbriquées	19
I.6 Procédures et fonctions	20
I.6.1 Procédures	20
I.6.2 Fonctions	20
I.6.3 Interruption prématurée d'une procédure et d'une fonction	20
I.6.4 Passage de paramètres	20
I.6.5 Paramètres facultatifs	21
I.6.6 Récursivité	21
I.6.7 Traitement des erreurs	22
a) L'instruction On Error	22
b) La commande Resume Next	22
c) la commande Resume Proceed	22
d) Requêtes portant sur les erreurs	23
e) Astuces pour le traitement d'erreur structuré	23
II La bibliothèque d'exécution de StarOffice Basic	24
II.1 Fonctions de conversion	24
II.2 Fonctions de vérification du type de variable	25
II.3 Chaînes de caractères	26
II.3.1 Utilisation des jeux de caractères	26
II.3.2 Sous chaîne	26
II.3.3 Recherche et remplacement	27
II.3.4 Formatage des chaînes	28
II.4 Date et heure	29
II.4.1 Spécification de la date et de l'heure à l'intérieur du code du programme	29
II.4.2 Extraction des informations de date et d'heure	29
II.4.3 Obtention de l'heure et de la date système	29
II.5 Fichiers et répertoires	30
II.5.1 Administration des fichiers	30
a) Recherche dans les répertoires	30
b) Création et suppression de répertoires	30
c) Copie, nouveau nom, suppression et vérification de l'existence de fichiers	30
d) Lecture et modification des propriétés d'un fichier	31
II.5.2 Écriture et lecture de fichiers texte	31
a) Écriture de fichiers texte	31
b) Lecture de fichiers texte	33

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

3/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.6 Boîtes de message et zones de saisie	34
II.6.1 Affichage de messages	34
II.6.2 Zone de saisie pour demander des chaînes simples	35
II.7 Autres fonctions	36
II.7.1 Beep	36
II.7.2 Shell	36
II.7.3 Wait	36
II.7.4 Environ	36
III Introduction à l' API StarOffice	37
III.1 Propriétés et méthodes	37
III.1.1 Propriétés	37
a) Propriétés réelles et propriétés imitées	37
III.1.2 Méthodes	37
III.2 Modules, services et interfaces	38
III.3 Outils pour l'utilisation d' UNO	38
III.3.1 La méthode supportsService	38
III.3.2 Propriétés de débogage	38
III.3.3 Référence de l'API	39
III.4 Présentation de quelques interfaces centrales	39
III.4.1 Création d'objets contextuels	39
a) Interface com.sun.star.lang.XMultiServiceFactory	39
III.4.2 Accès par nom aux objets subordonnés	39
a) L'interface com.sun.star.container.XNameAccess	40
b) L'interface com.sun.star.container.XNameContainer	41
III.4.3 Accès par indice aux objets subordonnés	41
a) L'interface com.sun.star.container.XIndexAccess	41
b) L'interface com.sun.star.container.XIndexContainer	42
III.4.4 Accès itératif aux objets subordonnés	42
a) Les interfaces com.sun.star.container.XEnumeration et XEnumerationAccess	42
IV Utilisation de documents StarOffice	43
IV.1 StarDesktop	43
IV.1.1 Informations de base sur les documents dans StarOffice	43
a) Noms de fichier en notation URL	43
b) Format de fichier XML	43
b.a) Compression des fichiers	43
IV.1.2 Création, ouverture et import de documents	44
a) Option SearchFlags	44
b) Options FileProperties	45
c) Création de nouveaux documents	46
IV.1.3 Objets Document	46
a) Enregistrement et export de documents	46
a.a) Options de la méthode storeAsURL	47

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

4/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

<u>b) Impression de documents</u>	48
<u>b.a) Options de la méthode print</u>	48
<u>b.b) Sélection et paramétrage de l'imprimante</u>	48
<u>IV.2 Modèles</u>	49
<u>V Comment faire une Macro pour choisir un Modèle via une Boite de Dialogue</u>	51
<u>V.1 Écriture d'une macro</u>	51
<u>V.2 Création de la librairie</u>	51
<u>V.3 Création de la boîte de dialogue</u>	55
<u>V.4 Assignation de la macro</u>	58
<u>VI Exemple de programmation basic</u>	61
<u>VI.1 Scanner un document et écrire les informations sur celui-ci</u>	61
<u>VI.2 Convertir et exporter des documents OOo en d'autres formats</u>	62
<u>VI.2.1 Définition de filtre en fonction du suffixe OO (sxc, stc)</u>	62
<u>VI.2.2 Transformer le document</u>	66
<u>VI.2.3 Remplacer une chaîne par une autre</u>	67
<u>VII L'API OpenOffice.org (presque) sans peine</u>	69
<u>VII.1 Caractéristiques du Basic OOo</u>	69
<u>VII.1.1 Enregistreur de macro</u>	69
<u>VII.1.2 Accès aux macros</u>	69
<u>VII.1.3 Esthétique et lisibilité</u>	69
<u>a) Définitions de variables</u>	70
<u>VII.1.4 Trouver le document</u>	71
<u>a) Le document en cours</u>	71
<u>b) Accéder à un autre document existant</u>	71
<u>c) Créer un nouveau document</u>	72
<u>d) Créer un nouveau document à partir d'un modèle</u>	72
<u>e) Quelques détails complémentaires</u>	73
<u>e.a) StarDesktop</u>	73
<u>e.b) ThisComponent</u>	73
<u>e.c) Arguments de LoadComponentFromURL</u>	74
<u>e.d) Propriétés pour ouvrir un document</u>	74
<u>VII.1.5 Sauver le document</u>	75
<u>a) Enregistrer sous / Enregistrer une copie sous</u>	76
<u>b) Fermer le document</u>	77
<u>VII.1.6 Tableur Calc</u>	77
<u>a) Trouver la feuille</u>	77
<u>a.a) Une feuille existante</u>	77
<u>a.b) Créer, renommer, dupliquer, supprimer une feuille</u>	78
<u>a.c) La feuille visible par l'utilisateur</u>	79
<u>b) Trouver la cellule</u>	79
<u>c) Zone de cellules</u>	80
<u>c.a) Manipuler une zone de cellules</u>	80

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

5/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

<u>c.b) Zone de cellules sélectionnées par l'utilisateur</u>	81
<u>c.c) Sélectionner visiblement une zone de cellules</u>	81
<u>d) Retrouver les coordonnées d'une cellule</u>	82
<u>e) Connaître le type de contenu de la cellule</u>	82
<u>f) Valeur numérique dans la cellule</u>	83
<u>g) Texte dans la cellule</u>	84
<u>g.a) Le curseur d'écriture de texte dans la cellule</u>	84
<u>g.b) Caractères spéciaux dans un texte</u>	84
<u>g.c) Déplacement du curseur de texte dans une cellule</u>	85
<u>h) Formule dans la cellule</u>	85
<u>i) Recalculer les formules des cellules</u>	86
<u>j) Modifier le format d'une cellule</u>	86
<u>j.a) Style d'une cellule</u>	86
<u>j.b) Couleur de fond de la cellule</u>	87
<u>j.c) Couleur de caractère</u>	88
<u>j.d) Taille de caractère</u>	88
<u>j.e) Justification horizontale</u>	88
<u>j.f) Justification verticale</u>	88
<u>j.g) Orientation verticale/horizontale</u>	89
<u>j.h) Orientation à angle quelconque</u>	89
<u>j.i) Retour à la ligne</u>	89
<u>k) Formater des zones de cellules</u>	90
<u>l) Lignes et colonnes</u>	90
<u>l.a) Définir un ensemble de colonnes</u>	90
<u>l.b) Ajouter ou supprimer des colonnes</u>	90
<u>l.c) Optimiser la largeur d'une colonne</u>	91
<u>l.d) Imposer une largeur de colonne</u>	91
<u>l.e) Ajouter ou supprimer des lignes</u>	91
<u>l.f) Imposer une hauteur de ligne</u>	91
<u>l.g) Cacher des lignes ou des colonnes</u>	91
<u>m) Manipuler des données en tableau</u>	92
<u>n) Curseur de cellule</u>	93
<u>o) Appeler des fonctions Calc dans une macro</u>	94
<u>p) Trier une table</u>	95
<u>p.a) Tri par colonnes</u>	95
<u>p.b) Tri par lignes</u>	97
<u>q) Les dessins dans Calc</u>	97
<u>q.a) Insérer une forme dessinée</u>	97
<u>q.b) Ancrage de la forme</u>	99
<u>q.c) Positionner la forme</u>	99
<u>q.d) Autres fonctionnalités autour de la forme</u>	100
<u>r) Les images dans Calc</u>	100

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

6/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

<u>r.a) Insérer une image</u>	100
<u>r.b) Positionner une image</u>	101
<u>r.c) Dimensionner une image</u>	101
<u>r.d) Insérer plusieurs images</u>	102
<u>r.e) Trouver une image par son nom</u>	103
<u>VII.1.7 Traitement de texte Writer</u>	104
<u>a) Trouver le texte</u>	104
<u>b) Gérer les curseurs</u>	104
<u>b.a) Curseur visible, curseur d'écriture</u>	104
<u>b.b) Déplacer le curseur d'écriture</u>	104
<u>b.c) Le curseur visible</u>	106
<u>b.d) Zone sélectionnée par l'utilisateur</u>	107
<u>b.e) Sélectionner visiblement une zone</u>	108
<u>c) Ecrire du texte</u>	109
<u>c.a) Lire, écrire</u>	109
<u>c.b) Caractères spéciaux dans un texte</u>	109
<u>c.c) Insérer un saut de page ou de colonne</u>	110
<u>c.d) Exemples de manipulation de texte</u>	111
<u>c.e) Astuce : utiliser les signets pour écrire un texte</u>	112
<u>d) Appliquer un style ou un formatage</u>	113
<u>d.a) Appliquer un style à un paragraphe</u>	113
<u>d.b) Affecter un style à un caractère</u>	113
<u>d.c) Affecter un formatage à un caractère</u>	114
<u>d.d) Remettre le formatage du style du paragraphe</u>	117
<u>e) Exercice d'écriture de texte</u>	117
<u>f) Tabuler un texte</u>	119
<u>g) Recherche et remplacement de texte</u>	120
<u>g.a) Remplacer une chaîne partout dans un texte</u>	120
<u>g.b) Remplacer un nom de style partout dans un texte</u>	122
<u>g.c) Chercher et modifier du texte</u>	122
<u>h) Les tableaux dans un document Writer</u>	125
<u>h.a) Insérer un tableau</u>	125
<u>h.b) Propriétés du tableau</u>	125
<u>h.c) Largeur du tableau</u>	126
<u>h.d) Se déplacer dans un tableau</u>	128
<u>h.e) Ecrire du texte dans une cellule de tableau</u>	131
<u>h.f) Somme des cellules d'un tableau</u>	132
<u>h.g) Accéder à une ligne</u>	132
<u>h.h) Colorer le fond d'un tableau</u>	133
<u>h.i) Insérer plusieurs tableaux</u>	133
<u>h.j) Trouver un tableau par son nom</u>	134
<u>h.k) Tableaux irréguliers</u>	134

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

7/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

<u>i) Les cadres dans un document Writer</u>	137
<u>i.a) Insérer un cadre</u>	137
<u>i.b) Les cadres élastiques</u>	137
<u>i.c) Les différents ancrages de cadre</u>	138
<u>i.d) Concepts utilisés pour le positionnement du cadre</u>	138
<u>i.e) Positionnement absolu du cadre</u>	139
<u>i.f) Positionnement horizontal relatif</u>	139
<u>i.g) Positionnement vertical relatif</u>	141
<u>i.h) Exemple complet d'insertion de cadre</u>	142
<u>i.i) Ecrire du texte dans un cadre</u>	143
<u>i.j) Insérer plusieurs cadres</u>	143
<u>i.k) Exemple de cadre dans un cadre</u>	144
<u>i.l) Couleur du fond du cadre</u>	145
<u>i.m) Trouver un cadre par son nom</u>	145
<u>j) Les dessins dans Writer</u>	147
<u>j.a) La page de dessin de Writer</u>	147
<u>j.b) Insérer une forme dessinée</u>	147
<u>j.c) Les différents ancrages d'une forme</u>	148
<u>j.d) Positionnement de la forme</u>	148
<u>j.e) Insérer plusieurs formes</u>	149
<u>j.f) Autres fonctionnalités autour de la forme</u>	150
<u>k) Les images dans Writer</u>	151
<u>k.a) Insérer une image</u>	151
<u>k.b) Positionner une image</u>	152
<u>k.c) Dimensionner une image</u>	152
<u>k.d) Insérer plusieurs images</u>	153
<u>k.e) Trouver une image par son nom</u>	154
<u>VII.1.8 Dessin Draw</u>	154
<u>a) Trouver la page de dessin</u>	154
<u>a.a) Une page existante</u>	154
<u>a.b) Créer, renommer, dupliquer, supprimer une page</u>	155
<u>a.c) La page visible par l'utilisateur</u>	157
<u>a.d) Propriétés d'une page de dessin</u>	157
<u>b) Les dessins dans Draw</u>	158
<u>b.a) Insérer une forme dessinée</u>	158
<u>b.b) Positionner la forme</u>	159
<u>b.c) Autres fonctionnalités des formes</u>	159
<u>c) Les images dans Draw</u>	160
<u>c.a) Insérer une image</u>	160
<u>c.b) Positionner une image</u>	161
<u>c.c) Dimensionner une image</u>	161
<u>c.d) Insérer plusieurs images</u>	162

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

8/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c.e) Trouver une image par son nom	163
VII.1.9 Présentation Impress	164
a) Fonctions identiques à Draw	164
b) Fonctions spécifiques à Impress	164
VII.1.10 Les formes : fonctions communes à toutes les applications	164
a) Propriétés des formes	164
a.a) Couleur de fond de la forme	164
a.b) Forme avec une ombre	165
a.c) Rotation de la forme	165
a.d) Effet parallélogramme	165
a.e) Apparence du trait de contour	166
a.f) Exemple récapitulatif	167
b) Dessiner des formes	169
b.a) Les différentes formes de base	169
b.b) Le rectangle	170
b.c) L'ellipse	170
b.d) La ligne simple	171
b.e) La ligne brisée	172
b.f) Le Polygone	173
b.g) Combiner plusieurs formes	174
b.h) Disposition relative des formes	176
b.i) Grouper des formes	176
c) Ecrire du texte dans la forme	177
d) Trouver une forme	178
d.a) Trouver les formes sélectionnées par l'utilisateur	178
d.b) Nommer une forme	178
d.c) Trouver une forme par son nom	178
d.d) Sélectionner visiblement une forme	179
VII.1.11 Informations annexes	180
a) Basic simplifie l'utilisation de l'API	180
a.a) Les parenthèses optionnelles	180
a.b) Les getXxxx et setXxxx	180
a.c) L'accès aux propriétés	181
b) Qu'est-ce qu'un mot ?	181
c) Les horribles constantes	182
d) Les Couleurs	182
e) Consulter l'API	183
VIII Licence	186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I Le langage de StarOffice Basic

I.1 Lignes de programme

I.1.1 Lignes liées

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

I.1.2 Plusieurs sections pour une ligne

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

I.1.3 Commentaires

```
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.
```

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

I.1.4 Marqueur

Un programme StarOffice Basic peut contenir des dizaines, des centaines, voire des milliers de marqueurs (noms donnés aux variables, constantes, fonctions, etc.)

- 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 S.
- 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.

```
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)
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

10/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.2 Utilisation des variables

I.2.1 Déclaration de variable implicite

Les trois variables sont déclarées implicitement

```
a = b + c
```

I.2.2 Déclaration de variable explicite

a) Première ligne de code interdisant les variables implicites.

```
Option Explicit
```

b) Variables de type variant.

```
MyVar = "Hello World" ' Assignment d'une chaîne de caractères
```

```
MyVar = 1 ' Assignment d'un nombre entier
```

```
MyVar = 1.0 ' Assignment d'un nombre à virgule flottante
```

```
MyVar = True ' Assignment d'une valeur logique
```

c) Déclaration d'une variable associée à un type particulier

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

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

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

d) Variables chaînes de caractères Unicode

```
Dim Variable As String
```

```
Dim Variable$
```

e) Spécification de chaînes explicites

```
Dim MyString As String
```

```
MyString = " Ceci est un test"
```

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

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

f) Variables entières

```
Dim Variable As Integer
```

```
Dim Variable%
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

11/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

g) Variables de type entier long

Entre -2 147 483 648 et 2 147 483 647, ils peuvent occuper quatre octets de mémoire au maximum.

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

h) Variables de type simple

Entre $3,402823 \times 10^{38}$ et $1,401298 \times 10^{-45}$, ils peuvent occuper quatre octets de mémoire au maximum.

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

i) Variables de type double

Entre $1,79769313486232 \times 10^{308}$ et $4,94065645841247 \times 10^{-324}$, ils peuvent occuper huit octets de mémoire au maximum.

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

j) Variables monétaires

Entre -922 337 203 685 477,5808 et +922 337 203 685 477,5807 et peut occuper huit octets de mémoire au maximum.

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

I.2.3 Spécification de nombres explicites

```
Dim A As Integer  
Dim B As Float  
A = 1210  
B = 2438  
A = + 121  
B = - 243
```

a) Nombres décimaux

Séparez la partie entière de la partie décimale à l'aide d'un point (.).

```
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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

12/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b) Notation exponentielle

```
Dim A As Double
A = 1.43E2 ' Correct
A = + 1.43E2 ' Correct
A = - 1.43E2 ' Correct
A = 1.43E-2 ' Correct
A = 1.43E -2 ' Incorrect
A = 1,43E-2 ' Incorrect
A = 1.43E2.2 ' Incorrect
```

c) Valeurs hexadécimales

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

d) Valeurs octales

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

e) Variables logiques

```
Dim Variable As Boolean
```

f) Variables de date

```
Dim Variable As Date
```

I.2.4 Champs de données

a) Matrices simples

```
Dim MyArray(3) ' quatre variables de type variant
Dim MyInteger(3) As Integer ' quatre variables entières
Dim MyInteger(5 To 10) ' 6 variables entières des indices 5 à 10
Dim MyInteger(-10 To -5) ' 6 variables entières des indices -10 à -5
```

b) Champs de données multidimensionnels

Matrice d'entiers à deux dimensions, comportant chacune six valeurs (pouvant être adressées par les valeurs d'indice de 0 à 5).

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

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

13/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c) Modifications dynamiques des dimensions des champs de données

```
ReDim MyArray(10) As Integer
```

Pour conserver les valeurs d'origine, utilisez la commande Preserve avec des des types de données identiques

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

I.2.5 Portée et durée de vie des variables

a) Variables locales

```
Sub Test  
    Dim MyInteger1 As Integer  
    Static MyInteger2 As Integer  
    ' ...  
End Sub
```

b) Variables du domaine public

Une variable du domaine public n'est accessible que durant l'exécution de la macro associée et est ensuite réinitialisée. Ces variables sont accessibles à tous les modules de leur bibliothèque

- Section d'en-tête d'un module par le mot-clé Dim.
- Public peut être employé à la place de Dim pour déclarer une variable du domaine public

```
Public A As Integer  
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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

14/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c) Variables globales

Leurs valeurs sont conservées même après l'exécution de la macro associée.

```
Global A As Integer
```

d) Variables privées

Pour un module

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
```

```
    C = 10
```

```
End Sub
```

```
Sub ShowVarA
```

```
    MsgBox C ' Affiche la variable C du module A.
```

```
End Sub
```

Module B

```
Private C As Integer
```

```
Sub SetModuleB
```

```
    C = 20
```

```
End Sub
```

```
Sub ShowVarB
```

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

```
End Sub
```

e) Constantes

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

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.3 Opérateurs

I.3.1 Opérateurs mathématiques

- + 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)

I.3.2 Opérateurs logiques

- 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)

I.3.3 Opérateurs de comparaison

Tous les types de variables élémentaires

- = 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

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

16/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.4 Instructions conditionnelles

I.4.1 If...Then...Else

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

I.4.2 Select...Case

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

```
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
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.5 Boucles

I.5.1 For...Next

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

```
Dim I
For I = 1 To 10 Step 0.5
    ' ... Corps de la boucle
Next I
```

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

```
Dim I
For I = 1 To 10
    If I = 5 Then
        Exit For
    End If
    ' ... Corps de la boucle
Next I
```

I.5.2 Do...Loop

Vérifie si la condition est toujours remplie avant chaque passage

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

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

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

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

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

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

```
Do
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
' ... corps de la boucle
Loop Until A > 10
Do
    If A = 4 Then
        Exit Do
    End If
    ' ... corps de la boucle
While A > 10
```

I.5.3 While...Wend

Vérifie si la condition est toujours remplie avant chaque passage

```
While A > 10
    ' ... corps de la boucle
Wend
```

I.5.4 Exemple de programme : tri à l'aide de boucles imbriquées

Tri à bulles

```
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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

19/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.6 Procédures et fonctions

I.6.1 Procédures

```
Sub Test
    ' ... ici se trouve le code de la procédure à proprement parler
End Sub
Test
```

I.6.2 Fonctions

```
Function Test
    ' ... ici se trouve le code de la fonction à proprement parler
    Test = 12 ' Une valeur de retour
    '...
    Test = 123 ' Une autre valeur de retour
End Function
```

```
Dim A
A = Test ' A = 0 si pas de valeur de retour
```

```
Function Test As Integer
    ' ... ici se trouve le code de la fonction à proprement parler
End Function
```

I.6.3 Interruption prématurée d'une procédure et d'une fonction

```
Sub Test
    Dim ErrorOccured As Boolean
    ' ...
    If ErrorOccured Then
        Exit Sub
    End If
    ' ...
End Sub
```

I.6.4 Passage de paramètres

Les paramètres sont normalement passés par référence

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

```
Sub Test
    Dim A As Integer
    A = 10
    ChangeValue(A) ' Le paramètre A vaut à présent 20
End Sub
Sub ChangeValue(TheValue As Integer)
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

20/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
TheValue = 20  
End Sub
```

Passage par valeur, dans l'exemple theValue ne change pas.

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

I.6.5 Paramètres facultatifs

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

```
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
```

I.6.6 Récursivité

```
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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

21/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

I.6.7 Traitement des erreurs

a) L'instruction On Error

```
Sub Test
    On Error Goto ErrorHandler
    ' ... lance la tâche au cours de laquelle
    ' une erreur est susceptible de se produire
Exit Sub
' Traitement des erreurs
ErrorHandler:
    ' ... code particulier pour le traitement de l'erreur
End Sub
```

b) La commande Resume Next

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
```

Pour reprendre un programme sans afficher de message d'erreur lorsqu'une erreur se produit, utilisez le format suivant avec précaution, car elle a un effet global.

```
Sub Test
    On Error Resume Next
    ' ... accomplit la tâche au cours de laquelle
    ' une erreur est susceptible de se produire
End Sub
```

c) la commande Resume Proceed

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
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

d) 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.

e) Astuces pour le traitement d'erreur structuré

La meilleure solution consiste à n'utiliser qu'une seule approche pour le traitement des erreurs à l'intérieur d'un programme

Le numéro de l'erreur est stockée dans la constante imaginaire ExpectedErrorNo.

```
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
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II La bibliothèque d'exécution de StarOffice Basic

II.1 Fonctions de conversion

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
```

- 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.
- CDbI(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.

```
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")
```

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
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.2 Fonctions de vérification du type de variable

- 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.

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.3 Chaînes de caractères

II.3.1 Utilisation des jeux de caractères

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

Inversion, 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 forcée.

```
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).

II.3.2 Sous 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.

```
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
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.3.3 Recherche et remplacement

```
ResultString = InStr (SearchString, MyString)
ResultString = InStr(StartPosition, SearchString, MyString)
ResultString = InStr(SearchString, MyString, 0) ' distingue les majuscules
et les minuscules
```

```
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 Mid de StarOffice Basic a été étendue

```
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.

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.3.4 Formatage des chaînes

- 0 - 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.
- , - Une virgule représente le caractère utilisé par le système d'exploitation comme séparateur de milliers.
- # - Le signe dièse représente un chiffre ou un emplacement qui n'est affiché que si la chaîne en entrée le nécessite.
- \$ - 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 "-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"
```

```
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"
```

```
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"
```

```
MyFormat = "#,##0.00 $"  
MyString = Format(-1579.8, MyFormat) ' Renvoie "-1 579,80 E"  
MyString = Format(1579.8, MyFormat) ' Renvoie "1 579,80 E"  
MyString = Format(0.4, MyFormat) ' Renvoie "0,40 E"  
MyString = Format(0.434, MyFormat) ' Renvoie "0,43 E"
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

28/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.4 Date et heure

II.4.1 Spécification de la date et de l'heure à l'intérieur du code du programme

Cette formulation peut provoquer des erreurs suivant la formulation des pays

```
Dim MyDate As Date  
MyDate = "1.1.2002"
```

Vaut mieux utiliser

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

Année, mois, jour

```
Dim MyVar As Date  
MyDate = TimeSerial (11, 23, 45)
```

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

II.4.2 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

II.4.3 Obtention de l'heure et de 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)

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

29/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.5 Fichiers et répertoires

II.5.1 Administration des fichiers

a) Recherche dans les répertoires

```
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

b) Création et suppression de répertoires

```
Mkdir ("C:\SubDir1\Subdir2") ' crée répertoires et sous répertoires
Rmdir ("C:\SubDir1\SubDir2\SubDir3\") ' supprime SubDir3 +
tout ce qu'il y a dedans
```

c) Copie, nouveau nom, suppression et vérification de l'existence de fichiers

```
FileCopy(Source, Destination) ' crée une copie du fichier Source
Name OldName As NewName ' Renomme le fichier
Kill (Filename) ' Supprime le fichier
FileExists (Filename) ' Renvoie un booleen
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

30/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

d) Lecture et modification des propriétés d'un fichier

La valeur de retour de « GetAttr » 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

```
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".

```
SetAttr("test.txt", 1) ' Passe un fichier en lecture seule.
SetAttr("test.txt", 0) ' Supprime le statut de lecture seule.
FileDateTime("test.txt") ' Renvoie la date et l'heure de la dernière
                          ' modification du fichier formatée
selon                    ' les paramètres nationaux
                          ' utilisés sur le
                          ' système.
FileLen("test.txt") ' Renvoie la longueur du fichier en octets.
```

II.5.2 Écriture et lecture de fichiers texte

a) Écriture de fichiers 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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

31/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

une ligne Close #FileNo

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

32/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b) Lecture de fichiers 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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

33/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.6 Boîtes de message et zones de saisie

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

II.6.1 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

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

34/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

`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

```
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".

II.6.2 Zone de saisie pour demander des chaînes simples

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.

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

II.7 Autres fonctions

II.7.1 Beep

`Beep` produit un son d'avertissement

II.7.2 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.

II.7.3 Wait

`Wait 2000` spécifie d'attendre 2 secondes (2 000 millisecondes).

II.7.4 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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

36/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

III Introduction à l' API StarOffice

Interface de programmation orientée objet

```
Dim Obj As Object
```

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

<http://api.openoffice.org/docs/common/ref/com/sun/star/frame/Desktop.html>

Cet appel déclare une variable objet nommée Obj et l'instancie en un service : com.sun.star.frame.Desktop.

Cela correspond aux termes de type et de classe employés dans d'autres langages de programmation.

Un Universal Network Object (UNO) peut supporter plusieurs services en même temps (un peu comme plusieurs classes à la fois).

III.1 Propriétés et méthodes

III.1.1 Propriétés

Les propriétés sont définies grâce à une simple assignation

```
Document.Title = "Manuel de programmation StarOffice 7.0"
```

```
Document.Filename = "progman.sxv"
```

a) Propriétés réelles et propriétés imitées

Il existe 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.

Par exemple, les objets caractère dans UNO, proposent les méthodes getPosition et setPosition qui permettent de connaître et de modifier le point clé associé. Le programmeur peut également accéder à ces valeurs via la propriété Position.

III.1.2 Méthodes

```
Document.Save()
```

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

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

37/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

III.2 Modules, services et interfaces

StarOffice dispose de centaines de services regroupés en modules.

com.sun.star.(module).(service)

`com.sun.star.frame.Desktop`

<http://api.openoffice.org/docs/common/ref/com/sun/star/frame/Desktop.html>

Au sens le plus strict du mot, un service dans UNO ne supporte pas des méthodes, mais des interfaces, qui, elles, proposent différentes méthodes. Les méthodes sont assignées (sous forme de combinaisons) au service dans des interfaces. Un grand nombre d'interfaces sont utilisées dans les différents services. Si vous êtes familier avec une interface, vous pourrez appliquer les connaissances acquises d'un service à un autre.

III.3 Outils pour l'utilisation d' UNO

III.3.1 La méthode supportsService

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

<http://api.openoffice.org/docs/common/ref/com/sun/star/text/Paragraph.html>

Détermine si l'objet TextElement supporte le service com.sun.star.text.Paragraph.

III.3.2 Propriétés de déboguage

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.

```
Dim Obj As Object
```

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

```
MsgBox Obj.DBG_Properties
```

```
MsgBox Obj.DBG_methods
```

```
MsgBox Obj.DBG_supportetInterfaces
```

Avant d'appeler une propriété, vous devez utiliser la fonction `IsEmpty` pour vérifier qu'elle est effectivement disponible.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

38/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

III.3.3 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/docs/common/ref/com/sun/star/module-ix.html>

III.4 Présentation de quelques interfaces centrales

III.4.1 Création d'objets contextuels

La fonction `createUnoService` crée un objet qui peut être utilisé de manière universelle, 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.

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

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

La méthode `createInstance`, définie dans l'interface `XMultiServiceFactory`, est utilisée en particulier dans les objets `Document`. L'objet de dessin peut, par exemple, être créé de la manière suivante en utilisant un objet `Spreadsheet`

```
Dim RectangleShape As Object  
RectangleShape = thisComponent.CreateInstance  
("com.sun.star.drawing.RectangleShape")
```

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

```
Dim Style as Object  
Textdocument = ThisComponent  
Families = Textdocument.StyleFamilies  
Style = Textdocument.CreateInstance("com.sun.star.style.ParagraphStyle")  
ParaStyles = Families.GetByName("ParagraphStyles")
```

<http://api.openoffice.org/docs/common/ref/com/sun/star/style/ParagraphStyle.html>

III.4.2 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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

39/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

a) L'interface `com.sun.star.container.XNameAccess`

<http://api.openoffice.org/docs/common/ref/com/sun/star/container/XNameAccess.html>

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 = ThisComponent.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 = ThisComponent.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
Spreadsheet= ThisComponent
Sheets = Spreadsheet.Sheets
If Sheets.HasByName("Feuille1") Then
    MsgBox " Feuille 1 existe"
Else
    MsgBox "Feuille1 n'existe pas"
End If
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b) 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 Object
Dim ParagraphStyles As Object
Dim NewStyle As Object
Textdoc=ThisComponent
StyleFamilies = Textdoc.StyleFamilies
ParagraphStyles = StyleFamilies.getByname("ParagraphStyles")
NewStyle = monDocument.CreateInstance("com.sun.star.style.ParagraphStyle")
'ParagraphStyles.insertByName("NewStyle", NewStyle)
'ParagraphStyles.replaceByName("ChangingStyle", NewStyle)
'ParagraphStyles.removeByName("OldStyle")
newStyle.ParentStyle = "Signature" ' hériter d'un style
newStyle.CharColor = RGB(0,100,255) ' changer la couleur End Sub
```

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

III.4.3 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.

a) L'interface `com.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 = ThisComponent.Sheets
For I = 0 to Sheets.getCount() - 1
    Sheet = Sheets.getByIndex(I) ' Édition de la feuille
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

41/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Next I

L'exemple montre une boucle parcourant toutes les 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.

b) 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.

III.4.4 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.

a) 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 = ParagraphEnumeration.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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

42/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

IV Utilisation de documents StarOffice

IV.1 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.

IV.1.1 Informations de base sur les documents dans StarOffice

a) Noms de fichier en notation URL

StarOffice utilise cette notation dans sa base

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

Fonctions de conversion

```
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
```

b) 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.

b.a) 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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

43/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

IV.1.2 Création, ouverture et import de documents

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

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

- URL du fichier associé.
- Frame est un nom pour l'objet cadre de la fenêtre que StarOffice crée en interne pour son administration.
 - `_blank` est généralement utilisé ici, et indique à StarOffice de créer une nouvelle fenêtre.
 - `_hidden` charge le document correspondant mais de le garder invisible.
- SearchFlags activé quand la fenêtre de travail (framework) est censée exister

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

- FileProperties

L'utilisateur peut ouvrir un nouveau document StarOffice avec ces 2 seuls premiers paramètres, les deux derniers pouvant ne contenir que des substituts (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())
```

a)Option SearchFlags

Le troisième paramètre est utilisé pour le remplacement du contenu de la fenêtre de document

Frame dans ce cas doit contenir un nom explicite

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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

44/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b) Options FileProperties

Le quatrième paramètre 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.

FileProperties 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.
- CharSet (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`
`/share/registry/res/fr/org/openoffice/Office/TypeDetection.properties`
- 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 = "scalcalc: 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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

45/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c)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.

IV.1.3 Objets Document

La fonction `loadComponentFromURL` présentée dans la section précédente renvoie un objet `Document`. Elle supporte le service `com.sun.star.document.OfficeDocument`,

<http://api.openoffice.org/docs/common/ref/com/sun/star/document/OfficeDocument.html>

qui fournit à son tour deux interfaces centrales

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

a)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

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

```
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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

46/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- `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
```

a.a) Options de la méthode `storeAsURL`

<http://api.openoffice.org/docs/common/ref/com/sun/star/frame/XStorable.html#storeAsURL>

<http://api.openoffice.org/docs/common/ref/com/sun/star/beans/PropertyValue.html>

<http://api.openoffice.org/docs/common/ref/com/sun/star/document/MediaDescriptor.html>

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
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

47/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Url = "file:///C:/test3.sxw"  
FileProperties(0).Name = "Overwrite"  
FileProperties(0).Value = True  
Doc.storeAsURL(sUrl, mFileProperties())
```

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)
```

b) Impression de documents

b.a) Options de la méthode print

- 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).

```
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())
```

b.b) 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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

48/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- 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.

IV.2 Modèles

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 = ThisComponent.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.

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

49/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

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.

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

V Comment faire une Macro pour choisir un Modèle via une Boite de Dialogue

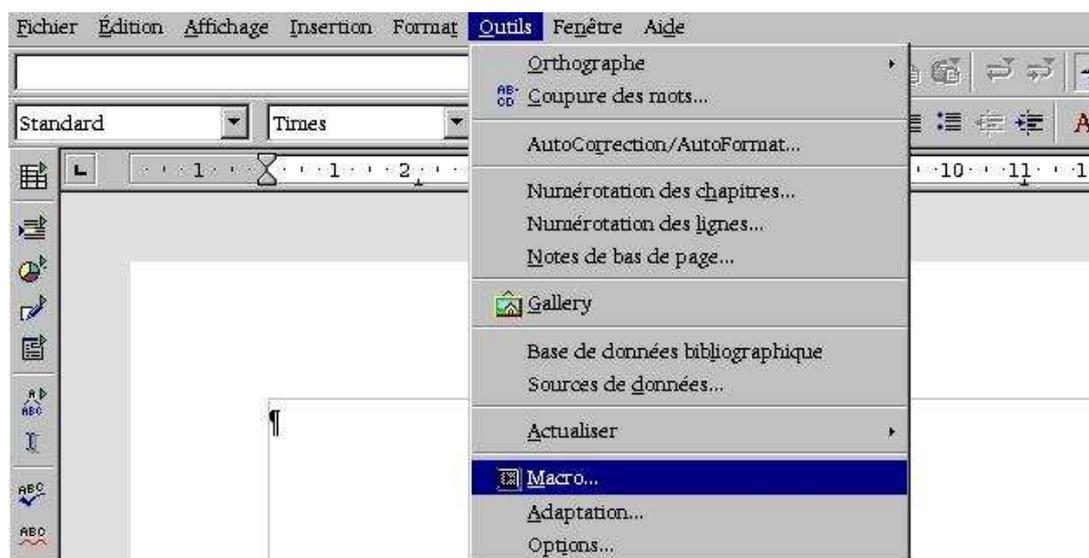
V.1 Écriture d'une macro

L'utilisation de modèles impose d'aller rechercher le modèle correspondant au type de documents que l'on veut réaliser. Le but de la macro que nous allons concevoir est de proposer, à l'utilisateur, les modèles disponibles par l'intermédiaire d'une boîte de dialogue et de créer le document en fonction de son choix.

La démarche consiste dans un premier temps à réaliser une librairie (bibliothèque), ensemble regroupant différentes fonctions qui seront utilisées dans notre macro et la boîte de dialogue affichée à l'utilisateur.

V.2 Création de la librairie

Dans un premier temps lancez OpenOffice.org Writer et accédez au menu **Outils**



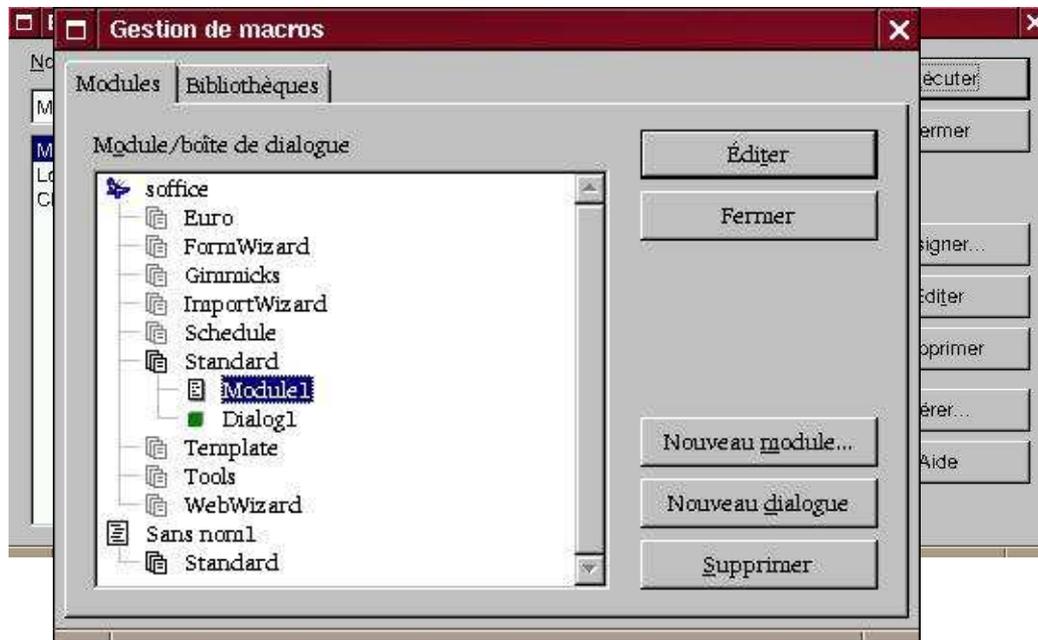
Puis Cliquer sur le bouton Gérer

Programmation Macro OpenOffice

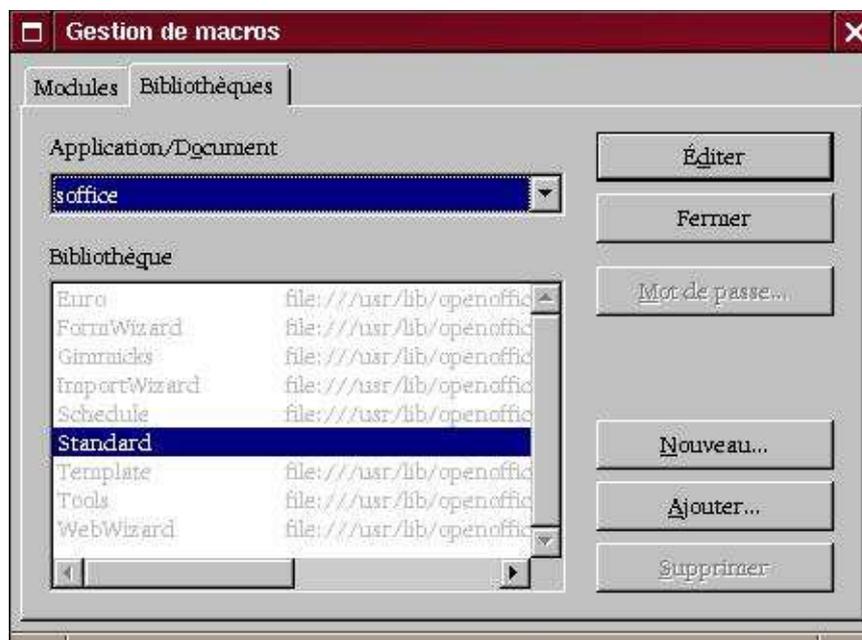
Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



Choisissez l'onglet **Bibliothèques**, puis le bouton **Nouveau**



Nommez la nouvelle bibliothèque

Programmation Macro OpenOffice

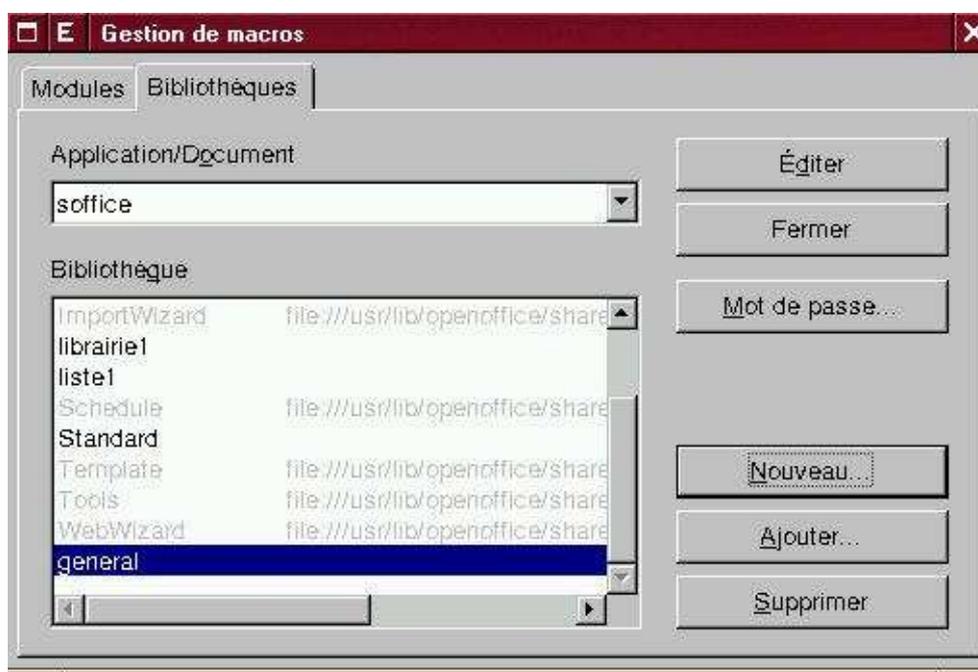
Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



Validez puis cliquez sur le bouton **Editer**



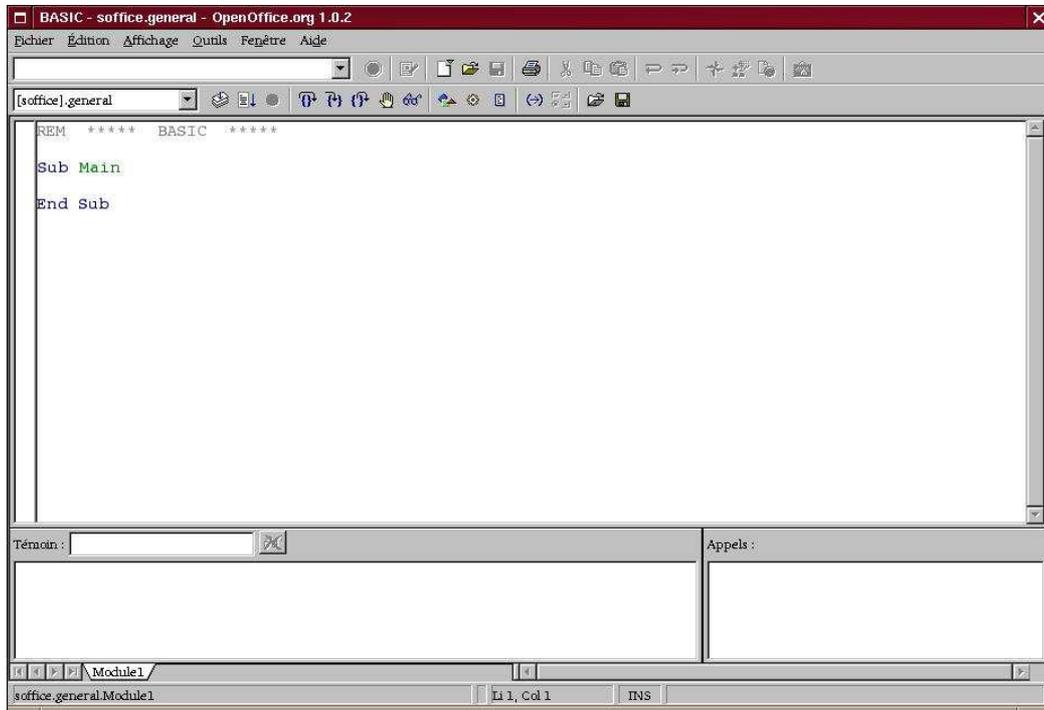
Le code associé à la macro s'affiche

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



Créez une Fonction

```
Function LoadDialog( Libname as String, DialogName as String)  
End Function
```

et une procédure

```
Sub Choix(event as Object)  
End Sub
```

le code sera complété par la suite. Nous aurons besoin de ces points d'entrée pour affecter les évènements aux éléments constituant la boîte de dialogue.

Programmation Macro OpenOffice

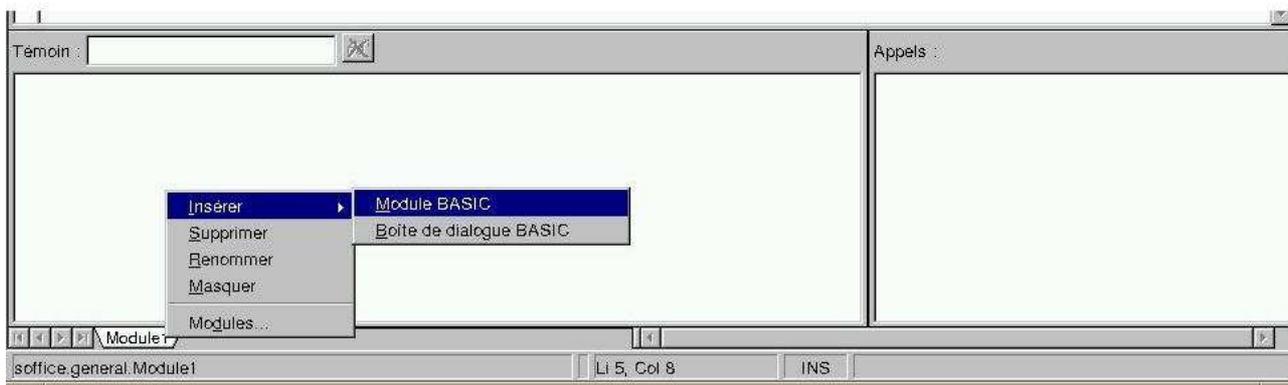
Statut : En cours

**Formation
OpenOffice**

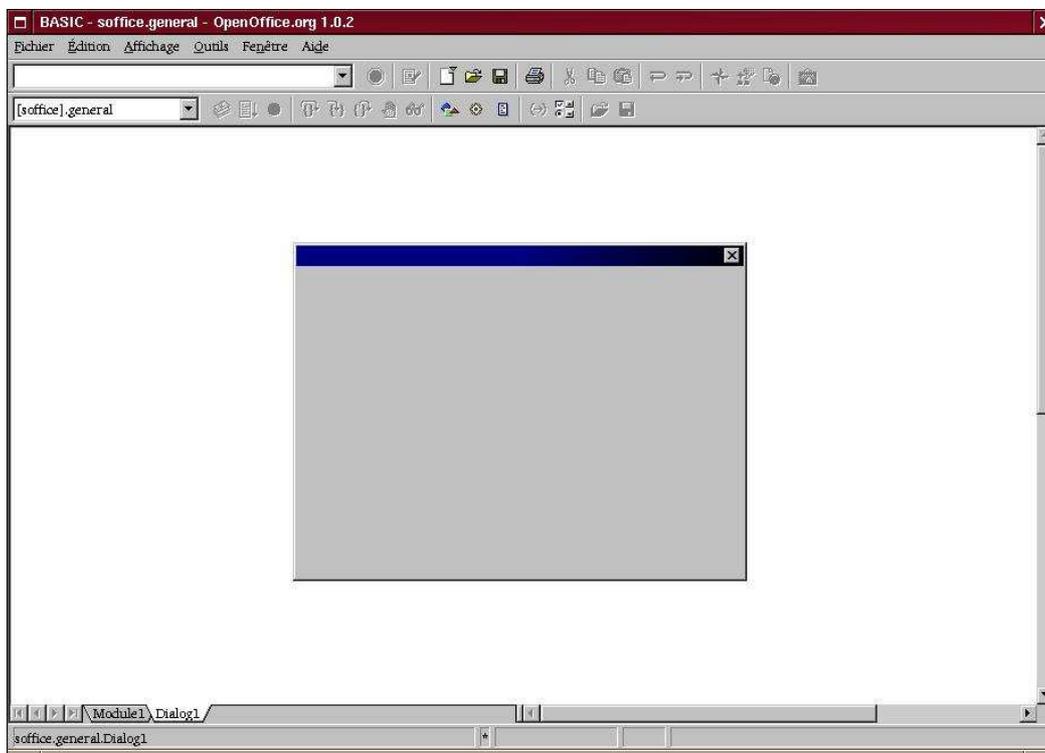
Révision : 1.1

V.3 Création de la boîte de dialogue

Déplacez le pointeur de la souris sur l'onglet **Module1** en bas à gauche et cliquez sur le bouton droit de la souris pour accéder au menu contextuel associé aux onglets. Choisissez **Insérer** puis **boîte de dialogue BASIC**.



Vous devez obtenir l'écran suivant.



Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

55/186

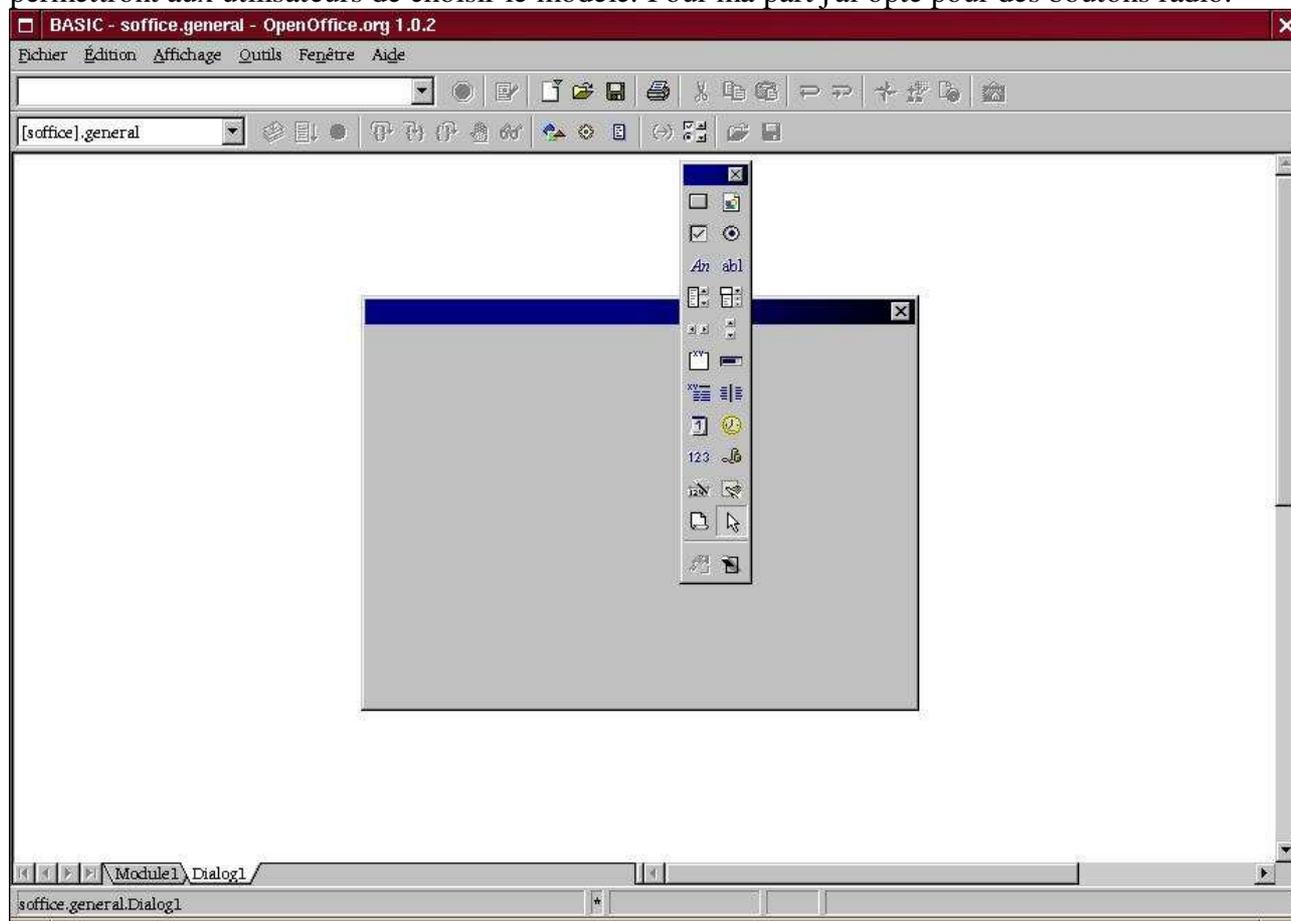
Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Sélectionner alors les outils pour incorporer dans cette boite de dialogue, les éléments qui permettront aux utilisateurs de choisir le modèle. Pour ma part j'ai opté pour des boutons radio.



Une fois l'élément placé sur cette boite de dialogue double cliquez sur celui-ci pour obtenir le menu **Propriétés** et allez dans un premier temps sous l'onglet **Général**, modifiez le nom et le titre. J'ai réduit le nom à 2 caractères (prendre une notation simple) nous nous en servons pour tester le choix de l'utilisateur.

Programmation Macro OpenOffice

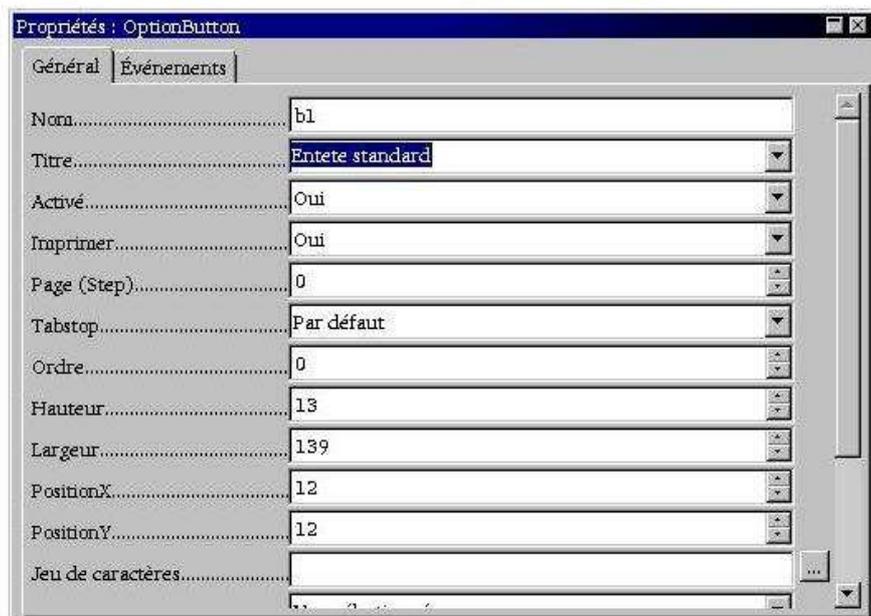
Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



Puis passez sur l'onglet **Evènements** et cliquez sur les points de suite pour affecter une action lors du déclenchement sur cet objet.



Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

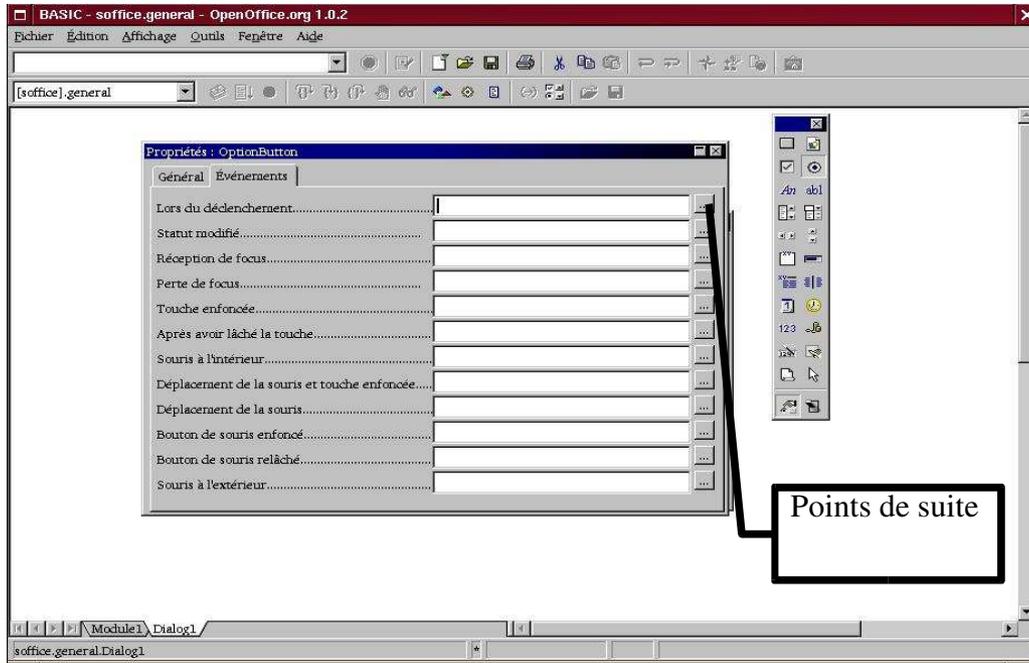
57/186

Programmation Macro OpenOffice

Statut : En cours

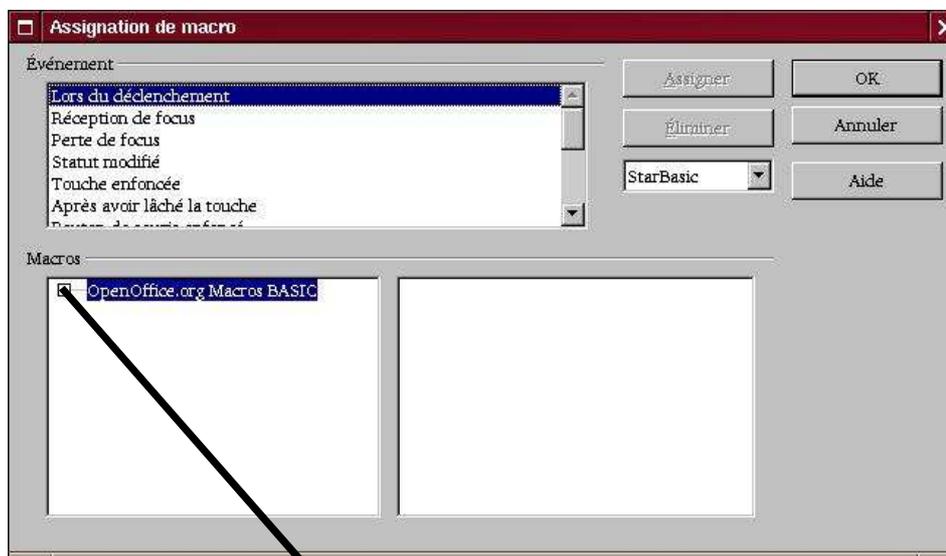
**Formation
OpenOffice**

Révision : 1.1



V.4 Assignment de la macro

Suivre les instructions sur les écrans ci-après :

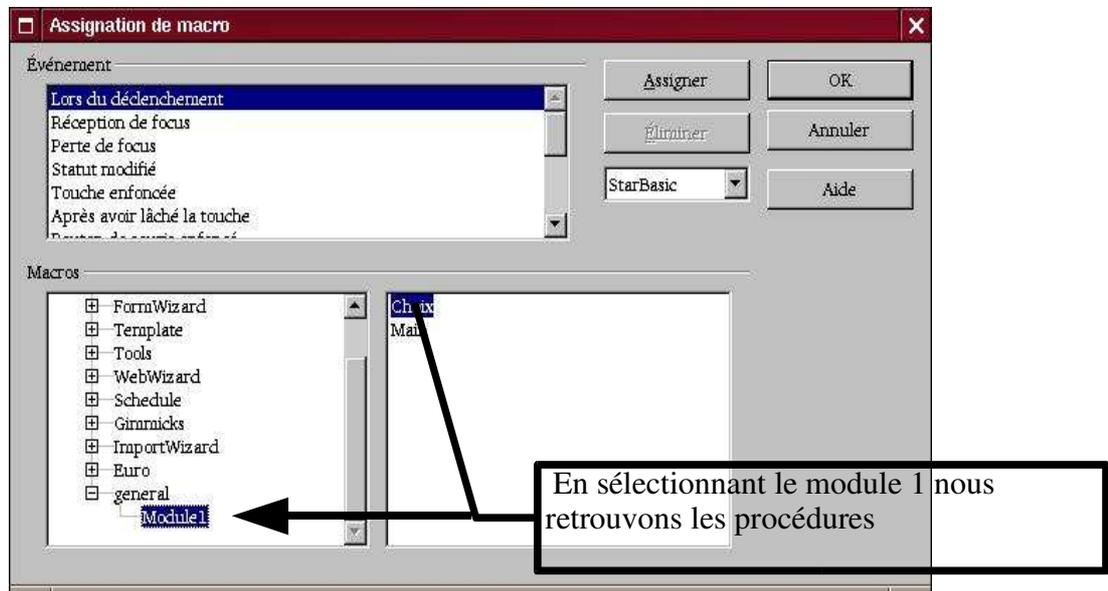


Programmation Macro OpenOffice

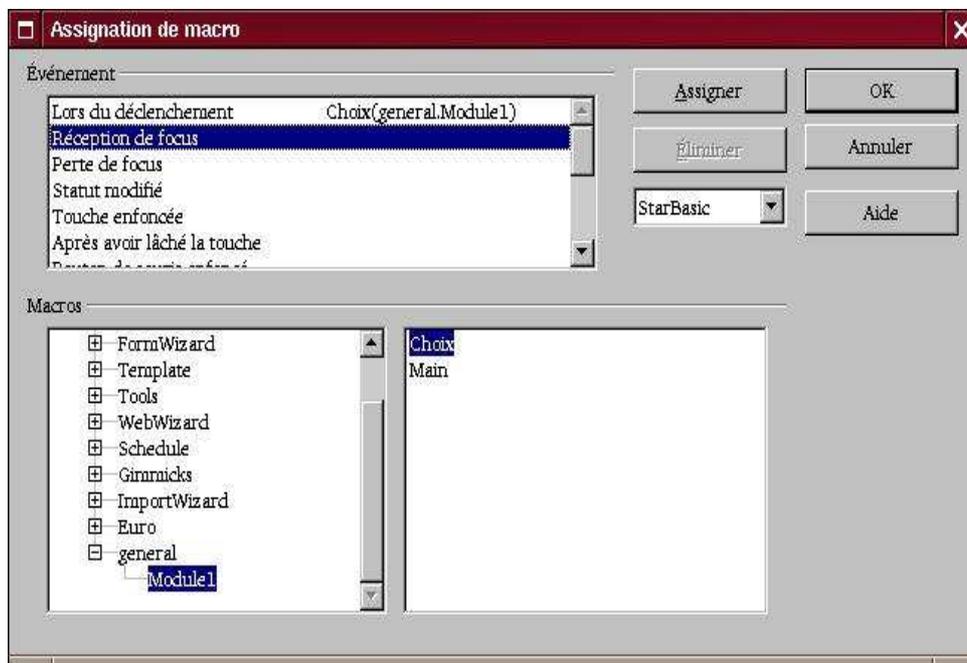
Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



N'oubliez pas de valider l'affectation en appuyant sur le bouton **Assigner** avant de refermer la



boite par le bouton **OK**

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

59/186

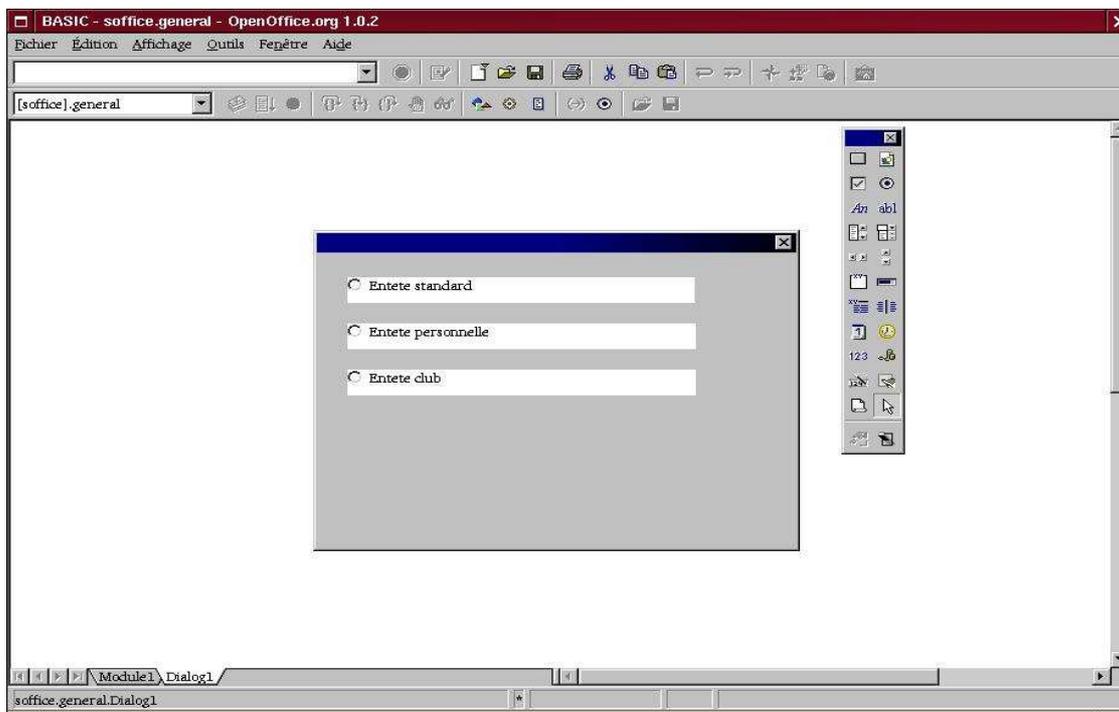
Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

La boîte de dialogue après insertion de plusieurs boutons radio.



Répétez la même opération pour les autres éléments que vous voulez incorporer dans la boîte de dialogue.

Reprenez sur l'onglet **Module1** et tapez alors le code suivant

```
REM ***** BASIC *****  
Dim oDialog AS Object  
Sub Main  
    oDialog = LoadDialog("general", "Dialog1")  
    oDialog.Execute()  
End Sub  
Function LoadDialog(LibName as String, DialogName as String)  
    Dim oLib as Object  
    Dim oLibDialog as Object  
    Dim oRuntimeDialog as Object  
    oLibContainer = DialogLibraries  
    oLibContainer.LoadLibrary(LibName)  
    oLib = oLibContainer.GetByName(LibName)  
    oLibDialog = oLib.GetByName(DialogName)
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

60/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
oRuntimeDialog = CreateUnoDialog( oLibDialog)
LoadDialog() = oRuntimeDialog
End Function

Sub Choix(event as Object)
  Dim oSource as Object
  Dim sName as String
  oSource = event.Source.model
  sName = oSource.Name

  Select case sName
    case "b1"
      oDocNewFile = STARDESKTOP.loadComponentFromURL
("file:///T:/documents/OO/writer/entete.stw", "_blank", 0, NoArg())
    case "b2"
      oDocNewFile = STARDESKTOP.loadComponentFromURL
("file:///T:/documents/OO/writer/entete2.stw", "_blank", 0, Noarg())
    case "b3"
      oDocNewFile = STARDESKTOP.loadComponentFromURL
("file:///T:/documents/OO/writer/entete3.stw", "_blank", 0, Noarg())
  End Select
  oDialog.EndExecute()
End Sub
```

VI Exemple de programmation basic

VI.1 Scanner un document et écrire les informations sur celui-ci

```
sub enumFormats()
'
'Laurent Godard - oooconv@free.fr
'
' Enumere les valeurs numériques des formats
' pour utilisation ultérieure avec par exemple
'
'      ODateTime = oDocument.createInstance
("com.sun.star.text.TextField.DateTime")
'      oDateTime.IsDate=True
'      oDateTime.NumberFormat = formatNum
'
'
' macro a utiliser dans un document Writer
'

oDocument = ThisComponent
oFormats = oDocument.getNumberFormats()
oText = oDocument.Text
oViewCursor = oDocument.CurrentController.getViewCursor()
oTextCursor = oText.createTextCursorByRange(oViewCursor.getStart
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

61/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
( ))
on error resume next

    imax=200
    for i=0 to imax
        chaine=""
        oFormat=oFormats.getbykey(i)
        chaine=cstr(i)+chr(9)+chr(9)+oFormat.formatstring+chr
(10)
        oText.insertString(oTextCursor, chaine, FALSE)
    next i
end sub
```

VI.2 Convertir et exporter des documents OOo en d'autres formats.

```
'Copyright (C) 2003 Laurent Godard
'ooo@godard.no-ip.org
```

```
dim FilterName as string
dim ExtDESt as string
dim FichDest as string
```

```
Sub Exporte(fichier, typedoc, code)
```

```
    call DefinirFiltre(fichier, typedoc, code)
    call TransformeDocument(fichier, TypeDoc, code)
```

```
End Sub
```

VI.2.1 Définition de filtre en fonction du suffixe OO (sxc, stc)

```
sub DefinirFiltre(fichier, typedoc, code)
    'Determine quel type d'export faire
    ExtSRC=right(fichier, 3)
    fichdest=fichier
    Select case ExtSRC
        case "sxc", "stc", "sdc":
            select case typedoc
                case "PDF":
FilterName="calc_pdf_Export"
ExtDESt=".pdf"
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

62/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```

                                case "MSXP":
                                    FilterName="MS Excel
97"
                                    ExtDESt=".xls"
                                case "MS95":
                                    FilterName="MS Excel
95"
                                    ExtDESt=".xls"
                                case "MS50":
                                    FilterName="MS Excel
5.0/95"
                                    ExtDESt=".xls"
                                end select
                                case "sxw","vor","stw","sdw":
                                    select case typedoc
                                        case "PDF":
FilterName="writer_pdf_Export"
                                    ExtDESt=".pdf"
                                case "MSXP":
                                    FilterName="MS Word
97"
                                    ExtDESt=".doc"
                                case "MS95":
                                    FilterName="MS Word
95"
                                    ExtDESt=".doc"
                                case "MS60":
                                    FilterName="MS
WinWord 6.0"
                                    ExtDESt=".doc"
                                case "RTF":
                                    FilterName="Rich Text
Format"
                                    ExtDESt=".rtf"
                                case "HTML":
                                    FilterName="HTML
(StarWriter) "
                                    ExtDESt=".html"
                                    FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "XHTML":
                                    FilterName="XHTML
plus MathML File"
                                    ExtDESt=".xhtml"
                                    FichDest=
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

63/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
TrouveNomRep (fichier, code, typedoc)
    case "TEX":
        FilterName="Latex
File"
        ExtDESt=".tex"
        FichDest=
TrouveNomRep (fichier, code, typedoc)
    end select
    case "sxi", "sti", "sdd", "sdp":
        select case typedoc
            case "PDF":
FilterName="impress_pdf_Export"
                ExtDESt=".pdf"
            case "MSXP":
                FilterName="MS
Powerpoint 97"
                ExtDESt=".ppt"
            case "JPG":
FilterName="image/jpeg"
                ExtDESt=".jpg"
                FichDest=
TrouveNomRep (fichier, code, typedoc)
            case "PNG":
FilterName="image/png"
                ExtDESt=".png"
                FichDest=
TrouveNomRep (fichier, code, typedoc)
            case "TIFF":
FilterName="image/tiff"
                ExtDESt=".tif"
                FichDest=
TrouveNomRep (fichier, code, typedoc)
            case "SVG":
FilterName="image/svg+xml"
                ExtDESt=".svg"
                FichDest=
TrouveNomRep (fichier, code, typedoc)
            case "EPS":
FilterName="application/postscript"
                ExtDESt=".eps"
                FichDest=
TrouveNomRep (fichier, code, typedoc)
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

64/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```

                                case "SWF":
FilterName="impress_flash_Export"
                                ExtDESt=".swf"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                end select
                                case "sxd", "std", "sda", "sdd":
                                select case typedoc
                                case "PDF":
FilterName="draw_pdf_Export"
                                ExtDESt=".pdf"
                                case "JPG":
FilterName="image/jpeg"
                                ExtDESt=".jpg"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "PNG":
FilterName="image/png"
                                ExtDESt=".png"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "TIFF":
FilterName="image/tiff"
                                ExtDESt=".tif"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "SVG":
FilterName="image/svg+xml"
                                ExtDESt=".svg"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "EPS":
FilterName="application/postscript"
                                ExtDESt=".eps"
                                FichDest=
TrouveNomRep (fichier, code, typedoc)
                                case "SWF":
FilterName="draw_flash_Export"
                                ExtDESt=".swf"
                                FichDest=
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

65/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
TrouveNomRep (fichier, code, typedoc)
    end select
end select
end sub
```

VI.2.2 Transformer le document

```
sub TransformeDocument (fichier, TypeDoc, code)
    on error resume next
    dim args()
    'dim args(0) as new com.sun.star.beans.PropertyValue
    'args(0).Name = "Hidden"
    'args(0).Value = True
    dim oDesktop, oDocument as object
    dim surl as string
    oDesktop=createUnoService("com.sun.star.frame.Desktop")
    surl="file:/// " + CorrigeNom(fichier)
    oDocument=oDesktop.loadComponentFromURL(surl, "_blank", 0, args())
    on error goto fin
    fichdest=CorrigeNom(fichdest)
    select case typedoc
        'graphical exports
        case "JPG", "PNG", "TIFF", "EPS", "SVG":
            oGraphic=CreateUnoService
("com.sun.star.drawing.GraphicExportFilter")
            dim args3(1) as new com.sun.star.beans.PropertyValue
            for i=0 to odocument.drawpages.getcount()-1
                opage=odocument.drawPages(i)
                ographic.setSourceDocument(opage)
                args3(0).Name = "URL"
                nom=opage.name
                args3(0).Value = "file:///"+RemplaceChaine
(fichdest, " ", "%20")+"-"+RemplaceChaine(nom, " ", "")+ ExtDest
                args3(1).Name = "MediaType"
                args3(1).Value = filtername
                oGraphic.filter(args3())
            next i
        'all other exports
        case else:
            dim args2(1) as new com.sun.star.beans.PropertyValue
            args2(0).Name = "InteractionHandler"
            args2(0).Value = ""
            args2(1).Name = "FilterName"
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

66/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
args2(1).Value = FilterName
NomFichier = "file:///"+RemplaceChaine(fichdest, " ", "%
20")+ ExtDest
odocument.storeToURL(NomFichier, args2())
end select
oDocument.dispose()
exit sub
fin:
on error resume next
oDocument.dispose()
end sub
```

```
function CorrigeNom(chaine as string) as string
'Remplace les antislash
chaine=remplacechaine(chaine, "\", "/")
'Remplace les deux points
chaine=remplacechaine(chaine, ":", "|")
CorrigeNom=chaine
end function
```

```
function TrouveNomRep(chemin as string, code as string, typedoc as string) as
string
i=instr(1, chemin, "/temp/"+code+"/")+len("/temp/"+code+"/")-1
j=len(chemin)+1
nomfichier=mid(chemin, i, j-i)
TrouveNomRep=mid(chemin, 1, i)+typedoc+nomfichier
end function
```

VI.2.3 Remplacer une chaîne par une autre

Général – Remplace une chaîne de caractère par une autre

Auteur : Laurent Godard

Révision : 03

Amélioration : Laurent Godard et Bernard Marcelly 6 Août 2003

La macro traite les chaînes de plus de 32767 caractères.

```
Function RemplaceChaine(ByVal chaine As String, src As String, dest As
String, casse As Boolean)
' fournit une chaîne dont toutes les occurrences de src ont été remplacées
par dest
'casse = true pour distinguer majuscules/minuscules, = false sinon
Dim lsrc As Integer, i As Integer, kas As Integer
dim limite as string
limite=""
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

67/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
kas = iif(casse, 0, 1)
lsrc = len(src)
i = instr(1, chaine, src, kas)
while i<>0
  while i<0
    limite=limite+left(chaine,32000)
    chaine=mid(chaine,32001)
    i=instr(1, chaine, src, kas)
  wend
  ' ici i est toujours positif non nul
  if i>1 then
    limite = limite + Left(chaine, i-1) +dest
  else ' ici i vaut toujours 1
    limite = limite +dest
  endif
  ' raccourcir en deux temps car risque : i+src > 32767
  chaine = Mid(chaine, i)
  chaine = Mid(chaine, 1+lsrc)
  i = instr(1, chaine, src, kas)
wend
RemplaceChaine = limite + chaine
End Function
```

Préparé par - Société Guidella- confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

68/186

VII L'API OpenOffice.org (presque) sans peine

VII.1 Caractéristiques du Basic OOo

VII.1.1 Enregistreur de macro

A partir de la version 1.1 de OpenOffice, un enregistreur de macro est disponible. Il vous permet de répéter une séquence de manipulations effectuées avec l'interface utilisateur de Writer ou Calc.

Notez que les macros obtenues ne fonctionnent pas dans les versions 1.0.x de OpenOffice.

Le code généré par l'enregistreur ne vous aidera pas à créer des macros vous-même. Il utilise un interface particulière (la fonction dispatch) qui sert à manipuler l'interface utilisateur, tout à fait adapté au but (répéter une séquence utilisateur) mais assez lourde. De plus le code généré n'est pas optimisé.

Après avoir acquis une certaine connaissance du langage macro et de l'API, l'enregistreur de macro pourra vous aider à effectuer des fonctions réalisables par l'interface utilisateur, dont vous ne trouvez pas l'équivalent API.

VII.1.2 Accès aux macros

1. Par le menu Outils / Macros... vous obtenez un panneau comportant une arborescence sur la partie gauche. Les feuilles de l'arborescence sont des modules. Chaque module peut contenir plusieurs macros.

L'arbre *soffice* comporte plusieurs bibliothèques (anglais: libraries), elles-mêmes composées de modules. Il existe de nombreuses macros pré-définies dans cet arbre. Leur lecture est ardue, mais instructive. La bibliothèque *standard* de *soffice* reçoit les macros qui seront accessibles par tous les documents OpenOffice. Chaque module de la bibliothèque *standard* est dans un fichier *.xba du répertoire OpenOffice.org/user/basic/Standard.

Chaque document ouvert possède un arbre à son nom, avec une bibliothèque *standard*. La bibliothèque *standard* d'un document reçoit les macros spécifiques à ce document. Chaque module fait l'objet d'un document xml intégré dans le fichier du document.

Si plusieurs documents sont ouverts en même temps, l'arborescence vous donne accès aux macros de tous les documents ! Faites très attention à définir votre macro dans le bon document !

Pour écrire une macro utilisez l'éditeur de macro intégré à OpenOffice, ou un éditeur externe, mais n'utilisez pas OpenOffice Writer, qui n'est pas adapté à cela.

Dans l'éditeur de macros vous bénéficiez d'une coloration syntaxique très utile, que je ne reproduit pas ici. Je vous conseille de copier/coller les exemples dans l'éditeur de macros et de les exécuter en pas à pas.

VII.1.3 Esthétique et lisibilité

Ce document utilise des styles spécifiques pour les instructions de macro :

- style de paragraphe : ligneMacro
- style de caractère : InstructionMacro
- style de caractère : InstructionMacroChangement
- style de caractère : *NomdeVariable*

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Adaptez éventuellement ces styles si vous avez des difficultés à lire ou imprimer, tout le document suivra.

La lecture de ce document sur écran permet de sauter directement au sujet qui vous intéresse grâce au navigateur de OpenOffice ou aux liens hypertexte.

Basic OOO ne fait pas la distinction entre lettres majuscules et minuscules. Y compris pour les noms de variables et les noms réservés. Mais en utilisant majuscule et minuscule, c'est plus lisible. Par contre certains éléments de l'API doivent être écrits avec les majuscules et minuscules indiquées; on le signalera dans ce document.

Vous pouvez insérer un nombre quelconque d'espaces entre deux mots Basic. Les lignes vides sont aussi ignorées par Basic.

Un commentaire commence avec `Rem` (pas obligatoirement en début de ligne) et se termine à la fin de la ligne. Une apostrophe simple `'` est équivalente à un `Rem`.

Dans les exemples fournis, j'utiliserai en général des variables avec un nom français pour bien les distinguer des primitives du langage Basic et démystifier l'aspect " formule magique ". Je trouve cela plus clair que la méthode de codage employée dans le guide Basic qui consiste à utiliser un nom anglais précédé d'une lettre préfixe (oCell pour : objet Cellule)

a) Définitions de variables

Conseils :

Insérez au tout début de votre module de macro l'instruction

```
Option Explicit
```

Grâce à cette instruction, vous aurez un message d'erreur pour toute variable non définie, ou mal orthographiée.

Définissez chaque variable en précisant son type explicitement. Donnez un nom significatif à vos variables importantes. C'est un tout petit peu plus long à écrire, mais bien plus clair, et vous réduirez le temps passé à chercher les erreurs de programmation.

Exemple

```
OPTION EXPLICIT
```

```
' variables communes au module entier  
Dim MonDocument As Object, LesFeuilles As Object  
Dim MaFeuille As Object
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

70/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Sub LesCompositeurs
```

```
Dim cellIndex As Object, cellEgale As Object ' variables locales  
Dim y1 As Integer, x2 As Integer ' variables locales au sous-programme  
rem -- ici les instructions de la subroutine --  
End Sub
```

Note :

Si vous déclarez plusieurs variables avec le même Dim, répétez le type pour chacune. Sinon les variables sans type sont implicitement des Variant (le type est défini à l'exécution).

```
Dim nombre3, nombre4 As Long rem nombre3 est un Variant
```

Ceci peut avoir des effets bizarres, un Variant étant initialisé à " nul ".

VII.1.4 Trouver le document

On initialise une variable objet pour la faire pointer sur le document à manipuler.

a)Le document en cours

En général le document sur lequel la macro va travailler est le document dont la fenêtre est en avant-plan (elle a le " focus "). L'initialisation se fait ainsi :

```
Dim MonDocument As Object  
MonDocument = ThisComponent
```

b)Accéder à un autre document existant

Pour pointer sur un document existant, pas obligatoirement affiché :

```
Dim MonDocument As Object  
Dim AdresseDoc As String  
Dim PropFich()  
  
AdresseDoc = "file:///home/testuser/work/undocument.sxc"  
  
MonDocument = StarDesktop.LoadComponentFromURL(AdresseDoc, "_blank", 0, PropFich)
```

Vous pouvez utiliser la plupart des lignes sans comprendre. Vous avez seulement à adapter la ligne qui définit *AdresseDoc*. Cette adresse est sous forme d'une URL. Pour les utilisateurs Windows voici l'équivalence :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

71/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
C:\Mes Documents\tata.sxc file:///C:/Mes%20Documents/tata.sxc
```

Basic fournit une commande pour convertir une adresse Windows en URL, exemple :

```
AdresseDoc = convertToURL("C:\Documents and Settings\Arthur\Mes documents\Ma  
Petite Doc.sxw")
```

Notez que l'URL peut aussi désigner une adresse sur le réseau local.

c)Créer un nouveau document

La méthode est celle du chapitre précédent, mais on utilise une adresse magique, qui dépend du type de document ; exemple pour un document Calc :

```
Dim MonDocument As Object  
Dim AdresseDoc As String  
Dim PropFich()  
  
AdresseDoc = "private:factory/scalc"  
MonDocument = StarDesktop.LoadComponentFromURL(AdresseDoc, "_blank", 0, PropFich)
```

Type de nouveau document	Adresse magique
Texte Writer	private:factory/swriter
Tableur Calc	private:factory/scalc
Dessin Draw	private:factory/sdraw
Présentation Impress	private:factory/simpress
Editeur de formules Math	private:factory/smath

Ces documents suivent le modèle par défaut en vigueur pour votre configuration de OpenOffice.

d)Créer un nouveau document à partir d'un modèle

Pour créer un nouveau document conforme à un modèle particulier, mettre dans *AdresseDoc* le chemin d'accès au modèle.

```
AdresseDoc = convertToURL("C:\Mes modeles\Doc2colonnes.stw")
```

C'est très pratique pour disposer immédiatement d'un ensemble de styles à votre goût, de feuilles Calc prédéfinies, d'un en-tête ou un pied de page, d'une mise en page en colonnes, ou d'un texte initial qui sera complété par la macro.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

72/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

e) Quelques détails complémentaires

e.a) StarDesktop

L'objet `StarDesktop` simplifie les choses. Sans lui on écrirait :

```
Dim MonBureau As Object, MonDocument As Object
Dim AdresseDoc As String
Dim PropFich()

MonBureau = createUnoService("com.sun.star.frame.Desktop")

AdresseDoc = "private:factory/scalc"
MonDocument = MonBureau.LoadComponentFromURL(AdresseDoc, "_blank", 0, PropFich)
```

e.b) ThisComponent

L'objet `ThisComponent` simplifie les choses.

Si la macro est dans un module du document, seul ce document sera pris en compte même si on appelle la macro depuis un autre document ouvert. Si la macro est dans un module de *soffice*, `ThisComponent` représente le document à partir duquel on appelle la macro.

On voit souvent la séquence suivante (francisée) :

```
MonBureau = createUnoService("com.sun.star.frame.Desktop")
MonDocument = MonBureau.getCurrentComponent()
MonTexte = MonDocument.Text
```

Ceci **n'est pas** l'équivalent de

```
MonDocument = ThisComponent
MonTexte = MonDocument.Text
```

mais de :

```
MonDocument = StarDesktop.CurrentComponent
MonTexte = MonDocument.Text
```

Si vous exécutez la première séquence de macro dans l'EDI (la fenêtre d'affichage des macros), la troisième instruction va tomber en faute : *propriété ou méthode introuvable*. Parce que `getCurrentComponent` renvoie la fenêtre de l'EDI, et non pas le document sur lequel la macro s'exerce.

La deuxième séquence fonctionne aussi dans l'EDI, preuve que `ThisComponent` est plus malin. Si vous voulez exécuter vos macros dans l'EDI, et profiter de ses

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

73/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

facilités de débogage, utilisez donc `ThisComponent`.

e.c) Arguments de `LoadComponentFromURL`

`LoadComponentFromURL` sert à charger un document; il sera affiché dans un cadre (une fenêtre).

- Le deuxième argument de `LoadComponentFromURL` est le nom du cadre cible. Si un cadre (une fenêtre) ayant ce nom existe, le document sera affiché dedans. Il existe certains noms réservés (voir API), mais vous n'en aurez probablement pas besoin.
- Le troisième argument de `LoadComponentFromURL` est en général inutilisé (valeur zéro).
- Le quatrième argument contient les propriétés souhaitées du fichier, voir ci-dessous.

e.d) Propriétés pour ouvrir un document

Vous pouvez préciser certaines informations, détaillées dans l'API, voir le service `MediaDescriptor`. Chaque information possède une structure de type Propriété; chaque propriété a un nom (Name) et une valeur (Value). Si vous n'avez pas d'information particulière à fournir, vous devez au moins fournir un élément vide, comme indiqué [plus haut](#).

Comme exemple, ouverture d'un document protégé par un mot de passe :

```
Dim MonDocument As Object
Dim AdresseDoc As String
Dim PropFich(0) As New com.sun.star.beans.PropertyValue

AdresseDoc = "file:///home/testuser//work/Finances.sxc"

PropFich(0).Name = "Password"
PropFich(0).Value = "Arthur1987"
MonDocument = StarDesktop.LoadComponentFromURL(AdresseDoc, "_blank", 0, PropFich() )
```

Respectez les majuscules-minuscules pour le nom de la propriété. Remarquez aussi que nous définissons `PropFich` comme un vecteur à un élément, et que les parenthèses deviennent obligatoires dans la dernière ligne. Pour 3 propriétés simultanées, `PropFich` serait défini comme :

```
Dim PropFich(2) As New com.sun.star.beans.PropertyValue
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Propriétés principales pour ouvrir un document

Propriété	Signification
<code>AsTemplate</code>	booléen; utile si on travaille sur un modèle. valeur true : (par défaut) un nouveau document sans titre est créé à partir du modèle désigné dans l'URL ; valeur false : le modèle est ouvert pour édition. <u>Remarque</u> Avec <code>AsTemplate = true</code> et une <i>AdresseDoc</i> correspondant à un document ordinaire (pas un modèle) on obtient un nouveau document basé sur celui en référence.
<code>Hidden</code>	booléen ; valeur true pour charger le document sans le rendre visible
<code>Password</code>	chaîne de caractères ; mot de passe pour ouvrir le document encrypté
<code>ReadOnly</code>	booléen ; valeur true pour ouvrir le document en lecture seule (pour l'utilisateur, pas pour l'API)

VII.1.5 Sauver le document

D'abord, le document a-t-il été modifié ? Ce test le détermine :

```
if MonDocument.IsModified then
    rem mettre ici les instructions pour sauver le document
end if
```

Si vous avez modifié un document existant, le sauver est tout simple :

```
MonDocument.Store
```

Si vous avez ouvert un nouveau document, vous devez préciser sous quel nom et à quelle adresse il doit être sauvé; on indique pour cela une URL.

Vous pouvez aussi préciser certaines informations, détaillées dans l'API, voir le service `MediaDescriptor`. Chaque information possède une structure de type `Propriété`; chaque propriété a un nom (`Name`) et une valeur (`Value`). Si vous n'avez pas d'information particulière à fournir, vous devez au moins fournir un élément vide.

Cet exemple écrase tout document du même nom à la même adresse :

```
Dim MonDocument As Object
Dim NomduFichier As String
Dim PropFich()
```

Préparé par - Société Guidella - confidentiel

`coursMacro.sxw`

Imprimé le 28 décembre 2004

75/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
NomduFichier = "file:///C:/Travail/___blabla.sxw"  
MonDocument = ThisComponent
```

```
MonDocument.storeAsURL(NomduFichier, PropFich)
```

Pour ne pas écraser un document existant, on utilisera la propriété `Overwrite` :

```
Dim MonDocument As Object  
Dim NomduFichier As String  
Dim PropFich(0) As New com.sun.star.beans.PropertyValue
```

```
NomduFichier = "file:///C:/Travail/___blabla.sxw"  
MonDocument = ThisComponent  
PropFich(0).Name = "Overwrite"  
PropFich(0).Value = false  
MonDocument.storeAsURL(NomduFichier, PropFich() )
```

Il y aura une exception (à traiter par programmation) si le document existe déjà.

Respectez les majuscules-minuscules pour le nom de la propriété. Remarquez aussi que nous définissons *PropFich* comme un vecteur à un élément, et que les parenthèses deviennent obligatoires dans la dernière ligne. Pour 3 propriétés simultanées, *PropFich* serait défini comme :

```
Dim PropFich(2) As New com.sun.star.beans.PropertyValue
```

Propriétés principales pour enregistrer le document

Propriété	Signification
Author	une chaîne de caractères pour satisfaire votre ego
Version	un nombre entier
Password	chaîne de caractères ; mot de passe pour ouvrir le document encrypté

a) Enregistrer sous / Enregistrer une copie sous

La méthode `storeAsURL` réalise l'équivalent du menu Fichier / Enregistrer sous...

Si vous avez ouvert un fichier TOTO et que vous l'enregistrez sous TATA, vous travaillez maintenant dans le document TATA.

Par contre, la méthode `storeToURL` réalise une copie du document en cours, et seulement cela. Ainsi, travaillant sur TOTO, vous faites une copie appelée TATA et continuez sur TOTO. Utile pour des sauvegardes régulières.

La méthode `storeToURL` utilise la même syntaxe que `storeAsURL`.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

76/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b) Fermer le document

La méthode `Dispose` ferme le document, et donc la fenêtre qui l'affichait.

Exemple :

```
Dim UnDocument As Object
Dim AdresseDoc As String

AdresseDoc = "file:///C:/Textes/blabla.sxw"
UnDocument = StarDesktop.LoadComponentFromURL(AdresseDoc, "_blank", 0, PropFich)
print "document ouvert !"
UnDocument.Dispose
```

VII.1.6 Tableur Calc

a) Trouver la feuille

Avant de travailler sur des cellules, vous devez préciser dans quel document elles se trouvent, et dans quelle feuille.

a.a) Une feuille existante

Chaque feuille de Calc comporte un onglet. Le nom des onglets est par défaut, dans la version localisée française, **Feuille1**, **Feuille2**, etc.

Avec l'API on accède ainsi à une feuille :

```
Dim MonDocument As Object
Dim MaFeuille As Object, LesFeuilles As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
print "Nombre de feuilles : "; LesFeuilles.Count
MaFeuille = LesFeuilles(0) ' Première feuille
```

`Sheets` fournit l'ensemble des feuilles du document. `Count` donne le nombre de feuilles existantes. Chaque feuille est accessible par un index.

On aurait pu se passer de la variable *LesFeuilles* en écrivant la dernière ligne :

```
MaFeuille = MonDocument.Sheets(0) ' Première feuille du tableur
```

Attention

Pour l'API les pages sont numérotées à partir de zéro, dans l'ordre affiché par l'interface Calc. Ainsi si l'utilisateur change l'ordre des feuilles, la macro écrira obstinément sur la feuille de rang n.

Une meilleure méthode consiste à récupérer la feuille ayant un nom donné :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

77/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object
```

```
MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByName("Oeuvres")
```

On peut tester si une feuille d'un nom donné existe :

```
if LesFeuilles.hasByName("Oeuvres") then
    rem la feuille existe bien
end if
```

a.b) Créer, renommer, dupliquer, supprimer une feuille

Créer une feuille vierge

La méthode `insertNewByName` insère une nouvelle feuille vierge à une position donnée parmi les feuilles du document, et lui donne un nom.

```
Dim MonDocument As Object
Dim NouvelleFeuille As Object, LesFeuilles As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
' insérer la feuille vierge XXX après la deuxième feuille
LesFeuilles.insertNewByName("Totalisation", 2)
NouvelleFeuille = LesFeuilles.getByName("Totalisation")
```

Renommer une feuille

Renommer une feuille revient à changer la valeur de la propriété `Name` de la feuille.

```
MaFeuille = LesFeuilles.getByName("Mois")
MaFeuille.Name = "Fevrier"
```

Déplacer une feuille

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByName("Fevrier")
' Déplacer en deuxième position
LesFeuilles.MoveByName("Fevrier", 1)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

78/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Dupliquer une feuille

```
Dim MonDocument As Object, LesFeuilles As Object  
  
MonDocument = ThisComponent  
LesFeuilles = MonDocument.Sheets  
' Créer une copie en troisième position  
LesFeuilles.CopyByName("Fevrier", "Mars", 2)
```

La nouvelle feuille Mars est la copie exacte de la feuille Fevrier, avec son contenu.

Supprimer une feuille

```
Dim MonDocument As Object, LesFeuilles As Object  
  
MonDocument = ThisComponent  
LesFeuilles = MonDocument.Sheets  
LesFeuilles.removeByName("Fevrier")
```

a.c) La feuille visible par l'utilisateur

La feuille visible par l'utilisateur se trouve ainsi :

```
MaFeuille = MonDocument.CurrentController.ActiveSheet  
print "Le nom de la feuille active est : "; MaFeuille.Name
```

Et pour rendre visible une autre feuille :

```
MaFeuille = MonDocument.Sheets(1)  
MonDocument.CurrentController.ActiveSheet = MaFeuille
```

b) Trouver la cellule

Pour manipuler une cellule, je vais définir une variable intermédiaire qui pointera sur la cellule.

```
Dim MonDocument As Object, LesFeuilles As Object  
Dim MaFeuille As Object, UneCellule As Object  
  
MonDocument = ThisComponent  
LesFeuilles = MonDocument.Sheets  
MaFeuille = LesFeuilles.getByName("Oeuvres")  
UneCellule = MaFeuille.getCellByPosition(3,5) ' cellule D6
```

Les paramètres de `getCellByPosition(x,y)` sont :

- x = rang de la colonne, avec 0 pour A, 1 pour B, 25 pour Z, etc
- y = rang de la ligne, avec 0 pour la ligne 1, 1 pour la ligne 2, etc.

Nous retrouverons cette méthode de positionnement dans d'autres fonctions.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

79/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c) Zone de cellules

c.a) Manipuler une zone de cellules

Il existe un type d'objet " zone de cellules ". On définit une zone de cellules de différentes manières :

```
Dim UneZone As Object
```

```
Rem zone B1 à C6 dans la 4 ème feuille du document Calc
```

```
UneZone = GetCellRange(MonDocument, 3, 1, 0, 2, 5)
```

```
Rem zone dans une feuille particulière
```

```
UneZone = MaFeuille.getCellRangeByName("B1:C6")
```

```
UneZone = MaFeuille.getCellRangeByName("$B2") ' zone 1 cellule
```

```
UneZone = MaFeuille.getCellRangeByName("NomdeZone")
```

```
UneZone = MaFeuille.getCellRangeByPosition(1,0,2,5) ' zone B1:C6
```

Pointer sur une cellule dans la zone :

```
UneCellule = UneZone.getCellByPosition(3,5)
```

ici les coordonnées, toujours numérotées à partir de zéro, sont relatives au coin haut-gauche de la zone.

Si la zone de cellule se réduit à une seule cellule, on a en fait un pointeur vers une cellule :

```
UneCellule = MaFeuille.getCellRangeByName("C13")
```

Remarque :

Nous avons vu deux utilisations de `getCellByPosition` :

```
1. MaFeuille.getCellByPosition(3,5)
```

```
2. UneZone.getCellByPosition(3,5)
```

Il est utile de savoir qu'il ne s'agit pas de la même fonction, mais de deux fonctions similaires.

Dans le premier cas, la fonction est spécifique à un objet de type feuille, dans le second cas la fonction est spécifique à un objet de type zone de cellule. Ce concept est typique de la conception orientée objet : chaque type d'objet possède des variables, des procédures et des fonctions qui lui sont propres; il est d'usage de reprendre le même nom entre deux types d'objets, quand les fonctions sont similaires. Par conséquent, pour lever l'ambiguïté on qualifie le nom de la fonction ou procédure ou variable, avant le nom de l'objet.

Si vous parcourez un jour la documentation de l'API vous trouverez nombre de fonctions et procédures synonymes. Vous ne pouvez pas les utiliser sans savoir à quel type d'objet elles appartiennent.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

80/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c.b) Zone de cellules sélectionnées par l'utilisateur

L'utilisateur peut sélectionner une zone de cellules, puis appeler une macro. La séquence suivante retrouve la zone sélectionnée.

```
Dim MonDocument As Object
Dim UneZone As Object

MonDocument = ThisComponent
UneZone = MonDocument.CurrentSelection
```

L'utilisateur pourrait sélectionner plusieurs zones à la fois, au lieu d'une seule zone. Dans ce cas, `CurrentSelection` fournit une liste des zones sélectionnées. Nous pouvons le savoir ainsi :

```
Dim MonDocument As Object
Dim UneZone As Object, LesSelections As Object
Dim selx As Integer

MonDocument = ThisComponent
LesSelections = MonDocument.CurrentSelection
if LesSelections.supportsService("com.sun.star.sheet.SheetCellRanges") then
  for selx = 0 to LesSelections.Count - 1
    UneZone = LesSelections(selx)
    ' ici, utiliser une des zones sélectionnées
  next
else
  ' une seule zone ou une cellule
end if
```

On utilise alors une boucle pour trouver successivement ces zones. Le nombre de zones est dans la propriété `Count`.

c.c) Sélectionner visiblement une zone de cellules

Le but est de sélectionner une zone par macro, et d'afficher cette sélection à l'utilisateur. Il faut une instruction spéciale, qui utilise en argument un pointeur vers une cellule ou une zone de cellules :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object, UneZone As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByName("Feuille3")
UneZone = MaFeuille.getCellRangeByName("B2:F6")
MonDocument.CurrentController.Select(UneZone)
```

Si vous avez exécuté cette macro dans l'EDI, mettez en avant-plan la fenêtre Calc pour voir la sélection du mot.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

81/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Notez aussi que la feuille concernée est automatiquement affichée.

d) Retrouver les coordonnées d'une cellule

Maintenant que nous savons obtenir une zone de cellules, et rechercher une cellule dans la zone, il est parfois utile de connaître les coordonnées absolues d'une zone ou d'une cellule.

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object, UneZone As Object, UneCellule As Object
Dim CoordCellule As Object, CoordZone As Object
Dim FeuilleCourante As Integer, ColonneCourante As Integer
Dim LigneCourante As Integer, FeuilledeZone As Integer
Dim PremiereColonneZone As Integer, DerniereColonneZone As Integer
Dim PremiereLigneZone As Integer, DerniereLigneZone As Integer

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByName("Feuille3")
UneZone = MaFeuille.getCellRangeByName("B1:G6")
UneCellule = UneZone.getCellByPosition(2,4) ' position dans la zone

CoordCellule = UneCellule.CellAddress
FeuilleCourante = CoordCellule.Sheet
ColonneCourante = CoordCellule.Column
LigneCourante = CoordCellule.Row

CoordZone = UneZone.RangeAddress
FeuilledeZone = CoordZone.Sheet
PremiereColonneZone = CoordZone.StartColumn
DerniereColonneZone = CoordZone.EndColumn
PremiereLigneZone = CoordZone.StartRow
DerniereLigneZone = CoordZone.EndRow
```

Remarquez la similarité des instructions pour une zone et pour une cellule.

Ici encore, les coordonnées obtenues sont numérotées à partir de zéro.

e) Connaître le type de contenu de la cellule

En supposant avoir écrit les lignes des chapitres précédents, on peut maintenant lire et écrire dans la cellule. Mais les instructions dépendent du genre d'information à manipuler.

Une cellule a un parmi quatre types possibles de contenu :

- cellule vide

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

82/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- valeur numérique
- un texte
- une formule (qui peut fournir une valeur numérique ou un texte)

`Type` fournit le type de contenu de la cellule, sous forme d'une valeur numérique.

Pour être indépendant de l'implémentation (qui peut évoluer), chaque type doit être désigné avec une constante prédéfinie, dont j'explique le mode d'emploi au chapitre [Les horribles constantes](#).

Cet exemple liste les constantes et montre comment les utiliser pour connaître le contenu d'une cellule :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object, UneCellule As Object
Dim ContTexte As Integer, ContValeur As Integer
Dim ContFormule As Integer, ContVide As Integer

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByname("Feuille2")
UneCellule = MaFeuille.getCellbyposition(2,9) ' cellule C10

rem constantes des types de contenu
ContVide= com.sun.star.table.CellContentType.EMPTY
ContValeur= com.sun.star.table.CellContentType.VALUE
ContTexte= com.sun.star.table.CellContentType.TEXT
ContFormule= com.sun.star.table.CellContentType.FORMULA

Select Case UneCellule.Type
Case ContVide
    Print "case vide !"
Case ContFormule
    Print "Formule"
Case ContValeur
    Print "Valeur"
Case ContTexte
    Print "Texte"
End Select
```

f) Valeur numérique dans la cellule

Lire et écrire une valeur numérique dans la cellule :

```
Dim x2 As Integer
x2 = UneCellule.Value
x2 = x2 + 17
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

83/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
UneCellule.Value = x2
```

Attention : `Value` vous donne le résultat numérique de la formule qui se trouve dans la cellule. Si la cellule contient un texte, ou si la formule donne un texte, `getValue` donne zéro.

g) Texte dans la cellule

Lire et écrire une chaîne de caractère dans la cellule :

```
Dim tx3 As String  
tx3 = UneCellule.String  
UneCellule.String = "Bonsoir"
```

La dernière instruction écrase tout texte existant dans la cellule. On peut faire mieux, mais c'est plus compliqué. Cependant le principe se retrouve pour écrire du texte dans Writer ou ailleurs.

g.a) Le curseur d'écriture de texte dans la cellule

Pour écrire un texte dans un texte existant dans une cellule, nous avons besoin d'un curseur d'écriture. Ce curseur indique le point d'insertion du nouveau texte, et avance à chaque insertion.

La séquence suivante insère un texte à droite de celui existant dans la cellule :

```
Dim MonCurseur As Object  
  
tx3 = "Bonjour"  
MonCurseur = uneCellule.createTextCursor  
UneCellule.insertString(MonCurseur, tx3, false)  
UneCellule.insertString(MonCurseur, " les amis", false)
```

Le curseur est positionné à sa création à la fin du texte existant dans la cellule. La gestion du curseur d'écriture est décrite au chapitre suivant.

Dans la procédure `insertString` le troisième argument signifie :

- `false` = insérer dans le texte
- `true` = écraser dans le texte

g.b) Caractères spéciaux dans un texte

Même dans une simple cellule on peut insérer une fin de paragraphe de texte :

```
Dim FinParagraphe As Integer  
FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK  
UneCellule.insertControlCharacter(MonCurseur, FinParagraphe, false)
```

Il existe une liste d' horribles constantes similaires :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

84/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
PARAGRAPH_BREAK	Insérer ici une marque de fin de paragraphe.
APPEND_PARAGRAPH	Insérer un paragraphe à la fin de celui en cours et se positionner au début du nouveau paragraphe.

g.c) Déplacement du curseur de texte dans une cellule

Pour insérer à un endroit quelconque nous avons besoin de modifier la position du curseur.

```
MonCurseur.goLeft(23, false) ' déplace de 23 caractères vers gauche
MonCurseur.goRight(12, false) ' déplace de 12 caractères vers droite
MonCurseur.gotoStart(false) ' vers début du texte
MonCurseur.gotoEnd(false) ' vers fin du texte
```

Le dernier argument des routines curseur signifie :

- `false` = déplacer le curseur
- `true` = déplacer le curseur en étendant la sélection (comme en glissant avec la souris)

On peut donc sélectionner une zone de plusieurs caractères; ceci fait, on récupère le texte sélectionné ainsi :

```
tx3 = MonCurseur.String
```

Inversement, on peut écraser un texte sélectionné avec un nouveau texte :

```
MonCurseur.String = "Bonsoir"
```

Il existe aussi : `collapseToStart`, `collapseToEnd`, `isCollapsed`, `gotoRange`. Autres informations sur le curseur : voir dans l'API : `XTextCursor` et `XTextRange`

h) Formule dans la cellule

Lire une formule dans la cellule :

```
Dim tx3 As String
tx3 = UneCellule.Formula 'la formule en version anglaise
tx3 = UneCellule.FormulaLocal 'la formule en version localisée
```

Si votre cellule contient `=Maintenant()` vous obtiendrez `=Now()` en version anglaise.

Ecrire une formule dans la cellule, en version localisée :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

85/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim x2 As Integer
x2 = 5
UneCellule.FormulaLocal= "=Arrondi (AutreF.A" + LTrim(Str(x2))+") "
```

La variable numérique x2 est transformée en chaîne de caractère, `LTrim` sert à supprimer l'espace avant le chiffre. Résultat : la formule référence la cellule A5 dans la feuille "AutreF" :

```
=ARRONDI (AutreF.A5)
```

On obtiendrait le même résultat en formulation anglaise :

```
UneCellule.Formula= "=Round (AutreF.A" + LTrim(Str(x2))+") "
```

i) Recalculer les formules des cellules

Voici comment mettre à jour les calculs de formules dans toutes les cellules du tableur :

```
MonDocument.calculateAll
```

Dans un tableur très complexe, on peut se limiter à recalculer les formules qui ne sont pas à jour :

```
MonDocument.calculate
```

Le recalcul est normalement automatique. On peut désactiver l'automatisme pour éviter des calculs inutiles, puis le réactiver quand on l'estime utile.

```
rem désactiver le recalcul automatique
MonDocument.enableAutomaticCalculation(false)
    rem - - effectuer divers travaux - -
rem activer le recalcul automatique
MonDocument.enableAutomaticCalculation(true)
```

Enfin, on peut déterminer si l'automatisme est en fonction :

```
if MonDocument.isAutomaticCalculationEnabled then
    rem le recalcul automatique est actuellement activé
end if
```

j) Modifier le format d'une cellule

Tout ce que nous avons fait jusqu'à maintenant ne change pas le formatage de la cellule. Le texte ou la valeur s'inscrivent avec le formatage existant.

Il existe de très nombreuses possibilités de formatage, voir dans l'API à `CellProperties`. Les plus courantes sont décrites ci-après.

j.a) Style d'une cellule

Le plus rapide pour formater une cellule est d'utiliser un style de cellule qui est

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

déjà défini dans le tableur. La propriété `CellStyle` est une chaîne de caractères qui est le nom du style. On peut lire le nom du style en cours ou changer de style :

```
print UneCellule.CellStyle  
UneCellule.CellStyle= "MonStyleAMoi"
```

Attention

Si vous récupérez le nom du style en cours, pour les styles standards fournis avec OOO vous obtiendrez le nom anglais, même avec une version localisée de OOO. Correspondances pour la version française :

Nom en français	Nom anglais obtenu
Résultat	Result
Résultat2	Result2
Standard	Default
Titre	Heading
Titre1	Heading1

Donc, en cas de doute, lisez le nom du style avec une petite macro.

Deuxième problème potentiel : la chaîne de caractère du style que vous affectez à une cellule doit reproduire exactement le nom du style existant : majuscules, minuscules, accents. Sinon le style de la cellule restera inchangé.

Les possibilités de formatage qui suivent sont spécifiques à la cellule concernée, elles ne modifient pas le style de cellule. De plus une interrogation du style de la cellule donnera toujours le même nom, alors que la cellule est en réalité formatée différemment.

Notez aussi que les propriétés de cellule peuvent être appliquées directement à une zone de cellule, par exemple la couleur de fond.

j.b) Couleur de fond de la cellule

```
UneCellule.CellBackColor = RGB(255,255,204) ' jaune pâle  
UneZone.CellBackColor = RGB(255,255,204) ' colorer toute une zone
```

La définition des couleurs est décrite au chapitre [Couleurs](#).

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

87/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

j.c) Couleur de caractère

```
UneCellule.CharColor = RGB(200,0,0) ' rouge
```

La définition des couleurs est décrite au chapitre [Couleurs](#).

j.d) Taille de caractère

La taille des caractères est définie en points

```
UneCellule.CharHeight = 12
```

j.e) Justification horizontale

Encore des [horribles constantes...](#)

```
UneCellule.HoriJustify = com.sun.star.table.CellHoriJustify.CENTER
```

Principales valeurs possibles :

Constante	Signification
STAND ARD	alignement par défaut selon le type du contenu (texte, nombre)
LEFT	Alignement à gauche
CENTE R	Alignement centré
RIGHT	Alignement à droite

j.f) Justification verticale

Encore des [horribles constantes...](#)

```
UneCellule.VertJustify = com.sun.star.table.CellVertJustify.TOP
```

Agrandir la hauteur de la ligne pour mieux voir l'effet. Valeurs possibles :

Constante	Signification
STAND ARD	alignement par défaut
TOP	alignement sur le haut de la cellule
CENTE R	alignement centré en hauteur
BOTTO M	alignement sur le bas de la cellule

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

88/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

j.g) Orientation verticale/horizontale

La propriété `Orientation` est prise en compte seulement si la propriété `RotateAngle` vaut zéro.

Encore des horribles constantes...

```
UneCellule.Orientation = com.sun.star.table.CellOrientation.TOPBOTTOM
```

Valeurs possibles :

Constante	Signification
STANDARD	Horizontal, tout simplement
TOPBOTTOM	Pour lire, pencher la tête à droite pour un texte européen
BOTTOMTOP	Pour lire, pencher la tête à gauche pour un texte européen
STACKED	Chaque lettre est horizontale, les lettres sont placées de haut en bas.

j.h) Orientation à angle quelconque

```
UneCellule.RotateAngle = 15*100
```

La valeur est l'angle en centièmes de degré; l'exemple correspond à un angle de 15°. Une valeur positive correspond au sens trigonométrique (le sens inverse des aiguilles d'une montre).

j.i) Retour à la ligne

Ce formatage est accessible manuellement par : Formater la cellule / onglet alignement / Enchaînements, Renvoi à la ligne. Un texte trop long pour la largeur de la cellule est "replié" sur plusieurs lignes automatiquement.

Cet exemple de code montre comment tester l'état actuel, et le modifier.

```
if UneCellule.IsTextWrapped then  
  print "Le retour à la ligne est actif"  
  UneCellule.IsTextWrapped= false 'désactiver le retour à la ligne'  
else  
  print "Le retour à la ligne est inactif"  
  UneCellule.IsTextWrapped= true 'activer le retour à la ligne'  
end if
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

89/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

k) Formater des zones de cellules

Les possibilités de formatage d'une cellule sont aussi disponibles pour une zone de cellules, et même pour des lignes ou colonnes (voir plus loin), et pour la feuille entière comme dans cet exemple :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles.getByName("BelleFeuille")
MaFeuille.CellBackColor = RGB(255,255,204) ' jaune pâle
MaFeuille.CharColor = RGB(200, 0, 0) ' rouge
```

l) Lignes et colonnes

l.a) Définir un ensemble de colonnes

Un ensemble de colonnes dans une feuille est défini à partir d'une zone de cellules. Cette zone pouvant se réduire à une seule cellule. Les colonnes sont celles qui "traversent" la zone. Reprenant les définitions précédentes, on définit un objet "collection de colonnes" ainsi:

```
Dim MesColonnes As Object, MaColonne As Object
Dim nbColonnes As Long, nbLignes As Long

UneZone = MaFeuille.getCellRangeByName("B1:C6")
MesColonnes = UneZone.Columns

UneCellule = MaFeuille.getCellByPosition(3,5) ' cellule D6
MaColonne = UneCellule.Columns
nbColonnes = UneZone.Columns.Count
nbLignes = UneZone.Rows.Count
```

l.b) Ajouter ou supprimer des colonnes

Les colonnes sont numérotées à partir de zéro.

Ajouter 3 colonnes groupées, dont la première aura le rang 4 dans la collection :

```
MesColonnes.insertByIndex(4,3)
```

Attention, limitation :

La méthode `insertByIndex` refuse d'insérer à la place de la colonne la plus à

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

90/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

droite dans une zone de colonne. Pour y arriver il faut donc se positionner sur cette colonne de droite, et insérer en position zero.

Supprimer 3 colonnes groupées, dont la première a le rang 5 dans la collection :

```
MesColonnes.removeByIndex(5,3)
```

l.c) Optimiser la largeur d'une colonne

Si vous avez dans une colonne des cellules contenant des textes de longueur différentes, il peut être utile d'ajuster la largeur de la colonne à la valeur nécessaire pour le plus long texte.

```
MaColonne.OptimalWidth = TRUE
```

l.d) Imposer une largeur de colonne

La propriété `Width` définit la largeur de la (ou des) colonne(s), en 1/100 de mm

```
MesColonnes.Width = 3000
```

l.e) Ajouter ou supprimer des lignes

Les instructions sont quasiment identiques :

```
MesLignes = UneZone.Rows  
MaLigne = UneCellule.Rows  
rem Ajouter 5 lignes groupées,  
rem dont la première aura le rang 2 dans la collection  
MesLignes.insertByIndex(2,5)  
MesLignes.removeByIndex(5,3)
```

l.f) Imposer une hauteur de ligne

La propriété `Height` définit la hauteur de la (ou des) ligne(s), en 1/100 de mm

```
MesLignes.Height = 1000
```

l.g) Cacher des lignes ou des colonnes

On peut cacher à l'utilisateur une colonne ou une ligne. L'exemple suivant cache les lignes et colonnes de la zone B2:C6.

```
Dim MaFeuille As Object, UneZone As Object  
Dim MesColonnes As Object, MesLignes As Object  
  
MaFeuille = ThisComponent.Sheets(0)  
Unezone = MaFeuille.getCellRangeByName("B2:C6")  
MesColonnes = Unezone.Columns  
MesColonnes.IsVisible = false
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

91/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MesLignes = UneZone.Rows  
MesLignes.IsVisible = false
```

Comme pour la plupart des propriétés, `IsVisible` peut être lue pour savoir si une colonne ou une ligne est cachée. Exemple :

```
UneCellule = MaFeuille.getCellByPosition(3,5) ' cellule D6  
if UneCellule.Rows.IsVisible and UneCellule.Columns.IsVisible then  
    print "La cellule est visible"  
else  
    print "La cellule est cachée"  
end if
```

m) Manipuler des données en tableau

Il est assez pénible de lire/écrire successivement dans plusieurs cellules. Or dans bien des cas, on doit effectuer des traitements sur toutes les cellules d'une zone.

Les instructions `getDataArray` et `setDataArray` permettent d'effectuer les traitements dans une variable tableau et de recopier les valeurs depuis/vers la zone de cellules.

L'exemple qui suit ajoute une valeur numérique au contenu de chacune des cellules de la zone sélectionnée par l'utilisateur. On suppose que les cellules contiennent des nombres ou sont vides.

```
Dim MonDocument As Object  
Dim UneZone As Object  
Dim x1 As Long, y1 As Long  
Dim mesValeurs, vLigne, UnElement  
  
UneZone = ThisComponent.currentSelection  
mesValeurs = UneZone.getDataArray  
for y1 = LBound(mesValeurs) to UBound(mesValeurs)  
    vligne = mesValeurs(y1)  
    for x1 = LBound(vligne) to UBound(vligne)  
        UnElement = vligne(x1) + 2000 + 10*x1 + 100*y1  
        vligne(x1) = UnElement  
    next  
    mesValeurs(y1) = vligne  
next  
UneZone.setDataArray(mesValeurs)
```

Pour la démonstration, le résultat dans chaque cellule reflète la valeur de `x1` et `y1`.

Cet exemple ne fonctionne que si les variables `mesValeurs` et `vLigne` sont de type Variant. Remarquez qu'elles sont déclarées comme des variables simples, alors que l'utilisation les définira comme des tableaux : c'est une possibilité cachée des Variant de OpenOffice Basic.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

92/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Il n'est pas possible de manipuler directement un élément dans *mesValeurs*, on doit utiliser des variables intermédiaires. Les fonctions `LBound` et `UBound` sont expliquées dans l'aide en ligne de Basic.

De manière identique, on va modifier une zone de cellules contenant des textes :

```
Dim UneZone As Object
Dim x1 As Long, y1 As Long
Dim mesValeurs, vLigne, UnElement

UneZone = ThisComponent.currentSelection
mesValeurs = UneZone.getDataArray
for y1 = LBound(mesValeurs) to UBound(mesValeurs)
  vLigne = mesValeurs(y1)
  for x1 = LBound(vLigne) to UBound(vLigne)
    UnElement = vLigne(x1) & " x1=" & x1 & " y1=" & y1
    vLigne(x1) = UnElement
  next
  mesValeurs(y1) = vLigne
next
UneZone.setDataArray(mesValeurs)
```

n) Curseur de cellule

Remarque : ce mécanisme semblerait défectueux, et il est avantageusement remplacé par le positionnement direct sur une cellule.

Avec un curseur de cellule, on peut se déplacer de cellule en cellule.

Méthode	Signification
<code>createCursor</code>	pour balayer toute la feuille
<code>createCursorByRange()</code>	pour balayer une zone de cellules
<code>gotoStart</code>	aller à la première cellule non vide
<code>gotoEnd</code>	depuis une zone non vide, aller à la dernière cellule non vide
<code>gotoOffset()</code>	déplacement par rapport à la position actuelle
<code>gotoPrev</code>	aller à la cellule précédente non protégée (celle à gauche en général)
<code>gotoNext</code>	aller à la cellule suivante non protégée (celle à droite en général)

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

93/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Exemple :

```
Dim MonCurseurCellule As Object, UneCellule As Object

MonCurseurCellule = MaFeuille.createCursor
MonCurseurCellule.gotoStart
MonCurseurCellule.gotoNext
UneCellule = MonCurseurCellule.getCellByPosition(0,0)
UneCellule.String = "Bonjour"

MonCurseurCellule.gotoEnd
rem descendre d'une ligne
MonCurseurCellule.gotoOffset(0, 1)
MonCurseurCellule.getCellByPosition(0,0).String = "Bonsoir"
```

o) Appeler des fonctions Calc dans une macro

L'exemple suivant appelle la fonction ADRESSE (voir aide en ligne de Calc, fonctions de la catégorie Classeur). La macro doit appeler la version anglaise de cette fonction, qui a pour nom ADDRESS.

Cette macro est une fonction qui renvoie une adresse de cellule sous forme de chaîne de caractères, à partir des coordonnées utilisées par l'API. Par exemple avec les coordonnées 2,6 :

- celladdr(2,6,4) renvoie "C7"
- celladdr(2,6,0) renvoie "\$C\$7"

```
Function celladdr(col As Integer, ligne As Integer, absol As Integer) As String
Dim acceder As Object
Dim params As Variant

acceder = CreateUnoService("com.sun.star.sheet.FunctionAccess")
params = Array(ligne+1, col+1, absol, "")
celladdr = acceder.callFunction("Address", params())
End Function
```

Le nom de la fonction est insensible aux majuscules/minuscules, mais doit être le nom anglais. Les paramètres d'appel de la fonction sont enfermés dans la variable tableau *params*, un élément par argument à passer.

La macro *TraduireFormule* vous aidera à traduire vos formules. Dans un classeur Calc, tapez votre formule française en A1 de la feuille 1, exécutez la macro, la traduction anglaise de la formule se trouve maintenant en A3. Cela marche même si votre formule a des erreurs de syntaxe !

```
Sub TraduireFormule
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

94/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim MonDocument As Object, LesFeuilles As Object
Dim MaFeuille As Object
Dim CelluleLocale As Object, CelluleAngl As Object

MonDocument = ThisComponent
LesFeuilles = MonDocument.Sheets
MaFeuille = LesFeuilles(0)
CelluleLocale = MaFeuille.getCellRangeByName("A1")
CelluleAngl = MaFeuille.getCellRangeByName("A3")
CelluleAngl.String = CelluleLocale.Formula
End Sub
```

Exemple, montrant l'intérêt de cette macro :

```
=ALEA.ENTRE.BORNES(56;999)
```

est traduite en :

```
=com.sun.star.sheet.addin.Analysis.getRandbetween(56;999)
```

D'autres fonctions de Calc ont pour argument une plage de cellules. En Basic cela correspond à fournir pour l'argument un tableau de valeurs ; si on veut transmettre la plage de cellules, on en récupère un tableau avec un `getDataArray`.

p) Trier une table

p.a) Tri par colonnes

Pour trier une table qui se trouve dans une feuille Calc il est nécessaire de remplir plusieurs descripteurs; pour un tri classique par colonnes, on remplit :

- d'une part un descripteur pour chaque colonne de tri qui précise :
 - de quelle colonne il s'agit,
 - et si le tri est ascendant ou descendant;
 - l'ensemble des ces descripteurs de colonnes constitue un tableau dont
 - l'index 0 indique la colonne à trier en premier,
 - l'index 1 la colonne pour départager les ex-aequo du premier critère tri,
 - l'index 2 la colonne pour départager les ex-aequo restants après le deuxième critère de tri.
- d'autre part un descripteur général de tri, qui précise :
 - si on trie des lignes ou des colonnes,

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

95/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- si la table contient une ligne d'en-tête (anglais : `Header`),
- si le tri est par colonne ou par ligne
- et l'ensemble des colonnes concernées, obtenu avec le descripteur précédent.

Ensuite, le tri est lancé avec la méthode `Sort`.

Voici un exemple de tri d'une table dont la ligne 1 contient les en-têtes de colonnes, à trier selon les colonnes B en ordre descendant, puis A en ascendant, puis D en ascendant. La zone à trier est une plage nommée, qui contient l'ensemble de la table.

```
Dim MaFeuille As Object, MaZone As Object
Dim ConfigTri(2) As New com.sun.star.util.SortField
Dim DescrTri(2) As New com.sun.star.beans.PropertyValue

MaFeuille = ThisComponent.Sheets.getByName("F_Tris")
MaZone = MaFeuille.getCellRangeByName("ZoneDeTris")

ConfigTri(0).Field = 1 ' colonne B
ConfigTri(0).SortAscending = false
ConfigTri(1).Field = 0 ' colonne A
ConfigTri(1).SortAscending = true
ConfigTri(2).Field = 3 ' colonne D
ConfigTri(2).SortAscending = true

DescrTri(0).Name = "SortFields"
DescrTri(0).Value = ConfigTri()
DescrTri(1).Name = "Orientation"
DescrTri(1).Value = com.sun.star.table.TableOrientation.ROWS
DescrTri(2).Name = "ContainsHeader"
DescrTri(2).Value = true

MaZone.Sort(DescrTri())
```

Remarques

1. Dans l'instruction `Dim ConfigTri(2)` le numéro d'index maximal correspond au nombre de colonnes servant de critère de tri. Si vous triezy selon une seule colonne vous écrirez `Dim ConfigTri(0)`
2. Si une colonne de tri contient des textes, le tri distingue les majuscules des minuscules.
3. La valeur donnée pour l'orientation est bien correcte pour un tri " par colonne " !
4. Les arguments de Name doivent être écrits avec les majuscules/minuscules

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

96/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

indiquées.

5. Les parenthèses vides sont indispensables dans la dernière instruction.
6. Comme avec l'interface utilisateur, il n'est pas possible de trier sur plus de 3 critères à la fois.

p.b) Tri par lignes

Dans le tri de lignes, l'en-tête éventuel est la colonne A, et le tri s'effectue par rapport au contenu d'une, ou plusieurs lignes : inclinez la tête de 90° sur votre gauche, vous retrouvez le tableau classique...

Reprise de l'exemple précédent, en configuration de tri de lignes :

```
Dim MaFeuille As Object, MaZone As Object
Dim ConfigTri(2) As New com.sun.star.util.SortField
Dim DescrTri(2) As New com.sun.star.beans.PropertyValue

MaFeuille = ThisComponent.Sheets.getByName("F_Trise")
MaZone = MaFeuille.getCellRangeByName("ZoneDeTrise")

ConfigTri(0).Field = 1 ' ligne 2
ConfigTri(0).SortAscending = false
ConfigTri(1).Field = 0 ' ligne 1
ConfigTri(1).SortAscending = true
ConfigTri(2).Field = 3 ' ligne 4
ConfigTri(2).SortAscending = true

DescrTri(0).Name = "SortFields"
DescrTri(0).Value = ConfigTri()
DescrTri(1).Name = "Orientation"
DescrTri(1).Value = com.sun.star.table.TableOrientation.COLUMNS
DescrTri(2).Name = "ContainsHeader"
DescrTri(2).Value = true

MaZone.Sort(DescrTri())
```

q) Les dessins dans Calc

Ici, on appelle une forme (anglais : shape) un dessin élémentaire, par exemple un rectangle, une ellipse, etc.

q.a) Insérer une forme dessinée

Une forme est insérée dans une page de dessin. Il existe autant de pages de dessin que de feuilles dans le tableur Calc, avec la même numérotation à partir de zéro.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

97/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Cet exemple minimal insère un rectangle dans la troisième feuille du tableur :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim PageDessin As Object
Dim MaForme As Object
Dim Taille1 As New com.sun.star.awt.Size

Taille1.Width = 5400
Taille1.Height = 2530

MonDocument = ThisComponent

PageDessin = MonDocument.DrawPages(2) ' page de la feuille Calc de rang 2
MaForme = MonDocument.CreateInstance("com.sun.star.drawing.RectangleShape")
MaForme.Size = Taille1
PageDessin.add(MaForme)
```

L'instruction `Dim Taille1` crée un objet de type Taille (en anglais : Size) qui servira à dimensionner la future forme. Basic ne permet pas de modifier directement la propriété `Size` d'une forme.

L'argument de `createInstance` est une [horrible constante](#) qui précise le type de forme dessinée.

Les principales formes élémentaires disponibles dans `com.sun.star.Drawing` sont :

Forme	Signification
<code>CaptionShape</code>	Etiquette
<code>ClosedBezierShape</code>	Courbe de Bézier fermée
<code>ConnectorShape</code>	Connecteur
<code>EllipseShape</code>	Ellipse ou cercle
<code>GraphicObjectShape</code>	Objet graphique
<code>LineShape</code>	Ligne
<code>MeasureShape</code>	Ligne de cote
<code>OpenBezierShape</code>	Courbe de Bézier ouverte
<code>PolyLineShape</code>	Ligne brisée
<code>PolyPolygonShape</code>	Polygone
<code>RectangleShape</code>	Rectangle ou carré
<code>TextShape</code>	Texte

Une fois le dessin ajouté à la page, on peut changer ses propriétés, listées plus loin.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

98/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

q.b) Ancrage de la forme

La forme obtenue avec les instructions de l'exemple précédent est positionné dans le coin en haut à gauche de la feuille, ancré à la cellule A1.

L'interface utilisateur permet (clic droit sur le dessin) de choisir un ancrage à la page ou à la cellule. Avec un ancrage à la cellule, on peut déplacer à la souris la forme vers une autre cellule.

Le positionnement par macro utilise en fait un ancrage intermédiaire entre " page " et " cellule ".

q.c) Positionner la forme

Le positionnement fait appel à une variable intermédiaire, qui définit la position d'un point, en coordonnées x et y. Ce sera la position du coin haut-gauche de la forme. Les coordonnées sont exprimées en 1/100 de mm, relativement au coin haut-gauche de la feuille.

```
Dim MonDocument As Object, LesFeuilles As Object
Dim PageDessin As Object
Dim MaForme As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Posit1 As New com.sun.star.awt.Point

Taille1.Width = 5400
Taille1.Height = 2530
Posit1.x = 5700 ' 55 mm vers la droite
Posit1.y = 8350 ' 83,5 mm vers le bas

MonDocument = ThisComponent

PageDessin = MonDocument.DrawPages(2) ' page de la feuille Calc de rang 2
MaForme = MonDocument.CreateInstance("com.sun.star.drawing.RectangleShape")
MaForme.Size = Taille1
MaForme.Position = Posit1
PageDessin.add(MaForme)
```

Le résultat de ces instructions est une forme ancrée à une cellule de la feuille, par exemple B18. Que s'est-il passé ? L'API a choisi comme cellule d'ancrage celle qui englobe le point de positionnement, et a complété le positionnement de la forme par rapport à cette cellule pour obtenir la position voulue. Effacez la forme; modifiez la largeur de la colonne A et la hauteur d'une des premières lignes; ré-exécutez la macro : le choix de la cellule d'ancrage changera.

En pratique, ceci veut dire que le positionnement est toujours calculé par rapport à la page, et converti en positionnement à la cellule.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

99/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Si, après insertion de la forme, on modifie la largeur de la colonne ou la hauteur de la ligne contenant la cellule d'ancrage, les dimensions de la forme changeront. Si on modifie la largeur d'une colonne précédente ou la hauteur d'une ligne précédente, la position de la forme changera.

q.d) Autres fonctionnalités autour de la forme

Les autres fonctionnalités sont identiques pour Calc et les autres applications. Elles sont décrites dans un [chapitre commun](#) :

- Propriétés communes
- Dessin de différentes formes
- Ecriture de texte dans la forme
- Donner un nom à une forme

r) Les images dans Calc

r.a) Insérer une image

Note : dans l'API version anglaise, le terme Graphic est utilisé pour désigner une image "bitmap" ou vectorielle. Il n'y a aucun rapport avec le concept français de graphique.

Cet exemple insère une image :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim PageDessin As Object
Dim MonImage As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent

PageDessin = MonDocument.DrawPages(2) ' page de la troisième feuille Calc
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")

PageDessin.add(MonImage)

Taille1.Width = 6000 ' largeur en 1/100 de mm
Taille1.Height = 6000 ' largeur en 1/100 de mm
MonImage.Size = Taille1
```

Si l'image originale n'a pas les mêmes proportions, l'image sur le document sera déformée. Nous verrons plus loin comment redimensionner en gardant les proportions originales.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

100/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Ces instructions n'insèrent pas l'image dans le document Calc, mais seulement un lien vers l'image. Si vous déplacez, renommez, supprimez le fichier image référencé, le document affichera une case de lien brisé avec l'adresse URL du fichier.

Ceci est une limitation actuelle de l'API.

Pour effectivement insérer dans le document des images insérées par macro, suivre cette procédure :

1. Menu Edition > Liens...
 - sélectionner l'ensemble des liens images
 - cliquer sur le bouton Déconnecter; confirmer
2. Sauver le document.

r.b) Positionner une image

L'ancrage et le positionnement d'une image et d'une forme sont identiques. Reportez-vous au chapitre concernant le [positionnement des formes](#) dans Calc.

r.c) Dimensionner une image

Pour dimensionner l'image sans changer les proportions, on récupère les dimensions de l'image en pixels. Ceci n'est possible qu'après l'instruction `PageDessin.add(MonImage)`. La proportion calculée sert à fixer la hauteur en fonction de la largeur. Ces deux dimensions sont en 1/100 de mm.

Cet exemple place l'image dans la feuille et redimensionne l'image à une largeur choisie, tout en gardant ses proportions :

```
Dim MonDocument As Object, LesFeuilles As Object
Dim PageDessin As Object
Dim MonImage As Object
Dim LeBitmap As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Posit1 As New com.sun.star.awt.Point
Dim Proportion As Double

MonDocument = ThisComponent

PageDessin = MonDocument.DrawPages(2) ' page de la feuille Calc de rang 2
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")

Posit1.x = 3000
Posit1.y = 6000
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

101/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonImage.Position = Posit1  
  
PageDessin.add(MonImage)  
  
LeBitmap = MonImage.GraphicObjectFillBitmap  
Taille1 = LeBitmap.Size ' taille en pixels !  
Proportion = Taille1.Height / Taille1.Width  
Taille1.Width = 6000 ' largeur en 1/100 de mm  
Taille1.Height = Taille1.Width * Proportion  
MonImage.Size = Taille1
```

r.d) Insérer plusieurs images

A chaque insertion d'image il est nécessaire d'obtenir un nouveau `GraphicObjectShape`, même si on insère plusieurs fois la même image, comme dans l'exemple qui suit. Il comporte un sous-programme facilitant le redimensionnement.

```
Sub Insérer2ImagesIdentiques  
Dim MonDocument As Object, LesFeuilles As Object  
Dim PageDessin As Object  
Dim MonImage As Object  
Dim Posit1 As New com.sun.star.awt.Point  
  
MonDocument = ThisComponent  
  
PageDessin = MonDocument.DrawPages(2) ' page de la feuille Calc de rang 2  
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")  
Posit1.x = 3000  
Posit1.y = 6000  
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")  
MonImage.Position = Posit1  
PageDessin.add(MonImage)  
LargeProp(MonImage, 6000)  
  
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")  
Posit1.x = 10000  
Posit1.y = 8000  
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")  
MonImage.Position = Posit1  
PageDessin.add(MonImage)  
LargeProp(MonImage, 3500)  
  
End Sub  
  
Sub LargeProp( UneImage As Object, Largeur As Long)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

102/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim LeBitmap As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Proportion As Double

LeBitmap = UneImage.GraphicObjectFillBitmap
Taille1 = LeBitmap.Size ' taille en pixels !
Proportion = Taille1.Height / Taille1.Width
Taille1.Width = Largeur ' largeur en 1/100 de mm
Taille1.Height = Taille1.Width * Proportion
UneImage.Size = Taille1
end sub
```

r.e) Trouver une image par son nom

Pour pouvoir retrouver une image, un bon moyen est de la nommer. Une fois l'image insérée dans le document, il suffit d'utiliser la propriété `Name` :

```
MonImage.Name = "Mon chien"
```

Toutes les images (et les formes) d'une page sont accessibles par un index, exemple :

```
MonImage = PageDessin(3)
```

Ceci est peu pratique, mais dans Calc il n'existe pas de méthode API pour retrouver une forme par son nom.

Pas de problème, nous allons utiliser la fonction *FindObjectByName* décrite dans un autre [chapitre](#). Avec cette fonction, nous pouvons retrouver l'image et la modifier :

```
Dim MonDocument As Object
Dim MonImage As Object
Dim PageDessin As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent
PageDessin = MonDocument.DrawPages(2)
MonImage = FindObjectByName( PageDessin, "Mon chien")
if IsNull(MonImage) then
    print "Nom inexistant !"
else
    Taille1.Width = 6000 ' largeur en 1/100 de mm
    Taille1.Height = 3000 ' hauteur en 1/100 de mm
    MonImage.Size = Taille1
end if
```

Attention : cette méthode retrouve un objet quelconque sur la page, aussi évitez de donner le même nom à une forme et à une image de la même page.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

103/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

VII.1.7 Traitement de texte Writer

a) Trouver le texte

Pour travailler sur un texte de Writer nous avons besoin de le désigner. Le fait de désigner le document lui-même est insuffisant. En effet, un document Writer contient différents objets, dont le texte ordinaire, les en-tête et pied de page, ou des cadres contenant du texte.

Pour une macro qui travaille sur le texte ordinaire du document en cours, on écrit :

```
Dim MonDocument As Object, MonTexte As Object  
MonDocument = ThisComponent  
MonTexte = MonDocument.Text
```

b) Gérer les curseurs

b.a) Curseur visible, curseur d'écriture

Il faut bien comprendre qu'il existe deux types de curseurs :

1. le curseur visible, c'est celui que vous voyez avancer quand vous tapez du texte dans Writer; il sera décrit plus loin
2. le curseur d'écriture, qui est utilisé par la macro pour désigner le texte qu'elle manipule.

Pour insérer du texte ordinaire à partir du début du document Writer, on définit ainsi le curseur d'écriture :

```
MonCurseur = MonTexte.createTextCursor
```

Par contre, pour insérer du texte ailleurs dans un texte existant nous avons besoin de déplacer ce curseur.

b.b) Déplacer le curseur d'écriture

Voici deux exemples de déplacement du curseur d'écriture :

```
MonCurseur.goRight(1, false) ' se déplacer à droite d'un caractère  
MonCurseur.goLeft(2, true) ' sélectionner les 2 caractères précédents
```

Faites bien attention : le curseur d'écriture se déplace, mais le curseur visible, lui, n'est pas déplacé.

Le deuxième argument de `goRight` et `goLeft` signifie :

- false = déplacer le curseur
- true = déplacer le curseur en étendant la sélection (comme en glissant avec la

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

souris). La sélection de travail est invisible.

Liste des instructions pour le curseur

Les instructions pour déplacer le curseur ont une syntaxe identique:

Méthode	Signification
<code>goRight (n, false)</code>	déplacer de n caractères à droite
<code>goLeft (n, false)</code>	déplacer de n caractères à gauche
<code>gotoStart (false)</code>	déplacer au début du texte complet
<code>gotoEnd (false)</code>	déplacer à la fin du texte complet
<code>gotoStartOfParagraph (false)</code>	déplacer au début du paragraphe en cours
<code>gotoEndOfParagraph (false)</code>	déplacer à la fin du paragraphe en cours
<code>gotoNextParagraph (false)</code>	déplacer au début du paragraphe suivant
<code>gotoPreviousParagraph (false)</code>	déplacer au début du paragraphe précédent
<code>gotoNextWord (false)</code>	déplacer au début du mot suivant
<code>gotoPreviousWord (false)</code>	déplacer au début du mot précédent
<code>gotoStartOfWord (false)</code>	déplacer au début du mot courant
<code>gotoEndOfWord (false)</code>	déplacer à la fin du mot courant
<code>gotoNextSentence (false)</code>	déplacer au début de la phrase suivante
<code>gotoPreviousSentence (false)</code>	déplacer au début de la phrase précédente
<code>gotoStartOfSentence (false)</code>	déplacer au début de la phrase courante
<code>gotoEndOfSentence (false)</code>	déplacer à la fin de la phrase courante

Instructions pour manipuler les extrémités du curseur

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

105/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Méthode	Signification
<code>collapseToStart</code>	amène la fin du curseur sur la position du début de curseur
<code>collapseToEnd</code>	amène le début du curseur sur la position de fin du curseur

Toutes ces instructions sont en réalité des fonctions, dont le résultat est vrai si l'action a pu être exécutée, ou faux sinon. On n'est pas obligé d'utiliser ce résultat.

Il existe des fonctions pour déterminer la position du curseur :

Fonction	Signification
<code>isStartOfParagraph</code>	résultat vrai si le curseur est au début d'un paragraphe
<code>isEndOfParagraph</code>	résultat vrai si le curseur est à la fin d'un paragraphe
<code>isStartOfWord</code>	résultat vrai si le curseur est au début du mot courant
<code>isEndOfWord</code>	résultat vrai si le curseur est à la fin du mot courant
<code>isStartSentence</code>	résultat vrai si le curseur est en début de phrase
<code>isEndSentence</code>	résultat vrai si le curseur est en fin de phrase
<code>isCollapsed</code>	résultat vrai si le début et la fin du curseur sont identiques

Les méthodes relatives aux mots et phrases peuvent donner des résultats surprenants. Voyez éventuellement plus loin les remarques concernant la [notion de mot](#).

b.c) Le curseur visible

Le curseur visible, c'est la barre verticale clignotante affichée sur votre texte à l'écran. C'est aussi une zone sélectionnée par l'utilisateur par exemple en faisant glisser la souris sur le texte.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

106/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Obtenir le curseur visible

```
Dim MonDocument As Object
Dim CurseurVisible As Object

MonDocument = ThisComponent
CurseurVisible = MonDocument.CurrentController.ViewCursor
```

Notez que *CurseurVisible* a certaines des possibilités d'un curseur d'écriture comme *MonCurseur*, mais si on déplace la position de *CurseurVisible*, ce ne sera pas reflété à l'écran. Pour utiliser toutes les possibilités, le mieux est d'initialiser un curseur d'écriture à la position du curseur visible.

Curseur d'écriture et curseur visible

Nous allons positionner un curseur d'écriture à la position du curseur visible. Mais le curseur visible peut sélectionner un point, ou une zone qui a un début et une fin.

Positionnons le curseur d'écriture au début de la zone du curseur visible :

```
Dim MonDocument As Object, MonTexte As Object
Dim CurseurVisible As Object, MonCurseur As Object

MonDocument = ThisComponent
CurseurVisible = MonDocument.currentcontroller.ViewCursor
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursorByRange
                (CurseurVisible.Start)
```

Positionnons le curseur d'écriture à la fin de la zone du curseur visible :

```
MonCurseur = MonTexte.createTextCursorByRange(CurseurVisible.End)
```

b.d) Zone sélectionnée par l'utilisateur

Il n'est pas utile d'initialiser un curseur d'écriture pour certaines tâches simples. Voici comment récupérer le texte sélectionné par l'utilisateur et le remplacer :

```
Dim MonDocument As Object
Dim CurseurVisible As Object
Dim TexteDeLaSelection As String

MonDocument = ThisComponent
CurseurVisible = MonDocument.currentcontroller.ViewCursor
TexteDeLaSelection = CurseurVisible.String
CurseurVisible.String = " nouveau texte "
```

Dans la séquence suivante on met en gras la zone de texte sélectionnée par l'utilisateur (la mise en gras est expliquée plus loin) :

```
Dim MonDocument As Object
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

107/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim CurseurVisible As Object  
  
MonDocument = ThisComponent  
CurseurVisible = MonDocument.currentcontroller.ViewCursor  
CurseurVisible.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

Notez que la séquence ci-dessus est aussi valide si plusieurs zones non contiguës sont sélectionnées par l'utilisateur : toutes les zones sont mises en gras !

Si l'utilisateur sélectionne plusieurs zones à la fois, il est plus souvent utile de traiter successivement chaque zone. La propriété `CurrentSelection` fournit une liste des zones sélectionnées. On utilise alors une boucle pour trouver successivement ces zones. Le nombre de zones est dans la propriété `Count`.

```
Dim MonDocument As Object  
Dim LesSelections As Object  
Dim selx As Integer, UneZone As Object  
  
MonDocument = ThisComponent  
LesSelections = MonDocument.CurrentSelection  
for selx = 0 to LesSelections.Count -1  
    UneZone = LesSelections(selx)  
    rem mettre en gras chaque zone sélectionnée  
    UneZone.CharWeight = com.sun.star.awt.FontWeight.BOLD  
next
```

Si vous n'êtes pas sûr que l'utilisateur n'a sélectionné qu'une seule zone, employez le codage ci-dessus. S'il n'y a aucune sélection, la boucle `for` n'est pas exécutée.

Attention, la variable `UneZone` n'est pas un curseur. Pour obtenir un curseur à partir d'une zone sélectionnée, faire :

```
Dim UnCurseur As Object  
UnCurseur = MonTexte.createTextCursorByRange (UneZone)
```

b.e) Sélectionner visiblement une zone

Le but est de sélectionner une zone par macro, et d'afficher cette sélection à l'utilisateur. Ce n'est pas en modifiant `CurseurVisible` que vous y arriverez. Il faut une instruction spéciale, qui utilise la position du curseur mis en argument :

```
MonCurseur.gotoNextWord(false)  
MonCurseur.gotoEndOfWord(true) ' selection d'un mot  
' Transférer la sélection sur le curseur visible  
MonDocument.CurrentController.Select (MonCurseur)
```

Si vous avez exécuté cette macro dans l'EDI, mettez en avant-plan la fenêtre Writer pour voir la sélection du mot.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

108/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c) Ecrire du texte

c.a) Lire, écrire

On peut donc sélectionner avec le curseur une zone de plusieurs caractères; ceci fait, on récupère le texte sélectionné ainsi :

```
Dim tx3 As String  
tx3 = MonCurseur.String
```

Inversement, on peut écraser un texte sélectionné avec un nouveau texte :

```
MonCurseur.String = "Bonsoir"
```

Attention : si vous utilisez ce principe avec l'objet Texte, vous écrasez tout texte existant :

```
MonTexte.String = "tout est écrasé par ceci"
```

Notez que Basic limite les chaînes de caractères à 64 000 caractères. Ces méthodes ne marchent donc pas pour des textes plus longs.

Habituellement nous écrirons du texte ainsi :

```
MonTexte.insertString( MonCurseur, "Ceci est un texte", false)
```

Après cette instruction, le curseur d'écriture est positionné à la fin du texte qu'elle a écrit.

Dans la procédure `insertString` le troisième argument signifie :

- `false` = insérer dans le texte
- `true` = écraser dans le texte

c.b) Caractères spéciaux dans un texte

Insérer une fin de paragraphe de texte :

```
Dim FinParagraphe As Integer  
FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK  
MonTexte.insertControlCharacter(MonCurseur, FinParagraphe, false)
```

Le dernier argument (`false`) a la même signification que pour `insertString`.

Nous avons employé la variable intermédiaire *FinParagraphe* pour simplifier le codage.

Il existe une liste d'[horribles constantes](#) similaires :

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
PARAGRAPH_BREAK	fin de paragraphe
LINE_BREAK	retour à la ligne
HARD_HYPHEN	tiret insécable
SOFT_HYPHEN	tiret conditionnel
HARD_SPACE	espace insécable
APPEND_PARAGRAPH	insérer un paragraphe après celui en cours et se positionner au début

Supprimer des paragraphes

Cet exemple utilise plusieurs des notions décrites jusqu'ici. Il parcourt un texte complet et supprime les marques de paragraphe en remplaçant chacune par un espace.

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor

MonCurseur.gotoStart (false)
Do While MonCurseur.gotoNextParagraph (false)
    MonCurseur.goLeft (1, true)
    MonCurseur.String = " "
Loop
```

c.c) Insérer un saut de page ou de colonne

Le saut de page est une propriété de paragraphe. Vous pouvez définir un style de paragraphe comportant un saut de page. Sinon, vous pouvez insérer une marque de saut de page dans le paragraphe en cours :

```
Dim genreSautpage As Integer
genreSautpage = com.sun.star.style.BreakType.PAGE_AFTER
MonCurseur.breakType = genreSautpage
```

Vous avez reconnu une horrible constante. Différents types de saut de page sont possibles :

Constante	Signification
PAGE_BEFORE	changer de page avant le paragraphe en cours
PAGE_AFTER	changer de page après le paragraphe en cours

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

110/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
PAGE_BOTH	changer de page avant et après le paragraphe en cours
NONE	supprimer le saut de page ou colonne qui existe dans le paragraphe
COLUMN_BEFORE	changer de colonne avant le paragraphe en cours
COLUMN_AFTER	changer de colonne après le paragraphe en cours
COLUMN_BOTH	changer de colonne avant et après le paragraphe en cours

Les sauts de type `COLUMN` sont utilisés dans un texte en colonnes. En effet, on peut avoir deux ou trois colonnes par page, et souhaiter changer de colonne sans obligatoirement changer de page.

c.d) Exemples de manipulation de texte

Cette macro va corriger vos fautes de frappe; elle inverse les deux caractères de part et d'autre de la position pointée par l'utilisateur :

```
Sub Inverser2car

Dim MonDocument As Object, MonTexte As Object
Dim CurseurVisible As Object, MonCurseur As Object
Dim tx0, tx1, tx2 As String

MonDocument = ThisComponent
MonTexte = MonDocument.Text
rem on suppose le curseur visible positionné entre les deux caractères
CurseurVisible = MonDocument.currentcontroller.ViewCursor
MonCurseur = MonTexte.createTextCursorByRange(CurseurVisible.Start)

MonCurseur.goRight(1, false) ' position après le 2eme caractère
MonCurseur.goLeft(2, true) ' sélectionner les 2 caractères précédents
tx0 = MonCurseur.String
tx1 = Left(tx0, 1)
tx2 = Right(tx0, 1)
tx0 = tx2 & tx1 ' inverser les 2 caractères
MonCurseur.String = tx0
'restituer la position initiale du curseur visible
CurseurVisible.GoLeft(1, false)

End Sub
```

Reprenons l'exemple de macro pour [supprimer des paragraphes](#). Cette nouvelle

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

111/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

version ne supprime les paragraphes que dans la zone sélectionnée par l'utilisateur :

```
Dim MonDocument As Object, MonTexte As Object
Dim CurseurVisible As Object, MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
CurseurVisible = MonDocument.currentcontroller.ViewCursor
rem creer un curseur de travail à la position du curseur visible
MonCurseur = MonTexte.createTextCursorByRange(CurseurVisible.Start)
Do While MonCurseur.gotoNextParagraph(false)
  if MonTexte.compareRegionEnds(MonCurseur, CurseurVisible) < 0 then Exit Do
  MonCurseur.goLeft(1, true)
  MonCurseur.String = " "
Loop
```

La fonction `compareRegionEnds` compare les positions de fin dans les deux curseurs.

Remarque : si l'utilisateur sélectionne plusieurs zones, la macro ne supprime rien.

c.e) Astuce : utiliser les signets pour écrire un texte

Un signet (anglais : bookmark) est un repère dans le texte. Manuellement vous insérez un signet à l'endroit du curseur visible avec le menu Insertion > Repère de texte... > nomDuSignet.

Vous pouvez vous simplifier considérablement l'écriture de texte par macro si vous remplissez un document pré-défini (ou un modèle de document) qui contiendra la structure fixe du document, et des signets pour repérer les endroits que la macro doit remplir. Evidemment la macro doit connaître le nom de chaque signet et le type d'information à y insérer.

L'exemple suivant initialise le curseur d'écriture à l'emplacement repéré par le signet nommé "ecrire_ici", puis écrit " Texte nouveau " :

```
MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonDocument.Bookmarks.getByname("ecrire_ici").getAnchor
MonTexte.insertstring( MonCurseur, " Texte nouveau ", false)
```

Le nom du signet doit être écrit exactement comme dans sa définition. L'instruction déclenchera une exception s'il n'existe aucun signet de ce nom. Vous pouvez tester l'existence du signet ainsi :

```
if MonDocument.Bookmarks.hasbyname("ecrire_ici") then
  rem le signet existe
end if
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

112/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

d) Appliquer un style ou un formatage

d.a) Appliquer un style à un paragraphe

Le moyen le plus efficace de formater un texte avec une macro est de lui appliquer des styles que vous avez définis au préalable. Si vous avez besoin de créer un nouveau document dans votre projet de macro, vous pouvez définir manuellement un document modèle avec les styles dont vous avez besoin, puis utiliser ce modèle pour créer le nouveau document.

La macro suivante fait exactement comme si vous affectiez vous-même le style “Titre 4” au deuxième paragraphe du document en cours. Testez-la avec un petit document de trois paragraphes.

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor
MonCurseur.gotoNextParagraph(false)
rem ici on est au deuxieme paragraphe du texte existant

MonCurseur.paraStyleName = "Titre 4"
```

Le curseur d'écriture peut être dans une position quelconque dans le paragraphe. J'aurais pu le décaler de 5 caractères, par exemple.

On peut aussi connaître le style d'un paragraphe :

```
print MonCurseur.paraStyleName
```

Attention

Si vous récupérez le nom du style en cours, pour les styles standards fournis avec OOo vous obtiendrez le nom anglais, même avec une version localisée de OOo. Dans l'exemple ci-dessus, on affecte le style “Titre 4” et on relira le style “Heading 4”. Par contre les styles que vous créez n'ont évidemment qu'un seul nom. Donc, en cas de doute, lisez le nom du style avec une petite macro.

Deuxième problème potentiel : la chaîne de caractère du style que vous affectez à un paragraphe doit reproduire exactement le nom du style existant : majuscules, minuscules, accents. Sinon le style du paragraphe restera inchangé.

d.b) Affecter un style à un caractère

Pour changer le style de certains caractères, on les sélectionne et on applique

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

113/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

CharStyleName :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextWord(false)
MonCurseur.gotoEndOfWord(true)

MonCurseur.CharStyleName = "MonStyleCaract"
```

d.c) Affecter un formatage à un caractère

Ceci est l'équivalent d'appliquer Gras, Italique, Souligné, etc, à un ou plusieurs caractères.

Exemple sur le deuxième mot du deuxième paragraphe :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextWord(false)
MonCurseur.gotoEndOfWord(true)

MonCurseur.CharWeight = com.sun.star.awt.FontWeight.BOLD
```

Les propriétés de caractère sont listées dans l'API : [CharacterProperties](#). Les principales :

Propriété	Signification
CharStyleName	Chaîne de caractère, nom du style de caractère (respecter majuscules et minuscules)
CharFontName	Chaîne de caractères, nom de la police (respecter majuscules et minuscules)
CharHeight	La taille du caractère, en points
CharWeight	La " <u>grasse</u> " du caractère

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

114/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Propriété	Signification
<code>CharShadowed</code>	booléen; valeur true pour mettre une ombre au caractère
<code>CharPosture</code>	Caractère <u>penché ou droit</u>
<code>CharUnderline</code>	Pour <u>souligner le caractère</u>
<code>CharColor</code>	<u>Couleur</u> du caractère
<code>CharBackColor</code>	<u>Couleur</u> de fond du caractère
<code>CharCrossedOut</code>	booléen; valeur true pour barrer le caractère
<code>CharEscapement</code>	Pour un caractère en <u>exposant ou en indice</u>
<code>CharEscapementHeight</code>	Pour un caractère en <u>exposant ou en indice</u>

Certaines propriétés utilisent des horribles constantes comme pour `CharWeight` dans l'exemple ci-dessus. Voici le détail des principales propriétés :

- `CharWeight` constantes comme :
`com.sun.star.awt.FontWeight.XXXX`

Constante	Signification
<code>NORMAL</code>	Normal
<code>BOLD</code>	Gras exemple
<code>THIN</code>	Maigre (selon disponibilité dans la police)
<code>ULTRABOLD</code>	Très gras (selon disponibilité dans la police)

D'autres valeurs peu courantes existent, voir l' API.

- `CharPosture` constantes comme :
`com.sun.star.awt.FontSlant.XXXX`

Préparé par - Société Guidella - confidentiel

`coursMacro.sxw`

Imprimé le 28 décembre 2004

115/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
NONE	caractère droit
ITALIC	caractère en italique <i>exemple</i>

- `CharUnderline` constantes comme :
`com.sun.star.awt.FontUnderline.XXXX`

Constante	Signification
NONE	normal, non souligné
SINGLE	souligné simple <u>exemple</u>
DOUBLE	souligné double <u>exemple</u>
DOTTED	souligné pointillé <u>exemple</u>
WAVE	souligné ondulé <u>exemple</u>
DASH	souligné tireté <u>exemple</u>
DASHDOT	souligné tiret-point <u>exemple</u>
BOLD	souligné gras <u>exemple</u>

D'autres valeurs peu courantes existent, voir l' API.

Caractère en exposant ou en indice

Mettre un caractère en exposant ou en indice nécessite deux propriétés de caractère : `CharEscapement` et `CharEscapementHeight`.

- `CharEscapement` spécifie la hauteur du caractère par rapport à un caractère normal, en pourcentage; une valeur positive pour un exposant, une valeur négative pour un indice.
- `CharEscapementHeight` spécifie la taille relative du caractère, en pourcentage, par rapport à la taille du caractère normal.

Exemple :

```
MonCurseur.gotoEndOfWord(false)
MonCurseur.goLeft(1, true) ' sélectionner un caractère
MonCurseur.CharEscapement = 20 ' hauteur du caractère / ligne
MonCurseur.CharEscapementHeight = 70 ' prop hauteur caractère / taille normale
```

Vous pouvez vérifier le résultat avec l'interface utilisateur : sélectionner le caractère modifié par la

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

116/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

macro, menu Format > Caractères... onglet Position.

Remarque : la documentation de l'API inverse les descriptions de ces deux propriétés.

d.d) Remettre le formatage du style du paragraphe

Pour tout nettoyer, c'est-à-dire ne laisser que le formatage propre au style du paragraphe en cours, on applique à une sélection de texte la méthode `setAllPropertiesToDefault`.

```
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextParagraph(true) ' selectionner tout un paragraphe
MonCurseur.setAllPropertiesToDefault
```

e)Exercice d'écriture de texte

Pour aller plus loin, on va faire un travail pratique. Ouvrez un nouveau document Writer, tapez quelques paragraphes en style "standard". Puis avec le menu Outils/Macro créez la macro suivante :

```
Option Explicit

Sub Main
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object
Dim FinParagraphe As Integer

MonDocument = ThisComponent
MonTexte = MonDocument.Text
FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK
MonCurseur= MonTexte.createTextCursor

MonTexte.insertString( MonCurseur, "Début de texte", false)
MonTexte.insertControlCharacter( MonCurseur, FinParagraphe , false)
MonTexte.insertString( MonCurseur, "Deuxième ligne", false)
MonTexte.insertControlCharacter( MonCurseur, FinParagraphe , false)
MonTexte.insertString( MonCurseur, "Début paragr 3 ", false)
MonCurseur.paraStyleName= "Titre 4"
MonCurseur.CharColor = 225
MonCurseur.CharShadowed = true
MonTexte.insertString( MonCurseur, " ajout en couleurs", false)

End Sub
```

Laissez le curseur visible à la fin des quelques lignes que vous avez écrites sur le document. Ensuite exécutez la macro en " étape par étape ", tout en visualisant le contenu du texte dans le document.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

117/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Après “ Début de texte ” vous constaterez que cette séquence est écrite en tout début de document : c’est dû à l’initialisation faite par `createTextCursor`.

La méthode `paraStyleName` affecte le style “ Titre 4 ” au paragraphe en cours et à tout le texte actuel du paragraphe.

Les instructions `CharColor` et `CharShadowed` affectent une couleur bleue et une ombre au curseur, c’est-à-dire aux caractères qui seront écrits à partir d’ici – comme si vous le faisiez manuellement -

La dernière instruction exécutée, on voit que seul ce dernier texte est en couleurs.

Variantes

Ré-exécutez la macro, en effectuant des variantes. Pour chaque essai, effacez manuellement le texte écrit précédemment par la macro (cliquez sur le bouton “ Annuler ” dans la fenêtre du document Write).

- Changez manuellement le style de la première ligne du texte en “ corps de texte ”. Exécutez la macro : les deux premières lignes garderont ce style, mais la troisième sera en Titre 4.
- Changez ainsi les dernières lignes de la macro :

```
MonCurseur.paraStyleName= "Titre 4"  
MonCurseur.gotoStartOfParagraph(false)  
MonCurseur.CharColor = 225  
MonCurseur.CharShadowed = true  
MonTexte.insertString( MonCurseur, " ajout en couleurs", false)  
MonCurseur.gotoEndOfParagraph(false)  
MonTexte.insertString( MonCurseur, " en noir ", false)
```

L’ajout en couleurs est maintenant en début de paragraphe, le texte ajouté “ en noir ” suit bien le style en cours à cette position.

- Remplacez `gotoStartOfParagraph` par `gotoStart` et voyez la différence.
- Remplacez aussi `gotoEndOfParagraph` par `gotoEnd` et voyez la différence.
- Remplacez aussi `false` par `true` dans `gotoEnd` et dans le dernier `insertString` ; exécutez pas à pas : à la fin, tout le texte après “ ajout en couleurs ” est remplacé par le dernier insert, qui s’affiche avec le style de caractère en cours ici.
- Remplacez maintenant le dernier `insertString` par

```
MonCurseur.String = " en noir "
```

Le résultat est identique.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

118/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

f) Tabuler un texte

Insérer une tabulation dans un texte consiste à ajouter le caractère dont la valeur décimale est 9, c'est-à-dire chr(9).

Les positions de tabulations (les taquets) sont celles définies dans le style du paragraphe en cours, ou celles définies par défaut.

Si vous souhaitez des taquets spécifiques, deux solutions :

1. la plus simple et pratique est d'utiliser un style de paragraphe que vous avez défini dans votre document;
2. ou bien, définir laborieusement par programmation les taquets du paragraphe en cours, voir la macro ci-après.

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object, Textel As String
Dim NouvParagr As Integer
Dim Tabul As String
Dim PositionTaquet As New com.sun.star.style.TabStop
Dim ListeTaquets(2) As Object

NouvParagr = com.sun.star.text.ControlCharacter.APPEND_PARAGRAPH
Tabul = chr(9)

With PositionTaquet
    .Position = 2500 ' 25 mm ( 2,5 cm )
    .Alignment = com.sun.star.style.TabAlign.LEFT
    .DecimalChar = ","
    .FillChar = " "
    ListeTaquets(0) = PositionTaquet
    .Position = 4700 ' 47 mm
    .Alignment = com.sun.star.style.TabAlign.CENTER
    ListeTaquets(1) = PositionTaquet
    .Position = 7010 ' 70,1 mm
    .Alignment = com.sun.star.style.TabAlign.RIGHT
    ListeTaquets(2) = PositionTaquet
end With

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor
rem Insérer un paragraphe après celui en cours (ici, le premier)
MonTexte.insertControlCharacter(MonCurseur, NouvParagr, false)
MonCurseur.ParaTabStops = ListeTaquets() ' imposer les taquets
Textel = "Début"+Tabul+"Tab0"+Tabul+"Tab1"+Tabul+"Tab2"
MonTexte.insertString( MonCurseur, Textel, false)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

119/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Vous devez affecter tous les taquets en une seule fois, avec un tableau comportant le nombre de taquets nécessaires. Chaque taquet est constitué de plusieurs valeurs :

Composante	Signification
Position	La position du taquet par rapport à la marge gauche, en 1/100 de mm.
Alignment	Le type de tabulation, sous forme d'une horrible constante décrite plus bas.
DecimalChar	Le caractère séparateur décimal (en France, la virgule).
FillChar	Le caractère de suite, qui remplit les blancs, par exemple un pointillé.

Les types de tabulation possibles sont :

Constante	Signification
LEFT	Le taquet de tabulation se trouvera à gauche du texte à tabuler.
CENTER	Le taquet de tabulation se trouvera au centre du texte à tabuler.
RIGHT	Le taquet de tabulation se trouvera à droite du texte à tabuler.
DECIMAL	Le taquet de tabulation se trouvera sur le séparateur de décimales.
DEFAULT	Utiliser le type de tabulation par défaut.

Remarque : les explications de l'API (version du 04/02/2003) sur les types de tabulation sont complètement fausses.

g) Recherche et remplacement de texte

g.a) Remplacer une chaîne partout dans un texte

Ecrivez dans un document Writer plusieurs fois la chaîne de caractères "Annette". La macro suivante remplace tous les [mots](#) "Annette" par "Josette" dans tout le document, sans modifier des chaînes de caractères comme "Jannette".

```
Dim MonDocument As Object
Dim JeCherche As Object
Dim Nombredefois As Long

MonDocument = ThisComponent
JeCherche= MonDocument.createReplaceDescriptor
with JeCherche
  .SearchString = "Annette"
  .ReplaceString = "Josette"
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

120/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
.SearchWords = true ' default : false
.SearchCaseSensitive = true 'default : false
end with
Nombredefois = MonDocument.replaceAll(JeCherche)
print Nombredefois, "remplacement(s)"
```

Explications :

On crée un objet *JeCherche* qui contiendra tous les paramètres nécessaires à ce remplacement. Son initialisation requiert plusieurs instructions; Pour éviter la répétition lassante de *JeCherche* à chaque instruction, on utilise la séquence *With JeCherche ... end With* (littéralement : Avec ...).

Les propriétés du descripteur de remplacement permettent de préciser la recherche :

Propriété	Signification
SearchString	La chaîne de caractères à rechercher
ReplaceString	La chaîne de caractères en remplacement. Par défaut, cette chaîne est vide, ce qui revient à supprimer.
SearchWords	booléen; true pour ne rechercher que des <u>mots</u> ; par défaut on recherche la séquence de caractère n'importe où.
SearchCaseSensitive	booléen; true pour distinguer les majuscules des minuscules dans la recherche; par défaut il n'y a pas de distinction. (1)
SearchBackwards	booléen; true pour faire une recherche à reculons (ça peut servir); par défaut on recherche dans le sens normal de lecture.
SearchRegularExpression	booléen; true pour faire une recherche avec la méthode des expressions régulières (si ça ne vous dit rien, ne l'utilisez pas); par défaut on recherche une simple égalité de chaîne. (2)
SearchStyles	booléen; true pour rechercher des paragraphes d'un style donné par SearchString; par défaut on cherche du texte.
SearchSimilarity	booléen; true pour rechercher un texte similaire au texte cherché (2) (3)
SearchSimilarityAdd	Nombre de caractères à ajouter pour retrouver le texte cherché
SearchSimilarityRemove	Nombre de caractères à retrancher pour retrouver le texte cherché
SearchSimilarityExchange	Nombre de caractères à changer pour retrouver le texte cherché
SearchSimilarityRelax	booléen; true pour essayer toute combinaison des trois critères précédents qui permette de retrouver le texte cherché

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

121/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Notes :

- (1) La recherche sans distinction majuscule/minuscule distingue cependant les caractères accentués.
- (2) Fonctionne seulement à partir de la version 1.1 de OpenOffice
- (3) Remplir tous les critères associés, les valeurs par défaut ne sont pas zéro.

g.b) Remplacer un nom de style partout dans un texte

Cette macro remplace tous les utilisations du style de paragraphe “ Titre 4 ” par le style “Titre 5” :

```
Dim MonDocument As Object
Dim JeCherche As Object
Dim MonCurseur As Object
Dim unstyle as string

MonDocument = ThisComponent
JeCherche= MonDocument.createReplaceDescriptor
with JeCherche
  .SearchString = "Titre 4"
  .ReplaceString = "Titre 5"
  .SearchStyles = true
end with
MonDocument.replaceAll (JeCherche)
```

Relisez à ce sujet les remarques sur les noms de [styles de paragraphe](#).

g.c) Chercher et modifier du texte

Les méthodes de ce chapitre permettent d’effectuer des modifications quelconques sur les éléments cibles, par exemple changer le formatage de certains mots. De plus, la recherche peut être limitée à une partie du texte entier.

Chercher dans tout le texte

Cette macro recherche les mots “ hello ” dans tout le texte et les met en gras.

```
Dim MonDocument As Object
Dim JeCherche As Object
Dim posTrouve As Object

MonDocument = ThisComponent
JeCherche= MonDocument.createSearchDescriptor
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

122/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
with JeCherche
  .SearchString = "hello"
  .SearchWords = true
end with

posTrouve = MonDocument.findFirst(JeCherche)
Do Until isNull(posTrouve)
  posTrouve.CharWeight = com.sun.star.awt.FontWeight.BOLD
  posTrouve = MonDocument.findNext(posTrouve.End, JeCherche)
Loop
```

La structure du descripteur de recherche est un sous-ensemble de celle du descripteur de recherche et remplacement.

L'instruction `findFirst` commence la recherche au début du texte. L'objet *posTrouve* est une zone de texte qui sélectionne la cible trouvée. On peut donc appliquer les méthodes déjà exposées pour modifier cette cible : [écrire du texte](#), [changer le style](#), [changer le formatage](#). Ceci éventuellement après une analyse de la cible (selon son formatage actuel, sa position dans le texte, etc).

L'instruction `findNext` recherche à partir de la position indiquée en premier argument. Comme cet argument est le bord de fin de la cible précédemment trouvée, la recherche reprend là où on l'avait laissée.

Finalement, la recherche échoue et `findNext` (ou `findFirst`) renvoie un élément nul.

Chercher un texte dans une sélection utilisateur

Cette variante suppose que l'utilisateur a sélectionné une seule zone dans le texte. On va rechercher le texte dans cette sélection.

```
Dim MonDocument As Object, MonTexte As Object
Dim CurseurVisible As Object
Dim JeCherche As Object
Dim posTrouve As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
CurseurVisible = MonDocument.currentcontroller.ViewCursor()
JeCherche = MonDocument.createSearchDescriptor

with JeCherche
  .SearchString = "hello"
  .SearchWords = true
end with
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

123/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
posTrouve = MonDocument.findNext(CurseurVisible.Start, JeCherche)
Do Until isNull(posTrouve)
  if MonTexte.compareRegionEnds(posTrouve, CurseurVisible) < 0 then Exit Do
  posTrouve.CharWeight = com.sun.star.awt.FontWeight.BOLD
  posTrouve = MonDocument.findNext(posTrouve.End, JeCherche)
Loop
```

La recherche commence avec `findNext` afin de démarrer à partir de la position de début de la sélection. La fonction `compareRegionEnds` compare les positions de fin dans les deux curseurs.

Chercher un texte dans une zone quelconque

Pour effectuer la recherche sur une zone quelconque du texte, il suffit d'initialiser un curseur par programmation. Dans cette variante, on recherche dans le deuxième paragraphe du texte :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object
Dim JeCherche As Object
Dim posTrouve As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor ' curseur au début du document
MonCurseur.gotoNextParagraph(false) ' curseur au début du 2eme paragr
MonCurseur.gotoNextParagraph(true) ' curseur sélectionne le 2eme paragr
JeCherche = MonDocument.createSearchDescriptor

with JeCherche
  .SearchString = "hello"
  .SearchWords = true
end with

posTrouve = MonDocument.findNext(MonCurseur.Start, JeCherche)
Do Until isNull(posTrouve)
  if MonTexte.compareRegionEnds(posTrouve, MonCurseur) < 0 then Exit Do
  posTrouve.CharWeight = com.sun.star.awt.FontWeight.BOLD
  posTrouve = MonDocument.findNext(posTrouve.End, JeCherche)
Loop
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

124/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

h) Les tableaux dans un document Writer

h.a) Insérer un tableau

En supposant le document Writer ouvert, avec un curseur d'écriture positionné là où vous voulez insérer le tableau, voici comment insérer un tableau de 5 lignes et 9 colonnes :

```
Dim MaTable As Object
MaTable = MonDocument.CreateInstance("com.sun.star.text.TextTable")
MaTable.initialize(5,9) ' nombre de : lignes, colonnes
MonTexte.insertTextContent(MonCurseur, MaTable, false)
```

L'argument de `createInstance` est une " formule magique " à recopier scrupuleusement.

La méthode `insertTextContent` permet d'insérer un objet quelconque dans le texte, à la position du curseur. Ici, l'objet est un tableau. Le troisième argument signifie :

- false = insérer dans le texte
- true = écraser dans le texte

h.b) Propriétés du tableau

Le contrôle de la largeur du tableau est traité au chapitre suivant. Principales autres propriétés :

Propriété	Signification
RepeatHeadline	booléen; true pour répéter la ligne d'en-tête à chaque nouvelle page; valeur false par défaut
BreakType	impose un saut de page ou de colonne avant ou après le tableau. Mêmes <u>valeurs</u> que pour le texte ordinaire
Split	booléen; false pour imposer que le tableau ne soit pas à cheval sur deux pages ou deux colonnes (à condition que le tableau tienne dans la page !); valeur true par défaut, donc le tableau peut s'étendre sur deux pages ou plus
BackColor	<u>couleur</u> du fond de l'ensemble des cellules de la table

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

125/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Propriété	Signification
<code>Name</code>	chaîne de caractères, nom du tableau

Exemple, qui utilise `with` pour alléger l'écriture :

```
MaTable = MonDocument.createInstance("com.sun.star.text.TextTable")
with MaTable
  .RepeatHeadline = true
  .BreakType = com.sun.star.style.BreakType.PAGE_AFTER
  .BackColor = RGB(255,255,204) ' tout en jaune pâle
  .initialize(55,4) ' nombre de : lignes, colonnes
end with
MonTexte.insertTextContent(MonCurseur, MaTable, false)
```

h.c) Largeur du tableau

Il existe plusieurs propriétés inter-dépendantes qui modifient la largeur d'un tableau. Comme les précédentes propriétés, il est plus clair de les initialiser avant l'appel de `initialize`.

Positionnement horizontal

La propriété `HoriOrient` définit comment le tableau est positionné horizontalement, par rapport aux marges. C'est une horrible constante avec comme valeur par défaut :

```
com.sun.star.text.HoriOrientation.FULL
```

Principales valeurs possibles :

Constante	Signification
<code>NONE</code>	tableau étalé de la marge de gauche à la marge de droite
<code>RIGHT</code>	tableau aligné sur la marge de droite
<code>LEFT</code>	tableau aligné sur la marge de gauche
<code>CENTER</code>	tableau centré sur l'espace disponible
<code>FULL</code>	tableau étalé sur tout l'espace disponible, sans marges de table
<code>LEFT_AND_WIDTH</code>	tableau aligné sur la marge de gauche, avec une largeur imposée par la propriété <code>WIDTH</code>

Comme la valeur par défaut est `FULL`, il est absolument nécessaire de mettre une autre valeur pour

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

126/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

pouvoir imposer une largeur.

Réglages de largeur

Propriété	Signification
IsWidthRelative	booléen; true si RelativeWidth est à prendre en compte; false si Width est à prendre en compte
RelativeWidth	largeur en pourcentage de l'espace libre entre marges de gauche et de droite; valeur entre 1 et 100
Width	largeur absolue, en 1/100 de mm
LeftMargin	largeur absolue de la marge gauche, en 1/100 de mm
RightMargin	largeur absolue de la marge droite, en 1/100 de mm

Applications pratiques

Tableau centré, largeur 80%

```
with MaTable
  .HoriOrient = com.sun.star.text.HoriOrientation.CENTER
  .IsWidthRelative = true
  .RelativeWidth = 80
  .initialize(10,4) ' nombre de : lignes, colonnes
end with
```

Sur OOo version 1.1.0 la largeur de la dernière colonne est incorrecte.

Les propriétés `LeftMargin` et `RightMargin` sont ignorées en orientation `CENTER`.

Tableau centré, largeur absolue

```
with MaTable
  .HoriOrient = com.sun.star.text.HoriOrientation.CENTER
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

127/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
.IsWidthRelative = false  
.Width = 8000  
.initialize(10,4) ' nombre de : lignes, colonnes  
end with
```

Sur OOO version 1.1.0 la largeur obtenue est plus grande que celle indiquée.
Les propriétés `LeftMargin` et `RightMargin` sont ignorées en orientation `CENTER`.

Tableau aligné à gauche, largeur 80%

```
with MaTable  
.HoriOrient = com.sun.star.text.HoriOrientation.LEFT_AND_WIDTH  
.IsWidthRelative = true  
.RelativeWidth = 80  
.LeftMargin = 20*100 ' 20 mm  
.initialize(10,4) ' nombre de : lignes, colonnes  
end with
```

Sur OOO version 1.1.0 la largeur de la dernière colonne est incorrecte.
La propriété `RightMargin` est ignorée en orientation `LEFT_AND_WIDTH`.

Tableau aligné à gauche, largeur absolue

```
with MaTable  
.HoriOrient = com.sun.star.text.HoriOrientation.LEFT_AND_WIDTH  
.IsWidthRelative = false  
.Width = 7000  
.LeftMargin = 10*100 ' 10 mm  
.initialize(10,4) ' nombre de : lignes, colonnes  
end with
```

Sur OOO version 1.1.0 la largeur obtenue est plus grande que celle indiquée.
La propriété `RightMargin` est ignorée en orientation `LEFT_AND_WIDTH`.

Tableau étalé entre les marges gauche et droite

```
with MaTable  
.HoriOrient = com.sun.star.text.HoriOrientation.NONE  
.LeftMargin = 20*100 ' 20 mm  
.RightMargin = 35*100 ' 15 mm  
.initialize(10,4) ' nombre de : lignes, colonnes  
end with
```

Les propriétés `IsWidthRelative`, `RelativeWidth`, `Width`, sont ignorées.

h.d) Se déplacer dans un tableau

Pour écrire dans les cellules d'un tableau, il faut savoir comment se positionner

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

128/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

sur une des cellules. Dans Writer, la cellule d'un tableau est repérée par son adresse alphanumérique, exemple " B7 ", qui est le même repérage que dans une feuille de tableur Calc. Ceci est valable pour des tableaux simples constitués d'une grille régulière de lignes et colonnes. Par exemple, on écrit un texte simple dans une cellule particulière ainsi :

```
Dim UneCellule As Object
UneCellule = MaTable.getCellByName("G3")
UneCellule.String = "un petit texte"
```

Se déplacer avec un curseur de cellule

On peut utiliser un curseur de cellule, notion identique à celle [vue avec Calc](#).

La macro suivante écrit dans toutes les cellules d'un tableau dont on connaît le nombre de lignes et colonnes (9 et 5 dans l'exemple), en allant de gauche à droite sur chaque ligne, et de la première à la dernière ligne :

```
Dim MonCurseurCellule As Object
Dim UneCellule As Object
Dim x1 As Integer, y1 As Integer

MonCurseurCellule = MaTable.createCursorByCellName("A1")
for x1 = 0 to 4
  for y1 = 0 to 8
    UneCellule = MaTable.getCellByName(MonCurseurCellule.RangeName)
    UneCellule.String = Str(x1*100 +y1)
    MonCurseurCellule.goRight(1, false)
  next y1
next x1
```

On définit le curseur de cellule en lui affectant simultanément une adresse initiale de cellule, ici : A1. Ce curseur sera déplacé de cellule en cellule avec la méthode `goRight`, qui descend d'une ligne en fin de ligne précédente.

Une cellule ne peut être obtenue qu'avec son adresse alphanumérique, en argument de `getCellByName`. La propriété `RangeName` la fournit à partir du curseur de cellule.

Modifier la propriété `String` d'une cellule de tableau est la solution la plus simple pour écrire un texte dans la cellule, mais elle fait perdre le formatage par défaut de la cellule. Nous verrons plus loin une méthode plus élaborée.

Liste des instructions pour le curseur

Le deuxième argument de `goRight` signifie :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

129/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- false = déplacer le curseur
- true = déplacer le curseur en sélectionnant plusieurs cellules.

Les instructions pour déplacer le curseur ont une syntaxe identique :

Méthode	Signification
<code>goRight(n, false)</code>	déplacer de n cellules à droite, avec saut de ligne éventuel
<code>goLeft(n, false)</code>	déplacer de n cellules à gauche, avec saut de ligne éventuel
<code>goUp(n, false)</code>	déplacer de n cellules vers le haut
<code>goDown(n, false)</code>	déplacer de n cellules vers le bas
<code>gotoStart(false)</code>	déplacer à la première cellule, en haut à gauche
<code>gotoEnd(false)</code>	déplacer à la dernière cellule, en bas à droite

Se déplacer sans curseur de cellule

L'utilisation d'un curseur de cellule est assez pénible quand on souhaite accéder à des cellules dont on connaît les coordonnées en ligne et colonne. La macro suivante reprend l'exemple précédent, mais avec une méthode d'accès direct à chaque cellule :

```
Dim UneCellule As Object
Dim x1 As Integer, y1 As Integer

for x1 = 0 to 4 ' index de ligne, style Calc
  for y1 = 0 to 8 ' index de colonne, style Calc
    UneCellule = MaTable.getCellByName(TextCell(x1, y1))
    UneCellule.String = Str(x1*100 + y1)
  next y1
next x1
```

La fonction *TextCell* n'est pas dans l'API, c'est une fonction que vous pouvez écrire ainsi :

```
rem conversion d'index ligne/colonne vers le nom de la cellule
Function TextCell(Ligne As Integer, Colonne As Integer) As String
if (Colonne > 25) or (Colonne < 0) or (Ligne < 0) then
  TextCell = "???"
else
  TextCell = Chr(ASC("A") + Colonne) + LTrim(Str(Ligne + 1))
end if
End Function
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

130/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Elle convertit une coordonnée numérique ligne/colonne (comptées à partir de zéro) en coordonnée alphanumérique. Elle fonctionne pour des tableaux ayant moins de 26 colonnes, ce qui devrait être suffisant pour vos besoins...

Zone de cellules

Nous pouvons sélectionner plusieurs cellules d'un tableau en déplaçant le curseur de cellule avec l'argument `true`. Exemple :

```
MonCurseurCellule = MaTable.createCursorByCellName("B3")
MonCurseurCellule.goRight(1, true)
MonCurseurCellule.goDown(1, true)
print MonCurseurCellule.Rangename
```

Si vous exécutez la séquence, `print` vous affichera " C4:B3 ". La zone sélectionnée est définie par les coordonnées des cellules en diagonale (B3:C4 serait équivalent).

Le curseur de cellule pointe maintenant une zone de cellules. Avec un tel curseur on peut effectuer des opérations communes à toutes les cellules de la zone.

Notez que `createCursorByCellName` ne sait que pointer vers une seule cellule, pas une zone.

h.e) Ecrire du texte dans une cellule de tableau

Ayant obtenu accès à une cellule par les moyens décrits plus haut, on peut y écrire un texte à l'aide d'un curseur d'écriture à l'intérieur de la cellule. Les possibilités d'écriture sont exactement les mêmes que pour écrire du texte ordinaire. Comme nous l'avons vu, la propriété `String` remplace tout le contenu d'une cellule par un texte, ou permet de récupérer le texte brut de la cellule; les méthodes `getString` et `setString` n'existent pas.

```
Dim CurseurDansCellule As Object
Dim FinParagraphe As Integer
FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK
UneCellule = MaTable.getCellByName("B3") ' (par exemple)

CurseurDansCellule = UneCellule.createTextCursor
UneCellule.insertString(CurseurDansCellule, "OpenOffice est", false)
UneCellule.insertControlCharacter(CurseurDansCellule, FinParagraphe, false)
CurseurDansCellule.CharWeight = com.sun.star.awt.FontWeight.BOLD
UneCellule.insertString(CurseurDansCellule, " formidable !", false)
```

Tout le reste est identique, caractères spéciaux, style, formatage. Reprenez les chapitres correspondants en remplaçant *MonCurseur* par *CurseurDansCellule*.

Si vous avez défini un signet dans une cellule d'un tableau, vous devez récupérer

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

l'objet cellule ainsi :

```
Dim MonDocument As Object, Mazone As Object
Dim CurseurDansCellule As Object, UneCellule As Object

MonDocument = ThisComponent
Mazone = MonDocument.Bookmarks("signet3").getAnchor
UneCellule = Mazone.Cell
CurseurDansCellule = UneCellule.createTextCursorByRange(Mazone)
UneCellule.insertString(CurseurDansCellule, " formidable !", false)
```

h.f) Somme des cellules d'un tableau

L'exemple suivant va remplir avec des nombres les cellules A2, A3, A4 d'un tableau, puis mettre dans A5 une formule calculant la somme de ces cellules.

```
UneCellule = MaTable.getCellByName("A2")
UneCellule.Value = 22.5
UneCellule = MaTable.getCellByName("A3")
UneCellule.Value = -84
UneCellule = MaTable.getCellByName("A4")
UneCellule.Value = 47/3
UneCellule = MaTable.getCellByName("A5")
UneCellule.Formula = "sum(<A2:A4>)"
```

Remarque

La propriété `Value` est nécessaire pour remplir la cellule avec une valeur numérique, et non pas un texte. Un nombre comportant des décimales doit être écrit à l'anglaise (avec un point décimal), il sera affiché dans le document texte en respectant la règle de localisation. L'expression `47/3` est calculée par Basic, et son résultat numérique mis dans la propriété `Value` de la cellule.

h.g) Accéder à une ligne

La propriété `Rows` renvoie la collection des lignes du tableau. On accède à chaque ligne par son numéro d'ordre (ligne zéro pour la première ligne). Cet exemple cite les principales propriétés; il force la hauteur de la 31ème ligne à 20 millimètres, et en colore le fond :

```
Dim LignesCellules As Object

LignesCellules = MaTable.Rows
LignesCellules(30).IsAutoHeight = false
LignesCellules(30).Height = 2000 ' 20 mm
LignesCellules(30).BackColor = RGB(255,255,204)
```

Pour forcer la hauteur de ligne il est nécessaire de désactiver la propriété `IsAutoHeight`, qui par

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

132/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

défaut adapter la hauteur au contenu des cellules de la ligne. La hauteur est exprimée en 1/100 mm.

h.h) Colorer le fond d'un tableau

Tout le tableau

```
MaTable.BackColor = RGB(255,255,204)
```

Une seule cellule

```
UneCellule.BackColor = RGB(255,255,204)
```

Une ligne

```
LignesCellules(0).BackColor = RGB(255,255,204) ' la première ligne
```

Une colonne

Désolé, il n'y a pas de propriété pour cette entité !

Solution : sélectionner la zone de cellules correspondant à la colonne. Exemple pour la colonne D d'un tableau de 5 lignes :

```
MonCursurCellule = MaTable.createCursorByCellName("D1")  
MonCursurCellule.goDown(4, true)  
MonCursurCellule.BackColor = RGB(255,255,204)
```

h.i) Insérer plusieurs tableaux

A chaque insertion d'un tableau il est nécessaire d'obtenir un nouveau `TextTable`. Il faut aussi réinitialiser à chaque fois les propriétés du tableau. Dans cet exemple on insère deux tableaux de mêmes dimensions et couleur de fond, dans deux paragraphes du document Writer :

```
MaTable = MonDocument.createInstance("com.sun.star.text.TextTable")  
MaTable.initialize(5,4)  
MaTable.BackColor = RGB(255,255,204)  
MonTexte.insertTextContent(MonCursur, MaTable, false)  
  
MonCursur.gotoNextParagraph(false)  
MonCursur.gotoNextParagraph(false)  
MonCursur.gotoNextParagraph(false)  
  
MaTable = MonDocument.createInstance("com.sun.star.text.TextTable")  
MaTable.initialize(5,4)  
MaTable.BackColor = RGB(255,255,204)  
MonTexte.insertTextContent(MonCursur, MaTable, false)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

133/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

h.j) Trouver un tableau par son nom

Vous pouvez simplifier considérablement l'écriture de texte par macro si vous remplissez un document pré-défini (ou un modèle de document) qui contiendra la structure fixe du document, et des tableaux déjà formatés mais vides, qui seront remplis par macro. Rappelons que chaque tableau possède un nom (par défaut : tableau1, tableau2, etc, par ordre de création). Avec le navigateur renommez chaque tableau avec un nom plus spécifique.

Ensuite, il suffit de désigner dans la macro le tableau par son nom :

```
MaTable = MonDocument.TextTables.getByName("jolieTable")
```

Maintenant vous pouvez utiliser toutes les instructions précédentes pour remplir le tableau ou le modifier. Evidemment la macro doit connaître le nom du tableau et le type d'information à y insérer. Le nom du tableau doit être écrit exactement comme dans sa définition.

L'instruction déclenchera une exception s'il n'existe aucun tableau de ce nom. Vous pouvez tester l'existence du tableau ainsi :

```
if MonDocument.TextTables.hasByName("jolieTable") then  
    rem le tableau existe  
end if
```

Enfin, on renomme un tableau très simplement :

```
MaTable = MonDocument.TextTables.getByName("jolieTable")  
MaTable.Name = "MonbeauTableau"
```

Un nom de tableau ne doit pas comporter de caractère espace.

h.k) Tableaux irréguliers

Prenons comme exemple un tableau initial de 5 lignes et 4 colonnes, bien régulier.

A1	B1	C1	D1
A2	B2	C2	D2
A3	B3	C3	D3
A4	B4	C4	D4
A5	B5	C5	D5

On a indiqué dans chaque cellule sa coordonnée alphanumérique.

Fusionner des cellules

Pour des questions d'esthétique ou de titrage il est parfois nécessaire de fusionner

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

134/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

plusieurs cellules contiguës. Pour cela on sélectionne la zone des cellules à fusionner, puis on utilise la méthode `mergeRange` :

```
MonCurseurCellule = MaTable.createCursorByCellName("B3")
MonCurseurCellule.goRight(1, true)
MonCurseurCellule.goDown(1, true)
MonCurseurCellule.mergeRange ' fusionner la zone B3:C4
```

Si les cellules sélectionnées contenaient du texte, les paragraphes seront accolés dans la cellule résultante.

Le problème, c'est que les coordonnées de cellules sont maintenant assez spéciales :

A1	B1	C1	D1
A2	B2	C2	D2
A3.1.1	B3		C3.1.1
A3.1.2			C3.1.2
A4	B4	C4	D4

En fait, OpenOffice considère maintenant qu'il s'agit d'un tableau de 4 lignes, dont on a scindé horizontalement la cellule A3 et la cellule C3. Pour désigner une "sous-cellule" on utilise la coordonnée alphanumérique de la cellule d'origine, et on lui rajoute le rang de colonne et le rang de ligne de la sous-cellule dans la cellule d'origine. Et ceci en comptant à partir de 1 ! Dans le tableau en exemple, la sous-cellule A3.1.2 est sur la première colonne et la deuxième ligne dans le quadrillage de l'ancienne cellule A3.

Astuce

Si vous cliquez à l'intérieur d'une cellule de tableau, vous verrez s'afficher ses coordonnées en bas à droite dans la fenêtre Writer ! Très pratique pour s'y retrouver dans les tableaux complexes.

Scinder une cellule

A partir d'un curseur de cellule pointant sur une cellule ou une zone de cellules, on peut scinder horizontalement ou verticalement chaque cellule initiale en plusieurs sous-cellules :

```
MonCurseurCellule.splitRange(2, true) ' scinder horizontalement
```

Le premier argument est le nombre de sous-cellules créées, dans l'exemple c'est 2, donc on se retrouve avec 3 sous-cellules par cellule scindée ! Le deuxième argument vaut :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

135/186

Programmation Macro OpenOffice

Statut : En cours

Formation OpenOffice

Révision : 1.1

- `false` pour scinder verticalement
- `true` pour scinder horizontalement

Résultat d'une scission horizontale :

A1	B1	C1	D1
A2	B2	C2	D2
A3	B3.1.1	C3.1.1	
	B3.1.2	C3.1.2	
	B3.1.3	C3.1.3	
A4	B4.1.1	C4.1.1	D4
	B4.1.2	C4.1.2	
	B4.1.3	C4.1.3	
A5	B5	C5	D5

Résultat d'une scission verticale :

A1	B1			C1			D1
A2	B2			C2			D2
A3	B3	C3	D3	E3	F3	G3	H3
A4	B4	C4	D4	E4	F4	G4	H4
A5	B5			C5			D5

Ici la méthode de repérage n'est pas cohérente avec la précédente, et ne correspond pas à ce qui est dit dans l'API à `getCellByName`. (Ceci est vu sur OOo version 1.0.2 et API daté du 04/02/2003)

Généralisation

Les opérations de scission ou fusion peuvent être répétées sur les nouvelles cellules. Mais l'adressage devient très complexe à gérer.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

136/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

i) Les cadres dans un document Writer

En anglais, le mot “ cadre ” se traduit par “ frame ”.

i.a) Insérer un cadre

Voici comment insérer un cadre, en supposant le document Writer ouvert, avec un curseur d’écriture positionné là où vous voulez insérer le cadre :

```
Dim MonCadre As Object  
  
MonCadre = MonDocument.CreateInstance("com.sun.star.text.TextFrame")  
  
MonCadre.Width = 10400 ' 104 mm largeur  
MonCadre.Height = 2530 ' 25,3 mm de haut  
  
MonTexte.InsertTextContent( MonCurseur, MonCadre, false)
```

L’argument de `createInstance` est une “ formule magique ” à recopier scrupuleusement.

La méthode `insertTextContent` permet d’insérer un objet quelconque dans le texte, à la position du curseur. Ici, l’objet est un cadre. Le troisième argument signifie :

- false = insérer dans le texte
- true = écraser dans le texte

L’exemple utilisait le minimum d’instructions. En pratique il faudrait préciser :

1. si la hauteur du cadre s’adapte à son contenu, ou non
2. par rapport à quoi le cadre se positionne (l’ancrage du cadre),
3. sa position absolue par rapport à l’ancrage, ou
4. sa position relative définie par :
 - i. la zone où peut se placer le cadre,
 - ii. sa position horizontale dans cette zone (à gauche, au centre, à droite),
 - iii. sa position verticale dans cette zone (en haut, au centre, en bas).

i.b) Les cadres élastiques

La hauteur du cadre peut s’adapter ou non au contenu du cadre, par exemple si on y inscrit du texte sur plusieurs lignes. Ceci est défini par la propriété `SizeType` :

```
MonCadre.SizeType = com.sun.star.text.SizeType.FIX
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

137/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Les différentes valeurs possibles sont les [horribles constantes](#) suivantes :

Constante	Signification
VARIABLE	La hauteur dépend du contenu
FIX	La hauteur est fixe, indiquée par <code>Height</code>
MIN	La hauteur est au minimum celle indiquée par <code>Height</code> (valeur par défaut)

i.c) Les différents ancrages de cadre

Le choix du type d'ancrage se fait ainsi :

```
MonCadre.AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
```

Les valeurs possibles d'ancrage sont des [horribles constantes](#), listées dans le tableau ci-après.

Constante	Signification
<code>AT_PARAGRAPH</code>	Ancrage par rapport au paragraphe pointé par le curseur (valeur par défaut)
<code>AS_CHARACTER</code>	Ancrage comme si le cadre était un caractère; la hauteur de la ligne s'adapte à la taille du cadre
<code>AT_PAGE</code>	Ancrage par rapport à la page <code>AnchorPageNo</code> contient le numéro de page; par défaut c'est la page où se trouve le curseur d'écriture.
<code>AT_FRAME</code>	Ancrage dans un cadre <code>AnchorFrame</code> contient le cadre qui servira d'ancrage; voir exemple plus loin
<code>AT_CHARACTER</code>	Ancrage par rapport au caractère pointé par le curseur

i.d) Concepts utilisés pour le positionnement du cadre

Un cadre est positionné :

1. soit de manière absolue, en précisant la distance en 1/100 de millimètre entre le coin haut-gauche de l'ancre et le coin haut-gauche du cadre.
2. soit de manière relative, c'est-à-dire en décrivant sa situation par rapport à une zone qui englobe le cadre. Cette zone englobante doit aussi être précisée.

Le positionnement absolu ou relatif peut être choisi indépendamment pour la

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

138/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

position horizontale et pour la position verticale.

Les nombreuses combinaisons possibles d'ancrage / zone englobante / position relative donnent des résultats parfois identiques, ou visibles seulement avec des tailles de cadre et de paragraphe compatibles. Faites des essais.

Si vous insérez manuellement un cadre dans un texte, un clic droit sur le cadre donne accès aux propriétés : ancrage et alignement. L'alignement correspond au positionnement relatif du cadre.

i.e) Positionnement absolu du cadre

Positionnement horizontal :

```
MonCadre.HoriOrient = com.sun.star.text.HoriOrientation.NONE  
MonCadre.HoriOrientPosition = 2500 ' 25 mm
```

Positionnement vertical :

```
MonCadre.VertOrient = com.sun.star.text.VertOrientation.NONE  
MonCadre.VertOrientPosition = 5500 ' 55 mm
```

Les [horribles constantes](#) xxx.NONE servent à préciser que le positionnement est absolu.

i.f) Positionnement horizontal relatif

Zone englobante

Cette zone englobante est spécifiée par la propriété `HoriOrientRelation` qui utilise des [horribles constantes](#), exemple :

```
MonCadre.HoriOrientRelation = com.sun.star.text.RelOrientation.PAGE_LEFT
```

Les valeurs possibles pour `HoriOrientRelation` dépendent du type d'ancrage.

Valeurs possibles pour un ancrage `AT_PARAGRAPH` :

Constante	Signification
<code>FRAME</code>	Le paragraphe entier, y compris ses marges
<code>PRINT _AREA</code>	Le paragraphe entier, sans ses marges (valeur par défaut)
<code>PAGE_ LEFT</code>	Dans la marge de gauche de la page; en principe le cadre doit être assez petit pour tenir à l'intérieur de la marge
<code>PAGE_ RIGHT</code>	Dans la marge de droite de la page; même remarque

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

139/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
<code>FRAME_LEFT</code>	Dans la marge de gauche du paragraphe; même remarque
<code>FRAME_RIGHT</code>	Dans la marge de droite du paragraphe; même remarque

Valeurs possibles pour un ancrage `AT_PAGE` :

Constante	Signification
<code>PAGE_LEFT</code>	Dans la marge de gauche de la page; en principe le cadre doit être assez petit pour tenir à l'intérieur de la marge
<code>PAGE_RIGHT</code>	Dans la marge de droite de la page; même remarque
<code>PAGE_FRAME</code>	La page entière, y compris ses marges
<code>PAGE_PRINT_AREA</code>	La page entière sans ses marges (valeur par défaut)

Valeurs possibles pour un ancrage `AT_FRAME` :

- comme pour `AT_PARAGRAPH`, mais dans ce cas il s'agit du paragraphe de texte à l'intérieur du premier cadre.

Situation par rapport à la zone englobante

La propriété `HoriOrient` définit comment le cadre est positionné horizontalement, par rapport à la zone englobante. Elle utilise des [horribles constantes](#), exemple :

```
MonCadre.HoriOrient = com.sun.star.text.HoriOrientation.LEFT
```

Les valeurs possibles :

Constante	Signification
<code>RIGHT</code>	cadre aligné sur la marge de droite
<code>LEFT</code>	cadre aligné sur la marge de gauche

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

140/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
CENTE R	cadre centré sur l'espace disponible

i.g) Positionnement vertical relatif

Zone englobante

Cette zone englobante est spécifiée par la propriété `VertOrientRelation` qui utilise des [horribles constantes](#), exemple :

```
MonCadre.VertOrientRelation = com.sun.star.text.RelOrientation.FRAME
```

Les valeurs possibles pour `VertOrientRelation` dépendent du type d'ancrage.

Valeurs possibles pour un ancrage `AT_PARAGRAPH` :

Constante	Signification
FRAME	Le paragraphe entier, y compris ses marges
PRINT _AREA	Le paragraphe entier, sans ses marges (valeur par défaut)

Valeurs possibles pour un ancrage `AT_PAGE` :

Constante	Signification
PAGE_ FRAME	La page entière, y compris ses marges
PAGE_ PRINT _AREA	La page entière sans ses marges (valeur par défaut)

Valeurs possibles pour un ancrage `AT_FRAME` :

- comme pour `AT_PARAGRAPH`, mais dans ce cas il s'agit du paragraphe de texte à l'intérieur du premier cadre.

Situation par rapport à la zone englobante

La propriété `VertOrient` définit comment le cadre est positionné verticalement, par rapport à la zone englobante. Elle utilise des [horribles constantes](#), exemple :

```
MonCadre.VertOrient = com.sun.star.text.VertOrientation.TOP
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

141/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Valeurs possibles pour une zone englobante `AT_PAGE` :

Constante	Signification
<code>TOP</code>	cadre en haut de page
<code>CENTER</code>	cadre au centre de la page
<code>BOTTOM</code>	cadre en bas de la page

Valeurs possibles pour une zone englobante `AT_PARAGRAPH` et orientation relative `PRINT_AREA` :

Constante	Signification
<code>TOP</code>	le haut du cadre est centré sur le haut de la ligne de texte
<code>CENTER</code>	le cadre est centré sur le milieu de la ligne de texte
<code>BOTTOM</code>	le bas du cadre est centré sur le bas de la ligne de texte

i.h) Exemple complet d'insertion de cadre

Dans cet exemple on crée un petit cadre, situé dans la marge gauche de la page, positionné à droite, et à mi-hauteur de la page. Puis on écrit la lettre X dans le cadre.

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object
Dim MonCadre As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor
rem - - positionnez le curseur où vous voulez - -

MonCadre = MonDocument.createInstance("com.sun.star.text.TextFrame")

With MonCadre
    .Width = 1000
    .Height = 800
    .SizeType = com.sun.star.text.SizeType.FIX
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
    .VertOrient = com.sun.star.text.VertOrientation.CENTER
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

142/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
.VertOrientRelation = com.sun.star.text.RelOrientation.PAGE_FRAME
.HoriOrient = com.sun.star.text.HoriOrientation.RIGHT
.HoriOrientRelation = com.sun.star.text.RelOrientation.PAGE_LEFT
End With
MonTexte.insertTextContent( MonCurseur, MonCadre, false)
MonCadre.Text.String = "X"
```

i.i) Ecrire du texte dans un cadre

Toutes les possibilités d'écriture de texte qui existent pour le texte principal sont aussi disponibles pour écrire dans un cadre. Il suffit d'utiliser une variable texte et un curseur d'écriture comme ceci :

```
Dim MonTexte1 As Object, MonCurseur1 As Object
rem - - initialisation et insertion du cadre - -
MonTexte1 = MonCadre.Text
MonCurseur1 = MonTexte1.createTextCursor
MonTexte1.insertString(MonCurseur1, "Texte dans le cadre", false)
```

i.j) Insérer plusieurs cadres

A chaque insertion d'un cadre il est nécessaire d'obtenir un nouveau `TextFrame`, même si on insère plusieurs fois le même cadre. Il faut aussi réinitialiser à chaque fois les propriétés du cadre. Dans cet exemple on insère deux cadres de mêmes dimensions, couleur, position, dans deux paragraphes du document Writer :

```
MonCadre = MonDocument.createInstance("com.sun.star.text.TextFrame")
With MonCadre
  .Height = 1500
  .Width = 4000
  .BackColor = RGB(255,200,255)
  .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PARAGRAPH
  .HoriOrient = com.sun.star.text.HoriOrientation.NONE
  .HoriOrientPosition = 2500 ' unité : 1/100 mm
End With
MonTexte.insertTextContent( MonCurseur, MonCadre, false)
MonCadre.Text.String = "insertion 1"

MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextParagraph(false)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

143/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonCadre = MonDocument.CreateInstance("com.sun.star.text.TextFrame")
With MonCadre
    .Height = 1500
    .Width = 4000
    .BackColor = RGB(255,200,255)
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PARAGRAPH
    .HoriOrient = com.sun.star.text.HoriOrientation.NONE
    .HoriOrientPosition = 2500 ' unité : 1/100 mm
End With
MonTexte.InsertTextContent( MonCasseur, MonCadre, false)
MonCadre.Text.String = "insertion 2"
```

i.k) Exemple de cadre dans un cadre

On peut parfaitement positionner un cadre dans un autre cadre, supposé plus grand.

```
Dim MonDocument As Object, MonTexte As Object, MonTexte1 As Object
Dim MonCasseur As Object, MonCasseur1 As Object
Dim MonCadre1 As Object, MonCadre2 As Object
Dim FinParagraphe As Integer
FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCasseur = MonTexte.CreateTextCursor
rem - - positionnez le curseur où vous voulez - -

MonCadre1 = MonDocument.CreateInstance("com.sun.star.text.TextFrame")

With MonCadre1
    .Width = 9400 ' 94 mm largeur
    .Height = 2530 ' 25,3 mm de haut
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PARAGRAPH
End With

MonTexte.InsertTextContent( MonCasseur, MonCadre1, false)
MonTexte1 = MonCadre1.Text
with MonTexte1
    MonCasseur1 = .CreateTextCursor
    .insertString(MonCasseur1, "Texte dans cadre1", false)
    .insertControlCharacter(MonCasseur1, FinParagraphe, false)
    .insertString(MonCasseur1, "Texte ligne 2", false)
    .insertControlCharacter(MonCasseur1, FinParagraphe, false)
    .insertString(MonCasseur1, "Texte ligne 3", false)
end with
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

144/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonCadre2 = MonDocument.createInstance("com.sun.star.text.TextFrame")

With MonCadre2
  .Width = 1000
  .Height = 1000
  .AnchorFrame = MonCadre1
  .AnchorType = com.sun.star.text.TextContentAnchorType.AT_FRAME
  .HoriOrient = com.sun.star.text.HoriOrientation.LEFT
End With
MonTexte1.insertTextContent( MonCurseur1, MonCadre2, false)
MonCadre2.Text.String = "X"
```

Les instructions `With` servent à éviter les répétitions de *MonTexte1* ou *MonCadre2*.

On pourrait aussi bien créer un *MonCurseur2* pour insérer des textes entiers, et même insérer un troisième cadre, et ainsi de suite, à condition d'avoir encore de la place sur la page...

i.l) Couleur du fond du cadre

Par défaut, le cadre hérite de la couleur de fond de la zone d'insertion. Pour imposer une couleur :

```
MonCadre.BackColor = RGB(255,200,255)
```

i.m) Trouver un cadre par son nom

Vous pouvez simplifier considérablement l'écriture de texte par macro si vous remplissez un document pré-défini (ou un modèle de document) qui contiendra la structure fixe du document, et des cadres déjà formatés mais vides, qui seront remplis par macro. Rappelons que chaque cadre possède un nom (par défaut : cadre1, cadre2, etc, par ordre de création). Avec le Navigateur renommez chaque cadre avec un nom plus spécifique.

Ensuite, il suffit de désigner dans la macro le cadre par son nom :

```
MonCadre = MonDocument.TextFrames.getByNamed("CadreBleu")
```

Maintenant vous pouvez utiliser toutes les instructions précédentes pour remplir le cadre ou le modifier. Evidemment la macro doit connaître le nom du cadre et le type d'information à y insérer. Le nom du cadre doit être écrit exactement comme dans sa définition.

L'instruction déclenchera une exception s'il n'existe aucun cadre de ce nom. Vous pouvez tester l'existence du cadre ainsi :

```
if MonDocument.TextFrames.hasByName("CadreBleu") then
  rem le cadre existe
end if
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Enfin, on renomme un cadre très simplement :

```
MonCadre = MonDocument.TextFrames.getByName("CadreBleu")  
MonCadre.Name = "MonBeauCadre"
```

Un nom de cadre ne doit pas comporter de caractère espace.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

146/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

j) Les dessins dans Writer

Ici, on appelle une forme (anglais : shape) un dessin élémentaire, par exemple un rectangle, une ellipse, etc.

j.a) La page de dessin de Writer

L'API signale que les formes dans Writer apparaissent sur une unique page de dessin (anglais : draw page). Il s'agit des feuilles que l'on peut voir dans un document Draw ou Impress.

En pratique, cette unique page de dessin pour Writer s'étend sur le document entier : vous pouvez insérer des formes sur un nombre quelconque de pages de texte. L'insertion d'une forme directement sur la page de dessin est possible, mais le positionnement est alors obligatoirement ancré à la page, ce qui est très restrictif.

Vous pouvez aussi insérer des dessins à l'intérieur de cadres du document Writer.

Pour obtenir la page de dessin d'un document Writer :

```
Dim PageDessin As Object
```

```
PageDessin = MonDocument.DrawPage
```

j.b) Insérer une forme dessinée

Voici comment insérer une forme rectangulaire, en supposant le document Writer ouvert, avec un curseur d'écriture positionné là où vous voulez insérer la forme :

```
Dim MonDocument As Object, MonTexte As Object  
Dim MonCurseur As Object  
Dim MaForme As Object  
Dim Taille1 As New com.sun.star.awt.Size
```

```
MonDocument = ThisComponent  
MonTexte = MonDocument.Text  
MonCurseur = MonTexte.createTextCursor
```

```
Taille1.Width = 1000  
Taille1.Height = 800
```

```
MaForme = MonDocument.createInstance("com.sun.star.drawing.RectangleShape")  
MaForme.Size = Taille1  
MonTexte.insertTextContent(MonCurseur, MaForme, false)
```

Contrairement aux cadres, il n'existe pas de propriété Width ou Height pour une forme. L'instruction `Dim Taille1` crée un objet de type Taille (en anglais : Size) qui servira à dimensionner

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

147/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

la future forme. Basic ne permet pas de modifier directement la propriété `Size` d'une forme.

L'argument de `createInstance` est une horrible constante qui précise le type de forme dessinée.

Les différentes formes de base sont listées dans le chapitre Dessiner des formes, qui est valable pour toutes les applications.

La méthode `insertTextContent` permet d'insérer un objet quelconque dans le texte, à la position du curseur. Ici, l'objet est une forme (un dessin élémentaire). Le troisième argument signifie :

- false = insérer dans le texte
- true = écraser dans le texte

L'exemple utilisait le minimum d'instructions. En pratique il faudrait préciser :

1. par rapport à quoi la forme se positionne (l'ancrage de la forme),
2. sa position absolue par rapport à l'ancre

j.c) Les différents ancrages d'une forme

Le choix du type d'ancrage se fait ainsi :

```
MaForme.AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
```

Les valeurs possibles d'ancrage sont des horribles constantes, listées dans le tableau ci-après.

Constante	Signification
<code>AT_PARAGRAPH</code>	Ancrage par rapport au paragraphe pointé par le curseur, marges comprises
<code>AS_CHARACTER</code>	Ancrage comme si la forme était un caractère; la hauteur de la ligne s'adapte à la taille de la forme (valeur par défaut)
<code>AT_PAGE</code>	Ancrage par rapport à la page entière avec ses marges <code>AnchorPageNo</code> contient le numéro de page; par défaut c'est la page où se trouve le curseur d'écriture.
<code>AT_CHARACTER</code>	Ancrage par rapport au caractère pointé par le curseur

j.d) Positionnement de la forme

Contrairement aux cadres, pour les formes seul le positionnement absolu est possible. Les distances sont exprimées en 1/100 de millimètres, mesurées du coin

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

haut-gauche de l'ancre au coin haut-gauche de la forme.

Bien que les instructions de positionnement absolu d'un cadre soient acceptées, il est préférable d'utiliser une méthode qui fait appel à une variable de position.

Exemple :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object
Dim MaForme As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor

Taille1.Width = 1000
Taille1.Height = 800
Position1.x = 5500
Position1.y = 12000

MaForme = MonDocument.CreateInstance("com.sun.star.drawing.RectangleShape")
With MaForme
    .Size = Taille1
    .Position = Position1
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
End With

MonTexte.insertTextContent( MonCurseur, MaForme, false)
```

j.e) Insérer plusieurs formes

A chaque insertion d'une forme il est nécessaire d'obtenir un nouveau `xxxShape`, même si on insère plusieurs fois le même type de forme. Il faut aussi réinitialiser à chaque fois les propriétés de la forme. Dans cet exemple on insère deux fois la même ellipse à la même position, dans la page 4 puis dans la page 1 du document Writer :

```
Taille1.Width = 5400
Taille1.Height = 2530
Position1.x = 5500
Position1.y = 12000

MaForme = MonDocument.CreateInstance("com.sun.star.drawing.EllipseShape")
With MaForme
    .Size = Taille1
    .Position = Position1
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

149/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
.FillColor = RGB(200, 150, 200)
.AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
.AnchorPageNo = 4
End With
MonTexte.insertTextContent( MonCursur, MaForme, false)
MaForme.String = "insertion 1"

MaForme = MonDocument.createInstance("com.sun.star.drawing.EllipseShape")
With MaForme
    .Size = Taille1
    .Position = Position1
    .FillColor = RGB(200, 150, 200)
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
    .AnchorPageNo = 1
End With
MonTexte.insertTextContent( MonCursur, MaForme, false)
MaForme.String = "insertion 2"
```

j.f) Autres fonctionnalités autour de la forme

Les propriétés `SurroundAnchorOnly`, `SurroundContour`, `ContourOutside`, `Opaque`, ne semblent pas avoir une influence sur l'adaptation du texte à la forme, quelque soit l'ancrage.

Les autres fonctionnalités sont identiques pour Writer et les autres applications. Elles sont décrites dans un [chapitre commun](#) :

- Propriétés communes
- Dessin de différentes formes
- Ecriture de texte dans la forme
- Donner un nom à une forme

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

k) Les images dans Writer

k.a) Insérer une image

Note : dans l'API version anglaise, le terme Graphic est utilisé pour désigner une image "bitmap" ou vectorielle. Il n'y a aucun rapport avec le concept français de graphique.

Voici une première méthode d'insertion d'image, qui suffit pour les images que vous n'avez pas à redimensionner, par exemple une petite image comme :



Supposons le document Writer ouvert, avec un curseur d'écriture positionné là où vous voulez insérer l'image :

```
Dim MonImage As Object
```

```
MonImage = MonDocument.CreateInstance("com.sun.star.text.GraphicObject")
```

```
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\tip.gif")
```

```
MonTexte.InsertTextContent(MonCurseur, MonImage, false)
```

L'argument de `createInstance` est une "formule magique" à recopier scrupuleusement.

La méthode `insertTextContent` permet d'insérer un objet quelconque dans le texte, à la position du curseur. Ici, l'objet est une image. Le troisième argument signifie :

- false = insérer dans le texte
- true = écraser dans le texte

Remarque

Ces instructions n'insèrent pas l'image dans le document Writer, mais seulement un lien vers l'image. Si vous déplacez, renommez, supprimez le fichier image référencé, le document affichera une case de lien brisé avec l'adresse URL du fichier.

Ceci est une limitation actuelle de l'API.

Pour effectivement insérer dans le document des images insérées par macro, suivre cette procédure :

1. Menu Edition > Liens...
 - sélectionner tous les liens d'images

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

151/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- cliquer sur le bouton Déconnecter; confirmer
2. Sauver le document

k.b) Positionner une image

Les principes de positionnement décrits pour les cadres sont applicables pour les images.

Dans cet exemple on insère une petite image, située dans la marge gauche de la page, positionnée à droite, et à mi-hauteur de la page :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonImage As Object
Dim MonCurseur As Object

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor

MonImage = MonDocument.CreateInstance("com.sun.star.text.GraphicObject")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\important.gif")

With MonImage
    .AnchorType = com.sun.star.text.TextContentAnchorType.AT_PAGE
    .VertOrient = com.sun.star.text.VertOrientation.CENTER
    .VertOrientRelation = com.sun.star.text.RelOrientation.PAGE_FRAME
    .HoriOrient = com.sun.star.text.HoriOrientation.RIGHT
    .HoriOrientRelation = com.sun.star.text.RelOrientation.PAGE_LEFT
End With

MonTexte.insertTextContent(MonCurseur, MonImage, false)
```

k.c) Dimensionner une image

La méthode d'insertion d'image que nous avons vue pose des problèmes pour redimensionner l'image. La deuxième méthode décrite ici permet le redimensionnement, mais oblige à toujours préciser les dimensions. Exemple :

```
Dim MonImage As Object
Dim Taille1 As New com.sun.star.awt.Size

MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonTexte.insertTextContent(MonCurseur, MonImage, false)
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Taille1.Width = 6000 ' largeur en 1/100 de mm
Taille1.Height = 6000 ' hauteur en 1/100 de mm
MonImage.Size = Taille1
```

Ces instructions n'insèrent pas l'image dans le document Writer, mais seulement un lien vers l'image. La remarque vue précédemment s'applique aussi ici.

Evidemment, si l'image originale n'a pas les mêmes proportions, l'image sur le document sera déformée. Pour conserver les proportions on rajoute quelques instructions :

```
Dim MonImage As Object, LeBitmap As Object
Dim Proportion As Double
Dim Taille1 As New com.sun.star.awt.Size

MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonTexte.InsertTextContent(MonCurseur, MonImage, false)

LeBitmap = MonImage.GraphicObjectFillBitmap
Taille1 = LeBitmap.Size ' taille en pixels !
Proportion = Taille1.Height / Taille1.Width
Taille1.Width = 6000 ' largeur en 1/100 de mm
Taille1.Height = Taille1.Width * Proportion
MonImage.Size = Taille1
```

Pour dimensionner l'image sans changer les proportions, on récupère les dimensions de l'image en pixels. Ceci n'est possible qu'après l'instruction `insertTextContent`. La proportion calculée sert à fixer la hauteur en fonction de la largeur. Ces deux dimensions sont en 1/100 de mm.

Remarque

Le positionnement relatif ne fonctionne pas avec cette méthode d'insertion d'image. Seul le positionnement absolu fonctionne.

k.d) Insérer plusieurs images

A chaque insertion d'image il est nécessaire d'obtenir un nouveau `GraphicObject`, même si on insère plusieurs fois la même image. Exemple, avec la méthode simple d'insertion :

```
MonImage = MonDocument.CreateInstance("com.sun.star.text.GraphicObject")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonTexte.InsertTextContent(MonCurseur, MonImage, false)

rem déplacer le curseur ailleurs dans le texte
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextParagraph(false)
MonCurseur.gotoNextParagraph(false)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

153/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonImage = MonDocument.CreateInstance("com.sun.star.text.GraphicObject")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonTexte.InsertTextContent(MonCurseur, MonImage, false)
```

k.e) Trouver une image par son nom

Pour pouvoir retrouver une image, un bon moyen est de la nommer. Une fois l'image insérée dans le document, il suffit d'utiliser la propriété `Name` :

```
MonImage.Name = "Mon chien"
```

Ensuite, on retrouve l'image avec les méthodes `getByName` et `hasByName` :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonImage As Object, LesImages As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent
LesImages = MonDocument.GraphicObjects

if LesImages.hasByName("Mon chien") then
    MonImage = LesImages.getByName("Mon chien")
    Taille1.Width = 6000 ' largeur en 1/100 de mm
    Taille1.Height = 3000 ' hauteur en 1/100 de mm
    MonImage.Size = Taille1
else
    print "Nom inexistant !"
end if
```

VII.1.8 Dessin Draw

Ici, on appelle une forme (anglais : shape) un dessin élémentaire, par exemple un rectangle, une ellipse, etc.

a) Trouver la page de dessin

a.a) Une page existante

Chaque page de dessin dans Draw comporte un onglet. Le nom des onglets est par défaut, dans la version localisée française, **Page 1**, **Page 2**, etc. Le numéro de page est le rang de l'onglet, de gauche à droite. Si vous déplacez avec la souris une page non renommée, le nom sur l'onglet va changer pour refléter sa position, ainsi que pour les pages suivantes non renommées.

Avec l'API on accède ainsi à une page :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

154/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Dim MonDocument As Object
Dim UnePage As Object, lesPages As Object

MonDocument = ThisComponent
lesPages = MonDocument.Drawpages
print "Nombre de pages : "; lesPages.Count
UnePage = lesPages(0) ' Première page de dessin
```

`DrawPages` fournit l'ensemble des pages du document. `Count` donne le nombre de pages existantes. Chaque page est accessible par un index.

On aurait pu se passer de la variable *lesPages* en écrivant la dernière ligne :

```
UnePage = MonDocument.Drawpages(0) ' Première page de dessin
```

Attention

Pour l'API les pages sont numérotées à partir de zéro, dans l'ordre affiché par l'interface Draw (toutes les pages, renommées ou non). Ainsi si l'utilisateur change l'ordre des pages, la macro écrira obstinément sur la page de rang n.

Une meilleure méthode consiste à récupérer la page ayant un nom donné :

```
Dim MonDocument As Object
Dim UnePage As Object, lesPages As Object

MonDocument = ThisComponent
lesPages = MonDocument.Drawpages
UnePage = lesPages.getByname("essai-1")
```

On peut tester si une page d'un nom donné existe :

```
if lesPages.hasbyname("essai-1") then
    rem la page existe bien
end if
```

Attention

L'API permet de récupérer le nom de l'onglet avec la propriété `Name` de la page. Pour les pages non renommées, le nom récupéré est : `page1`, `page2`, etc. L'espace a disparu, et le nom commence par une minuscule.

Draw sait si l'onglet est renommé, mais l'API ne fournit pas cette information.

a.b) Créer, renommer, dupliquer, supprimer une page

L'API concernant les pages de Draw n'est pas aussi élaboré que pour les feuilles de Calc.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

155/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Créer une page vierge

On insère la nouvelle page à droite d'une position donnée parmi les pages du document.

```
Dim MonDocument As Object
Dim NouvellePage As Object, lesPages As Object

MonDocument = ThisComponent
lesPages = MonDocument.Drawpages
' insère une page après la deuxième page
NouvellePage = lesPages.insertNewByIndex(1)
```

Dans cet exemple la nouvelle page aura pour index 2, et pour nom **Page 3**.

Renommer une page

Renommer une page revient à changer la valeur de la propriété `Name` de la page.

```
UnePage = lesPages.getByName("essai-1")
UnePage.Name = "Sommaire"
```

Déplacer une page

Les fonctions de l'API ne permettent pas de modifier l'ordre des pages.

Dupliquer une page

```
Dim MonDocument As Object
Dim UnePage As Object, lesPages As Object
Dim PageDupl As Object

MonDocument = ThisComponent
lesPages = MonDocument.Drawpages
UnePage = lesPages.getByName("essai-1")
PageDupl = MonDocument.Duplicate(UnePage)
```

La nouvelle page est la copie exacte de la page donnée en argument, avec son contenu. L'onglet de cette nouvelle page sera situé à droite de la page de référence, il aura pour nom **Page n** avec pour n le rang du nouvel onglet. On peut ensuite renommer la page.

Supprimer une page

```
Dim MonDocument As Object
Dim UnePage As Object, lesPages As Object

MonDocument = ThisComponent
lesPages = MonDocument.Drawpages
UnePage = lesPages.getByName("essai-1")
lesPages.remove(UnePage)
```

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

a.c) La page visible par l'utilisateur

Cet exemple obtient la page visible par l'utilisateur, affiche le nom de cette page, et rend visible la page " testPage " à la place.

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object, PageUsager As Object

MonDocument = ThisComponent
rem récupérer la page visible
PageUsager = MonDocument.CurrentController.CurrentPage
Print "La page affichée est : "; PageUsager.Name

UnePage = MonDocument.DrawPages.getByName("testPage")
rem changer la page visible
MonDocument.CurrentController.CurrentPage = UnePage
```

a.d) Propriétés d'une page de dessin

Une page de dessin possède des propriétés, essentiellement :

Propriété	Signification
Name	chaîne de caractères, le nom de la page affiché sur l'onglet (voir remarque)
BorderLeft	Marge de gauche en 1/100 de mm
BorderRight	Marge de droite en 1/100 de mm
BorderTop	Marge du haut en 1/100 de mm
BorderBottom	Marge du bas en 1/100 de mm
Height	Hauteur totale de la page en 1/100 de mm
Width	Largeur totale de la page en 1/100 de mm
Number	(en lecture seulement) Rang de la page dans l'ordre des onglets, numérotation à partir de 1
Orientation	Portrait : <code>com.sun.star.view.PaperOrientation.PORTRAIT</code> ou Paysage : <code>com.sun.star.view.PaperOrientation.LANDSCAPE</code>

Exemple d'utilisation :

```
Dim MonDocument As Object
Dim UnePage As Object
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

157/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")

With UnePage
  print "Marges G, D", .BorderLeft , .BorderRight
  print "Marges H, B", .BorderTop, .BorderBottom
  print "Dimensions de la page H, L", .Height, .Width
  print "Rang de la page", .Number
  Select Case .Orientation
    Case com.sun.star.view.PaperOrientation.PORTRAIT
      print "Portrait"
    Case com.sun.star.view.PaperOrientation.LANDSCAPE
      print "Paysage"
  end select
end with
```

b) Les dessins dans Draw

b.a) Insérer une forme dessinée

Cet exemple minimal insère un rectangle dans une page de dessin nommée "essai-1" :

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("essai-1")
Taille1.Width = 13400 ' 134 mm de large
Taille1.Height = 2530 ' 25,3 mm de haut

MaForme = MonDocument.createInstance("com.sun.star.drawing.RectangleShape")
With MaForme
  .Size = Taille1
  .FillColor = RGB(137, 111, 87)
End With
UnePage.add(MaForme)
```

L'instruction `Dim Taille1` crée un objet de type Taille (en anglais : Size) qui servira à dimensionner la future forme. Basic ne permet pas de modifier directement la propriété `Size` d'une forme.

L'argument de `createInstance` est une [horrible constante](#) qui précise le type de forme dessinée.

Les différentes formes de base sont listées dans le chapitre [Dessiner des formes](#), qui est valable pour toutes les applications.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

158/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Une fois le dessin ajouté à la page, on peut changer ses propriétés, listées plus loin.

b.b) Positionner la forme

L'exemple précédent a positionné la forme tout en haut à gauche dans la page.

Le positionnement fait appel à une variable intermédiaire, qui définit la position d'un point, en coordonnées x et y. Ce sera la position du coin haut-gauche de la forme. Les coordonnées sont exprimées en 1/100 de mm, relativement au coin haut-gauche de la page de dessin.

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("essai-1")
Taille1.Width = 13400 ' 134 mm de large
Taille1.Height = 2530 ' 25,3 mm de haut
Position1.x = 2500 ' 25 mm à droite du coin haut-gauche de la page
Position1.y = 5300 ' 53 mm plus bas que le coin haut-gauche de la page

MaForme = MonDocument.CreateInstance("com.sun.star.drawing.RectangleShape")
With MaForme
    .Size = Taille1
    .Position = Position1
    .FillColor = RGB(137, 111, 87)
End With
UnePage.add(MaForme)
```

Note

Si vous faites un clic droit sur une forme, et affichez sa position, les valeurs sont indiquées par rapport aux marges de la page. Alors que le positionnement par macro est relatif à la page entière.

b.c) Autres fonctionnalités des formes

Les autres fonctionnalités sont identiques pour Draw et les autres applications. Elles sont décrites dans un [chapitre commun](#) :

- Propriétés communes
- Dessin de différentes formes
- Ecriture de texte dans la forme

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

159/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

- Donner un nom à une forme

Si vous souhaitez aller plus loin dans la programmation de dessin dans Draw, consultez le site en anglais de Danny Brewer :

<http://kosh.datateamsys.com/~danny/OOo/DannysDrawPowerTools>

Vous y verrez notamment :

- comment programmer une “ tortue ” qui dessine en tirant des traits sur la page de dessin
- un dessin animé fait avec des cercles rebondissant sur les marges de la page de dessin
- de nombreux exemples de macros pour le dessin.

c) Les images dans Draw

c.a) Insérer une image

Note : dans l'API version anglaise, le terme Graphic est utilisé pour désigner une image “ bitmap ” ou vectorielle. Il n'y a aucun rapport avec le concept français de graphique.

Cet exemple insère une image :

```
Dim MonDocument As Object
Dim UnePage As Object
Dim MonImage As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent

UnePage = MonDocument.DrawPages(2) ' page de la troisième feuille Draw
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")

UnePage.add(MonImage)

Taille1.Width = 6000 ' largeur en 1/100 de mm
Taille1.Height = 6000 ' largeur en 1/100 de mm
MonImage.Size = Taille1
```

Si l'image originale n'a pas les mêmes proportions, l'image sur le document sera déformée. Nous verrons plus loin comment redimensionner en gardant les proportions originales.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

160/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Ces instructions n'insèrent pas l'image dans le document Draw, mais seulement un lien vers l'image. Si vous déplacez, renommez, supprimez le fichier image référencé, le document affichera une case de lien brisé avec l'adresse URL du fichier.

Ceci est une limitation actuelle de l'API.

Pour effectivement insérer dans le document des images insérées par macro, suivre cette procédure :

3. Menu Edition > Liens...

- sélectionner l'ensemble des liens images
- cliquer sur le bouton Déconnecter; confirmer

4. Sauver le document.

c.b) Positionner une image

L'ancrage et le positionnement d'une image et d'une forme sont identiques. Reportez-vous au chapitre concernant le [positionnement des formes](#) dans Draw.

c.c) Dimensionner une image

Pour dimensionner l'image sans changer les proportions, on récupère les dimensions de l'image en pixels. Ceci n'est possible qu'après l'instruction `PageDessin.add(MonImage)`. La proportion calculée sert à fixer la hauteur en fonction de la largeur. Ces deux dimensions sont en 1/100 de mm.

Cet exemple positionne l'image dans la feuille et redimensionne l'image à une largeur choisie, tout en gardant ses proportions :

```
Dim MonDocument As Object
Dim UnePage As Object
Dim MonImage As Object
Dim LeBitmap As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point
Dim Proportion As Double

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages(2) ' page de la troisième feuille Draw
MonImage = MonDocument.createInstance("com.sun.star.drawing.GraphicObjectShape")
MonImage.GraphicURL = ConvertToURL(" C:\Liste Images\Medor.jpg ")
Position1.x = 2500 ' 25 mm à droite du coin haut-gauche de la page
Position1.y = 5300 ' 53 mm plus bas que le coin haut-gauche de la page
UnePage.add(MonImage)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

161/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
LeBitmap = MonImage.GraphicObjectFillBitmap
Taille1 = LeBitmap.Size ' taille en pixels !
Proportion = Taille1.Height / Taille1.Width
Taille1.Width = 6000 ' largeur en 1/100 de mm
Taille1.Height = Taille1.Width * Proportion
MonImage.Size = Taille1
MonImage.Position = Position1
```

c.d) Insérer plusieurs images

A chaque insertion d'image il est nécessaire d'obtenir un nouveau **GraphicObjectShape**, même si on insère plusieurs fois la même image, comme dans l'exemple qui suit. Il comporte un sous-programme facilitant le redimensionnement.

```
Sub Insérer2ImagesIdentiques
Dim MonDocument As Object, LesFeuilles As Object
Dim PageDessin As Object
Dim MonImage As Object
Dim Posit1 As New com.sun.star.awt.Point

MonDocument = ThisComponent

PageDessin = MonDocument.DrawPages(2) ' page de la feuille Calc de rang 2
MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
Posit1.x = 3000
Posit1.y = 6000
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonImage.Position = Posit1
PageDessin.add(MonImage)
LargeProp(MonImage, 6000)

MonImage = MonDocument.CreateInstance("com.sun.star.drawing.GraphicObjectShape")
Posit1.x = 10000
Posit1.y = 8000
MonImage.GraphicURL = ConvertToURL("C:\Liste Images\Medor.jpg")
MonImage.Position = Posit1
PageDessin.add(MonImage)
LargeProp(MonImage, 3500)

End Sub

Sub LargeProp( UneImage As Object, Largeur As Long)
Dim LeBitmap As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Proportion As Double
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

162/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
LeBitmap = UneImage.GraphicObjectFillBitmap
Taille1 = LeBitmap.Size ' taille en pixels !
Proportion = Taille1.Height / Taille1.Width
Taille1.Width = Largeur ' largeur en 1/100 de mm
Taille1.Height = Taille1.Width * Proportion
UneImage.Size = Taille1
end sub
```

c.e) Trouver une image par son nom

Pour pouvoir retrouver une image, un bon moyen est de la nommer. Une fois l'image insérée dans le document, il suffit d'utiliser la propriété `Name` :

```
MonImage.Name = "Mon chien"
```

Toutes les images (et les formes) d'une page sont accessibles par un index, exemple :

```
UneForme = UnePage(3)
```

Ceci est peu pratique, mais dans Draw il n'existe pas de méthode API pour retrouver une forme par son nom.

Pas de problème, nous allons utiliser la fonction *FindObjectByName* décrite dans un autre [chapitre](#). Avec cette fonction, nous pouvons retrouver l'image et la modifier :

```
Dim MonDocument As Object, MonTexte As Object
Dim MonImage As Object
Dim UnePage As Object
Dim Taille1 As New com.sun.star.awt.Size

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")

MonImage = FindObjectByName(UnPage, "Mon chien")
if IsNull(MonImage) then
    print "Nom inexistant !"
else
    Taille1.Width = 6000 ' largeur en 1/100 de mm
    Taille1.Height = 3000 ' hauteur en 1/100 de mm
    MonImage.Size = Taille1
end if
```

Attention : cette méthode retrouve un objet quelconque sur la page, aussi évitez de donner le même nom à une forme et à une image de la même page.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

163/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

VII.1.9 Présentation Impress

a) Fonctions identiques à Draw

Les fonctionnalités de Draw, comme la notion de page, les dessins, les images, sont utilisables à l'identique pour une présentation Impress.

b) Fonctions spécifiques à Impress

On peut spécifier une action déclenchée par un clic sur un objet de la diapo en cours. Cette action peut consister à exécuter une macro, ce qui donne toutes possibilités.

Quand aux propriétés spécifiques aux formes dans Impress, elles sont relatives aux effets, et seront probablement seulement configurées à la conception des diapos.

VII.1.10 Les formes : fonctions communes à toutes les applications

a) Propriétés des formes

a.a) Couleur de fond de la forme

Sans instruction particulière, la forme est créée avec une couleur de fond uniforme, dont le coloris est une valeur par défaut. Pour imposer une autre couleur :

```
MaForme.FillColor = RGB(255,200,255)
```

Mais on peut avoir des couleurs non uniformes, ou pas de couleur, selon la valeur de la propriété `FillStyle`.

```
MaForme.FillStyle = com.sun.star.drawing.FillStyle.NONE
```

Les valeurs sont des [horribles constantes](#) :

Constante	Signification
NONE	transparent
SOLID	couleur uniforme(valeur par défaut)
GRADIENT	dégradé de couleur – consulter l'API pour détails
HATCH	quadrillage – consulter l'API pour détails

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

164/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Constante	Signification
BITMAP	utilisation d'une image – consulter l'API pour détails

Avec une couleur `SOLID` on peut régler la transparence avec la propriété `FillTransparence` :

```
MaForme.FillTransparence = 60 ' pourcentage
```

La valeur est en pourcentage, 0 pour une couleur opaque, 100 pour une couleur totalement transparente.

a.b) Forme avec une ombre

L'ombre (anglais : shadow) sert à présenter la forme comme flottant au-dessus de la feuille.

```
With MaForme
  .Shadow = true
  .ShadowColor = RGB(100, 100, 100)
  .ShadowXDistance = -250
  .ShadowYDistance = 250
end with
```

Le paramètre `Shadow` est un booléen qui indique qu'une ombre est nécessaire (sans ombre par défaut)

Le paramètre `ShadowColor` précise la couleur de l'ombre (noire par défaut)

Les deux suivants précisent l'extension de l'ombre par rapport à la forme, exprimée en 1/100 de mm. Une valeur négative sur X met l'ombre à gauche, une valeur négative sur Y met l'ombre en bas. On choisit ainsi l'angle de l'éclairage.

a.c) Rotation de la forme

```
MaForme.RotateAngle = 4500 ' 25 degrés
```

L'angle est exprimé en 1/100 de degré par rapport à l'horizontale; une valeur positive correspond au sens trigonométrique (le sens inverse des aiguilles d'une montre).

a.d) Effet parallélogramme

Cet effet transforme un rectangle en parallélogramme.

```
MaForme.ShearAngle = 3000 ' 30 degrés de cisaillement
```

L'angle est exprimé en 1/100 de degré par rapport au côté vertical d'un rectangle; une valeur positive correspond à l'inverse du sens trigonométrique (le sens des aiguilles d'une montre). Pour un autre type de forme, l'effet est équivalent.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

165/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

a.e) Apparence du trait de contour

Le trait qui trace le contour de la forme a de multiples propriétés. Voici les principales.

Type de trait

```
MaForme.LineStyle = com.sun.star.drawing.LineStyle.DASH
```

Les valeurs sont des [horribles constantes](#) :

Constante	Signification
NONE	le trait est invisible
SOLID	trait continu (valeur par défaut)
DASH	trait tireté; la forme du tireté dépend de la propriété <code>LineDash</code>

La propriété `LineDash` est un élément composite :

Sous-Propriété	Signification
<code>Style</code>	la forme du tireté (constante <code>com.sun.star.drawing.DashStyle.xxx</code>)
<code>Dots</code>	nombre entier; nombre de points dans le tireté
<code>DotLen</code>	longueur du point en 1/100 de mm
<code>Dashes</code>	nombre entier; nombre de tirets dans le tireté
<code>DashLen</code>	longueur d'un tiret en 1/100 de mm
<code>Distance</code>	distance entre les points en 1/100 de mm

Les différentes formes de tireté :

Constante	Signification
RECT	un rectangle
ROUND	un disque
RECTRELATIVE	un rectangle proportionnel à la longueur de la ligne
ROUNDRELATIVE	un disque proportionnel à la longueur de la ligne

Un exemple clarifiera peut-être un peu :

```
Dim LesTirets As New com.sun.star.drawing.LineDash
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

166/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
With LesTirets
  .Style = com.sun.star.drawing.DashStyle.RECT
  .Dots = 3
  .DotLen = 50
  .Dashes = 2
  .DashLen = 200
  .Distance = 100
end with

With MaForme
  rem ( autres initialisations de la forme )
  .LineStyle = com.sun.star.drawing.LineStyle.DASH
  .LineDash = LesTirets
end with
```

Couleur du trait

```
MaForme.LineColor = RGB(250, 0, 0)
```

Epaisseur du trait

```
MaForme.LineWidth = 100
```

L'épaisseur est exprimée en 1/100 de mm.

a.f) Exemple récapitulatif

Cet exemple reprend la plupart des notions de dessin. La forme obtenue est visible en dessous, utiliser un zoom de 200% pour bien voir les détails.

```
Option Explicit

Sub ExemplePourDocIntroAPI
Dim MonDocument As Object, MonTexte As Object
Dim MonCurseur As Object
Dim MaForme As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point
Dim LesTirets As New com.sun.star.drawing.LineDash

MonDocument = ThisComponent
MonTexte = MonDocument.Text
MonCurseur = MonTexte.createTextCursor

MonCurseur = MonDocument.Bookmarks.getByName("ResultatExempleDessin").getAnchor
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

167/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
Taille1.Width = 7400
Taille1.Height = 2530
Position1.x = 4500
Position1.y = 12300
With LesTirets
  .Style = com.sun.star.drawing.DashStyle.RECT
  .Dots = 3
  .DotLen = 50
  .Dashes = 2
  .DashLen = 200
  .Distance = 150
end with

MaForme = MonDocument.createInstance("com.sun.star.drawing.RectangleShape")
MaForme.Size = Taille1

MonTexte.insertTextContent( MonCurseur, MaForme, false)
With MaForme
  .String = "OpenOffice"
  .Shadow = true
  .ShadowColor = RGB(100, 100, 100) ' gris sombre
  .ShadowXDistance = 800
  .ShadowYDistance = 250
  .RotateAngle = 1500 ' 15 degrés
  .ShearAngle = 3000 ' 30 degrés
  .FillColor = RGB(200, 150, 200)
  .LineStyle = com.sun.star.drawing.LineStyle.DASH
  .LineDash = LesTirets
  .LineColor = RGB(250, 0, 0) ' rouge
  .LineWidth = 80
  ' .FillStyle = com.sun.star.drawing.FillStyle.NONE
end with

End Sub
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

168/186

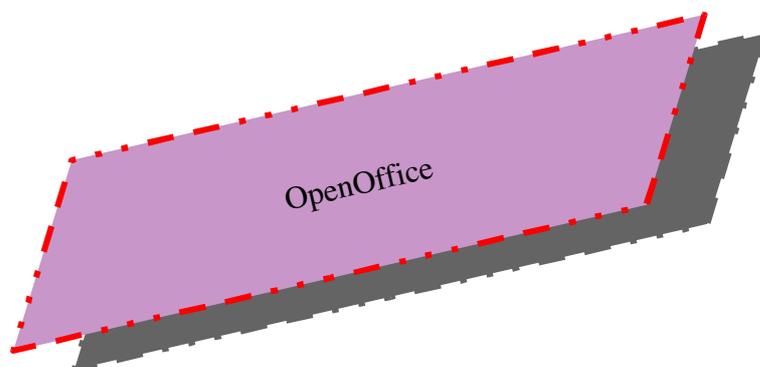
Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Résultat



b) Dessiner des formes

b.a) Les différentes formes de base

Les principales formes élémentaires disponibles dans `com.sun.star.Drawing` sont :

Forme	Signification
<code>CaptionShape</code>	Etiquette
<code>ClosedBezierShape</code>	Courbe de Bézier fermée (1)
<code>ConnectorShape</code>	Connecteur
<code>EllipseShape</code>	Ellipse ou cercle
<code>GraphicObjectShape</code>	Objet graphique
<code>LineShape</code>	Ligne
<code>MeasureShape</code>	Ligne de cote
<code>OpenBezierShape</code>	Courbe de Bézier ouverte (1)
<code>PolyLineShape</code>	Ligne brisée
<code>PolyPolygonShape</code>	Polygone
<code>RectangleShape</code>	Rectangle ou carré
<code>TextShape</code>	Texte

(1) Je ne suis pas arrivé à dessiner ces formes, ni en OOo 1.0.2 ni en 1.1RC.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

169/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

b.b) Le rectangle

Déjà vu dans les exemples de base, il se définit par sa taille (hauteur, largeur) et sa position sur la page. Si la hauteur est égale à la largeur, évidemment vous obtenez un carré.

Une propriété spécifique est `CornerRadius` qui sert à arrondir les angles. Sa valeur est le rayon du quart de cercle employé pour les quatre coins.

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
Taille1.Width = 13400 ' 134 mm de large
Taille1.Height = 2530 ' 25,3 mm de haut
Position1.x = 2500 ' 25 mm à droite du coin haut-gauche de la page
Position1.y = 5300 ' 53 mm plus bas que le coin haut-gauche de la page
MaForme = MonDocument.CreateInstance("com.sun.star.drawing.RectangleShape")
With MaForme
    .Size = Taille1
    .Position = Position1
    .CornerRadius = 250 ' 2,5 mm
    .FillColor = RGB(137, 111, 87)
End With
UnePage.add(MaForme)
```

b.c) L'ellipse

Déjà vu dans les exemples de base, elle se définit par sa taille (hauteur, largeur) et sa position sur la page. Si la hauteur est égale à la largeur, évidemment vous obtenez un cercle.

Plusieurs propriétés permettent de dessiner des sections d'ellipse.

Propriété	Signification
<code>CircleKind</code>	Type d'ellipse
<code>CircleStartAngle</code>	Angle initial de l'arc de la section
<code>CircleEndAngle</code>	Angle final de l'arc de la section

Les angles sont mesurés en 1/100 de degré, valeurs positives dans le sens

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

170/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

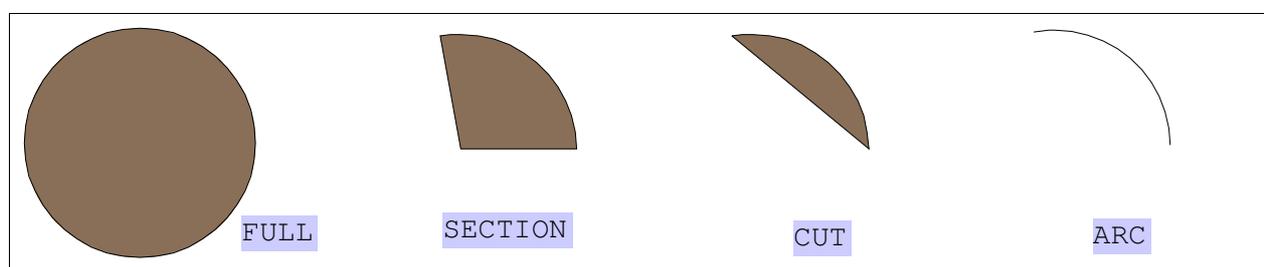
Révision : 1.1

trigonométrique (le sens inverse des aiguilles d'une montre). La position zéro de l'angle est l'horizontale, avec le centre de rotation situé à gauche de l'arc. Dans les dessins ci-dessous, l'angle de départ est 0 degré, l'angle final est 100 degrés.

Le type d'ellipse est une horrible constante, dont la valeur par défaut est :

```
com.sun.star.drawing.CircleKind.FULL
```

Avec la valeur `FULL` les propriétés de valeur d'angle ne sont pas significatives. Voici les autres constantes possibles :



Exemple de dessin d'une section de disque :

```
MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
Taille1.Width = 3000 ' 30 mm de large
Taille1.Height = 3000 ' 30 mm de haut
Position1.x = 2500 ' 25 mm à droite du coin haut-gauche de la page
Position1.y = 5300 ' 53 mm plus bas que le coin haut-gauche de la page
MaForme = MonDocument.createInstance("com.sun.star.drawing.EllipseShape")
With MaForme
  .Size = Taille1
  .Position = Position1
  .CircleKind = com.sun.star.drawing.CircleKind.SECTION
  .CircleStartAngle = 0
  .CircleEndAngle = 10000 ' angle en 1/100 de degré
  .FillColor = RGB(137, 111, 87)
End With
UnePage.add(MaForme)
```

b.d) La ligne simple

Une ligne simple se définit par les coordonnées de ses extrémités. La macro suivante dessine cette ligne :

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1



```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim CoordPt(1) As New com.sun.star.awt.Point

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = MonDocument.createInstance("com.sun.star.drawing.LineShape")

CoordPt(0).X = 11000 ' position par rapport à la page
CoordPt(0).Y = 8000
CoordPt(1).X = 16000
CoordPt(1).Y = 6000

UnePage.add(MaForme)
MaForme.PolyPolygon = Array(CoordPt())
MaForme.LineColor = RGB(200, 220, 20)
MaForme.LineWidth = 100 ' 1 mm
```

b.e) La ligne brisée

On décrit la ligne brisée avec les coordonnées de ses points successifs. La description des points doit être remplie après insertion de la forme dans la page. La macro suivante dessine cette ligne brisée :



```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim CoordPt(7) As New com.sun.star.awt.Point
Const Y1 = 6000, Y2 = 8000, Ecart = 1000 ' unité 1/100 de mm

MonDocument = ThisComponent
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

172/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = MonDocument.createInstance("com.sun.star.drawing.PolyLineShape")

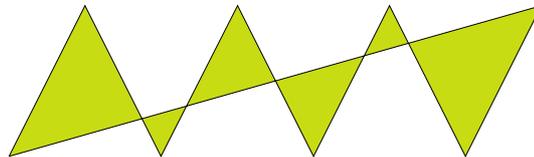
CoordPt(0).X = 11000 ' position par rapport à la page
CoordPt(0).Y = Y2
CoordPt(1).X = CoordPt(0).X + Ecart
CoordPt(1).Y = Y1
CoordPt(2).X = CoordPt(1).X + Ecart
CoordPt(2).Y = Y2
CoordPt(3).X = CoordPt(2).X + Ecart
CoordPt(3).Y = Y1
CoordPt(4).X = CoordPt(3).X + Ecart
CoordPt(4).Y = Y2
CoordPt(5).X = CoordPt(4).X + Ecart
CoordPt(5).Y = Y1
CoordPt(6).X = CoordPt(5).X + Ecart
CoordPt(6).Y = Y2
CoordPt(7).X = CoordPt(6).X + Ecart
CoordPt(7).Y = Y1

UnePage.add(MaForme)
MaForme.PolyPolygon = Array(CoordPt())
MaForme.LineColor = RGB(200, 220, 20)
MaForme.LineWidth = 100 ' 1 mm
```

En fait, `LineShape` et `PolyLineShape` sont interchangeables dans ces deux dernières macros.

b.f) Le Polygone

On décrit le polygone avec les coordonnées de ses points successifs, sachant que le dernier point sera relié automatiquement au premier. La description des points doit être remplie après insertion de la forme dans la page. La macro suivante dessine ce polygone :



```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim CoordPt(7) As New com.sun.star.awt.Point
Const Y1 = 6000, Y2 = 8000, Ecart = 1000 ' unité 1/100 de mm
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

173/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = MonDocument.createInstance("com.sun.star.drawing.PolyPolygonShape")

CoordPt(0).X = 11000 ' position par rapport à la page
CoordPt(0).Y = Y2
CoordPt(1).X = CoordPt(0).X + Ecart
CoordPt(1).Y = Y1
CoordPt(2).X = CoordPt(1).X + Ecart
CoordPt(2).Y = Y2
CoordPt(3).X = CoordPt(2).X + Ecart
CoordPt(3).Y = Y1
CoordPt(4).X = CoordPt(3).X + Ecart
CoordPt(4).Y = Y2
CoordPt(5).X = CoordPt(4).X + Ecart
CoordPt(5).Y = Y1
CoordPt(6).X = CoordPt(5).X + Ecart
CoordPt(6).Y = Y2
CoordPt(7).X = CoordPt(6).X + Ecart
CoordPt(7).Y = Y1

UnePage.add(MaForme)
MaForme.PolyPolygon = Array(CoordPt())
MaForme.FillColor = RGB(200, 220, 20)
```

b.g) Combiner plusieurs formes

En réalité, on peut insérer en une seule fois plusieurs formes élémentaires du même type. C'est la raison pour laquelle l'instruction `Array(...)` a été employée. Le résultat est une seule forme combinée. Comme exemple, reprenons le polygone, et insérons-le deux fois en le décalant horizontalement :

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
Dim CoordPt(7) As New com.sun.star.awt.Point
Dim CoordPt2(7) As New com.sun.star.awt.Point
Const Y1 = 6000, Y2 = 8000, Ecart = 1000 ' unité 1/100 de mm

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = MonDocument.createInstance("com.sun.star.drawing.PolyPolygonShape")

CoordPt(0).X = 11000 ' position par rapport à la page
CoordPt(0).Y = Y2
CoordPt(1).X = CoordPt(0).X + Ecart
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

174/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

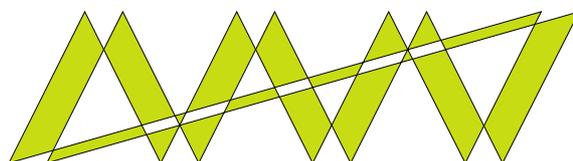
```
CoordPt (1) .Y = Y1
CoordPt (2) .X = CoordPt (1) .X + Ecart
CoordPt (2) .Y = Y2
CoordPt (3) .X = CoordPt (2) .X + Ecart
CoordPt (3) .Y = Y1
CoordPt (4) .X = CoordPt (3) .X + Ecart
CoordPt (4) .Y = Y2
CoordPt (5) .X = CoordPt (4) .X + Ecart
CoordPt (5) .Y = Y1
CoordPt (6) .X = CoordPt (5) .X + Ecart
CoordPt (6) .Y = Y2
CoordPt (7) .X = CoordPt (6) .X + Ecart
CoordPt (7) .Y = Y1

CoordPt2 (0) .X = 11500 ' position par rapport à la page
CoordPt2 (0) .Y = Y2
CoordPt2 (1) .X = CoordPt2 (0) .X + Ecart
CoordPt2 (1) .Y = Y1
CoordPt2 (2) .X = CoordPt2 (1) .X + Ecart
CoordPt2 (2) .Y = Y2
CoordPt2 (3) .X = CoordPt2 (2) .X + Ecart
CoordPt2 (3) .Y = Y1
CoordPt2 (4) .X = CoordPt2 (3) .X + Ecart
CoordPt2 (4) .Y = Y2
CoordPt2 (5) .X = CoordPt2 (4) .X + Ecart
CoordPt2 (5) .Y = Y1
CoordPt2 (6) .X = CoordPt2 (5) .X + Ecart
CoordPt2 (6) .Y = Y2
CoordPt2 (7) .X = CoordPt2 (6) .X + Ecart
CoordPt2 (7) .Y = Y1

UnePage.add (MaForme)

MaForme.PolyPolygon = Array (CoordPt (), CoordPt2 ())
MaForme.FillColor = RGB (200, 220, 20)
```

L'image résultante est celle-ci :



Notez l'effet de combinaison, qui "enlève" les surfaces communes aux deux formes élémentaires.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

175/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Avec de l'habitude on peut réaliser par macro des dessins précis qui seraient plus difficile de dessiner manuellement avec l'interface graphique. Les créateurs graphiques peuvent y trouver une source d'inspiration illimitée.

b.h) Disposition relative des formes

Une forme peut en cacher une autre. Tout dépend de l'ordre dans lequel elles sont affichées. Cet ordre est défini par la propriété `ZOrder`, une forme donnée peut cacher toutes celles qui possèdent un `ZOrder` inférieur.

En lisant puis modifiant le `ZOrder` de vos formes vous décidez de l'ordre effectif d'affichage, indépendamment de l'ordre dans lequel vous les avez dessinées.

b.i) Grouper des formes

La macro ci-dessous regroupe trois formes.

```
Dim MonDocument As Object
Dim MaForme1 As Object, MaForme2 As Object, MaForme3 As Object
Dim UnePage As Object
Dim Taille1 As New com.sun.star.awt.Size
Dim Position1 As New com.sun.star.awt.Point
Dim Collect1 As Object, MonGroupe As Object

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")

Taille1.Width = 4000
Taille1.Height = 4000
Position1.x = 3000
Position1.y = 5300
MaForme1 = MonDocument.CreateInstance("com.sun.star.drawing.EllipseShape")
MaForme1.Size = Taille1
MaForme1.Position = Position1
UnePage.add(MaForme1)

Position1.x = 9000
Position1.y = 5300
MaForme2 = MonDocument.CreateInstance("com.sun.star.drawing.EllipseShape")
MaForme2.Size = Taille1
MaForme2.Position = Position1
UnePage.add(MaForme2)

Taille1.Width = 5600
Taille1.Height = 8000
Position1.x = 5200
Position1.y = 6500
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

176/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MaForme3 = MonDocument.CreateInstance("com.sun.star.drawing.EllipseShape")
MaForme3.Size = Taille1
MaForme3.Position = Position1
UnePage.add(MaForme3)
```

```
Collect1 = createUnoService("com.sun.star.drawing.ShapeCollection")
Collect1.Add(MaForme1)
Collect1.Add(MaForme2)
Collect1.Add(MaForme3)
MonGroupe = UnePage.group(Collect1)
```

Il est ensuite possible de déplacer ou redimensionner la forme composite résultante, mais il n'est pas possible de modifier par macro les autres caractéristiques : trait, couleur, texte. Effectuez donc ces formatages avant de grouper les formes.

c)Ecrire du texte dans la forme

Une fois la forme insérée avec l'instruction `Add`, un simple texte s'écrit ainsi :

```
MaForme.String = "OpenOffice"
```

Cette manière de faire ne nous permet pas de choisir le format du texte. Pour obtenir des possibilités plus étendues, on utilise les mêmes fonctionnalités de texte que Writer. Les instructions suivantes écrivent deux lignes dans la forme, en choisissant la taille des caractères la graisse et en changeant la police courante; elles seront claires après avoir lu les premiers chapitres sur Writer.

```
Dim MonTextel As Object, MonCurseur1 As Object
Dim FinParagraphe As Integer

FinParagraphe = com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK

rem - initialisation et insertion de la forme, voir plus haut -

MonTextel = MaForme.Text
MonCurseur1 = MaForme.createTextCursor
MonCurseur1.CharHeight = 16
MonTextel = MaForme.Text
MonTextel.InsertString(MonCurseur1, "OpenOffice", false)
MonTextel.InsertControlCharacter(MonCurseur1, FinParagraphe, false)
MonCurseur1.CharFontName = "Arial"
MonCurseur1.CharHeight = 24
MonCurseur1.CharWeight = com.sun.star.awt.FontWeight.BOLD
MonTextel.InsertString(MonCurseur1, "En gras, police Arial", false)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

177/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

d) Trouver une forme

d.a) Trouver les formes sélectionnées par l'utilisateur

L'utilisateur peut sélectionner une ou plusieurs formes. La propriété `CurrentSelection` fournit une liste des formes et images sélectionnées dans la page. Mais attention, si rien n'est sélectionné, cette propriété ne peut être utilisée; il faut tester son existence avant !

Cet exemple balaie les formes sélectionnées et impose une couleur de remplissage. Cela ne marchera évidemment que pour des formes qui le permettent.

```
Dim MaForme As Object, LesFormes As Object
Dim UnePage As Object
Dim s1 As Long

MonDocument = ThisComponent
UnePage = MonDocument.CurrentController.CurrentPage
if isObject(MonDocument.CurrentController.Selection) then
  LesFormes = MonDocument.CurrentSelection
  for s1 = 0 to LesFormes.Count - 1
    MaForme = LesFormes(s1)
    MaForme.FillColor = RGB(100,200, 100)
  next
end if
```

d.b) Nommer une forme

Pour pouvoir retrouver une forme, un bon moyen est de la nommer. Une fois la forme insérée dans le document, il suffit d'utiliser la propriété `Name` :

```
MaForme.Name = "Premier Objet"
```

Attention : vous pouvez ainsi donner le même nom à plusieurs objets de la même page !

En version 1.0.x, l'API a un avantage sur l'interface usager, qui ne permet pas de renommer une forme simple. En version 1.1, cette limite est levée.

Après exécution le Navigateur vous affichera les objets nommés dans la page; éventuellement faire un double clic sur l'icône de la page, dans le Navigateur.

d.c) Trouver une forme par son nom

Toutes les formes (et les images) d'une page sont accessibles par un index, exemple :

```
UneForme = UnePage(3)
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

178/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

Ceci est peu pratique, mais il n'existe pas de méthode API pour retrouver une forme par son nom. Pas de problème, nous allons écrire une fonction très générale, inspirée par le remarquable travail de Danny Brewer, voir son site web (en anglais) :

<http://kosh.datateamsys.com/~danny/OOo/DannysDrawPowerTools>

```
Function FindObjectByName(LaPage As Object, NomForme As String ) As Object
Dim formeX As Object
Dim fl As Long
  For fl = 0 To LaPage.Count - 1
    formeX = LaPage(fl)
    If formeX.Name = NomForme Then
      FindObjectByName = formeX
      Exit Function
    EndIf
  Next
rem ici la fonction retourne null
End Function
```

Maintenant nous pouvons retrouver la forme et la modifier :

```
rem - exemple pour un document Draw -
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object

MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = FindObjectByName(UnPage, "Premier Objet")
if IsNull(MaForme) then
  print "Nom inexistant !"
else
  MaForme.FillColor = RGB(207, 150, 180)
end if
```

Attention : cette méthode retrouve un objet quelconque sur la page, aussi évitez de donner le même nom à une forme et à une image de la même page.

d.d) Sélectionner visiblement une forme

Nous allons sélectionner une forme que nous avons trouvée par son nom. Cette macro est écrite pour Draw; on suppose que cette forme est dans une autre page que celle qui est affichée.

```
Dim MonDocument As Object
Dim MaForme As Object
Dim UnePage As Object
```

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

179/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
MonDocument = ThisComponent
UnePage = MonDocument.DrawPages.getByName("testPage")
MaForme = FindObjectByName(UnPage, "Premier Objet")
if IsNull(MaForme) then
    print "Nom inexistant !"
else
    MonDocument.CurrentController.CurrentPage = UnePage
    MonDocument.CurrentController.Select(MaForme)
end if
```

Il est nécessaire, comme nous l'avons fait, d'afficher la page de dessin où se trouve la forme, sinon la sélection ne se fera pas.

VII.1.11 Informations annexes

a) Basic simplifie l'utilisation de l'API

Si vous lisez dans des forums ou d'autres documents des macros Basic utilisant l'API, vous verrez des constructions un peu différentes de celles décrites ici. La raison est qu'il y a plusieurs manières de faire la même chose en Basic, et certaines macros sont inspirées du langage Java, qui n'accepte que certaines constructions plus lourdes.

a.a) Les parenthèses optionnelles

Quand une méthode ne comporte aucun argument, les parenthèses ne sont pas nécessaires.

```
MonDocument.Store()
```

Simplification Basic

```
MonDocument.Store
```

a.b) Les getXxxx et setXxxx

L'API utilise abondamment des méthodes en `getXxxx` pour obtenir une information, et en `setXxxx` pour modifier l'information.

Basic fournit des **pseudo**-propriétés sous forme d'un `Xxxx` accessible en lecture et/ou écriture. Écriture utilisant strictement l'API :

```
tx3 = UneCellule.getString()
UneCellule.setString("Bonsoir")
```

Simplification Basic :

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

180/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

```
tx3 = UneCellule.String  
UneCellule.String = "Bonsoir"
```

`String` n'est pas une véritable propriété de cellule, et n'est pas décrit dans la documentation API. C'est une facilité offerte par le Basic OpenOffice.

a.c) L'accès aux propriétés

La plupart des propriétés sont accessibles en écriture et en lecture. Cela veut dire qu'on peut en général modifier ou récupérer la valeur actuelle d'une propriété.
Exemple :

```
couleur = UneCellule.CellBackColor ' récupérer la valeur actuelle  
UneCellule.CellBackColor = RGB(255,255,204) ' affecter une valeur
```

Vous pourrez rencontrer dans la littérature une forme plus complexe mais équivalente pour manipuler une propriété :

```
couleur = UneCellule.getPropertyValue("CellBackColor")  
UneCellule.setPropertyValue("CellBackColor", RGB(255,255,204))
```

Dans cette tournure le nom de la propriété doit respecter la casse (majuscules, minuscules).

b) Qu'est-ce qu'un mot ?

Un mot est une séquence de caractères entre deux séparateurs de mots ! Bien, mais qu'est-ce qu'un séparateur de mot ? C'est soit un espace, un caractère de fin de paragraphe ou de fin de ligne, ou un caractère parmi ceux de la chaîne de caractères `WordSeparator` qui est une propriété du document.

Vous pouvez voir et modifier cette chaîne sur un document Writer ouvert, en ouvrant le menu Fichier / Propriétés, onglet Statistiques. Déclenchez l'aide en ligne pour avoir quelques explications de plus. Le nombre de mots d'un document est dans sa propriété `WordCount`.

La notion de mot est un peu différente pour la gestion de curseur. Vous vous en apercevrez si vous déplacez le curseur visible en tapant Control+ touche de déplacement à droite. Le curseur bute sur des caractères comme " + / , ". Ce sera aussi le cas avec `gotoNextWord`.

La propriété de document `WordCount` effectue un calcul de mots sophistiqué qui n'est malheureusement pas accessible comme routine indépendante. Si vous voulez compter le nombre de mots d'une sélection ne cherchez pas à faire une macro, elle donnera des résultats différents. Contentez-vous d'ouvrir un document Writer vide, d'y coller la sélection, et de lire le nombre de mots dans l'onglet Statistiques du menu Propriétés.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

181/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

c) Les horribles constantes

L'API fait un large usage de constantes ou des valeurs d'énumération; ce sont en fait des valeurs numériques simples comme 1, 2, 3, 4. Vous ne devez pas utiliser ces nombres, car ils peuvent évoluer avec l'implémentation. Par contre les noms de ces constantes sont plus stables.

Mais le même concept peut donner lieu à des constantes dans différents modules. Par exemple vous pouvez rechercher dans l'index de l'API les définitions de `LEFT` : environ 26 définitions indépendantes, avec des valeurs réelles souvent très différentes.

Pour distinguer les constantes relatives à deux modules différents, il est nécessaire de qualifier le nom d'une constante avec le nom complet du module, qui est défini hiérarchiquement.

Cette hiérarchie est aussi celle de l'arborescence des [pages HTML de l'API](#). Lorsque vous arrivez sur une page décrivant les valeurs d'une énumération vous aurez quelque chose comme :

Adresse de la page HTML : `blabla/com/sun/star/table/CellHoriJustify.html`

Au début de cette page HTML vous verrez :

```
:: com :: sun :: star :: table ::
```

```
enum CellHoriJustify
```

Et différentes valeurs comme `STANDARD LEFT CENTER` etc.

La règle de transcription vers une constante Basic est facilement déduite de cet exemple :

```
Dim maJustif As Integer
rem Utilisation d'une variable intermédiaire pour faciliter la lecture
maJustif = com.sun.star.table.CellHoriJustify.STANDARD
UneCellule.HoriJustify = maJustif

rem Utilisation directe de la constante
UneCellule.HoriJustify = com.sun.star.table.CellHoriJustify.STANDARD
```

Attention

Le nom complet de la constante doit être exactement recopié, majuscules et minuscules.

d) Les Couleurs

Dans OpenOffice une couleur est définie par trois valeurs entières , chacune entre

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

182/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

0 et 255, qui représentent l'intensité des couleurs primaires Rouge, Vert, Bleu (en anglais : Red, Green, Blue).

Chaque couleur est représentable dans un octet, les trois octets étant combinés sous la forme d'un entier long (type de donnée : Long).

Connaissant les valeurs de chaque couleur primaire on peut déterminer la valeur OpenOffice de la couleur résultante :

```
Dim couleur As Long
Dim partRouge As Integer, partVerte As Integer, partBleue As Integer

partRouge = 120
partBleue = 60
partVerte = 220
couleur = RGB(partRouge, partVerte, partBleue)
```

Connaissant la valeur de couleur on peut retrouver la valeur de chaque couleur primaire :

```
Dim couleur As Long
Dim partRouge As Integer, partVerte As Integer, partBleue As Integer

couleur = 8253534
partBleue = Blue(couleur)
partVerte = Green(couleur)
partRouge = Red(couleur)
```

Astuce

L'outil de configuration de palette de couleurs, accessible via le menu Outils / Options / OpenOffice.org / Couleurs vous permet de trouver le triplet correspondant à la couleur de votre choix. Cliquez sur le bouton Editer, cliquez sur la couleur de votre choix dans l'arc-en-ciel, recopiez sur un bout de papier les valeurs Rouge-Vert-Bleu qui apparaissent, et fermez la fenêtre par Annuler.

e) Consulter l'API

Les procédures et fonctions décrites ici ne sont qu'un infime sous-ensemble de celles disponibles. Pour en savoir plus, il faut consulter la documentation de l'API . Mais elle est immense et déroutante, difficile à exploiter, et **en anglais seulement**. Voici quelques maigres aides à la lecture.

Le nouveau document de Sun : [StarOffice 6.0 Software Basic Programmer's Guide](#) seulement disponible en anglais, contient des conseils nombreux pour la migration Visual Basic vers OpenOffice Basic, une bonne introduction à l'API, et traite de certains sujets API non abordés ici.

L'outil Xray, disponible sur le site

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

183/186

Programmation Macro OpenOffice

Statut : En cours

Formation OpenOffice

Révision : 1.1

<http://fr.openoffice.org/Documentation/Index.html> , vous permettra de connaître les possibilités des objets API manipulés dans vos macros. Dans sa version 2, il facilite les consultations de l'API.

Le SDK est disponible sur le site <http://api.openoffice.org/> et plus précisément à la page http://www.openoffice.org/dev_docs/source/sdk/ .

La version OpenOffice.org SDK 1.1 disponible à la date de ce document pèse 40,5 Moctets pour Windows, 27,9 Moctets pour Linux et 29,2 Moctets pour Solaris.

Le SDK se décompose en une immense arborescence qui comprend l'API, le Developer's Guide, et de nombreux autres documents.

Le Developer'Guide est disponible dans le SDK en format PDF (un pavé de 900 pages indigestes) et sous forme HTML très pratique à consulter, avec des liens vers la documentation API, elle-même en HTML. Vous trouverez dans le Guide des informations sur la "philosophie" de l'API et deux chapitres sur son implémentation avec Basic. En pratique le Guide est seulement accessible à des spécialistes informaticiens; les exemples sont en général fournis en Java.

L'API seule est seulement disponible sous forme de plus de 4000 pages HTML, [consultables en ligne](#) . Les liens sont nombreux entre les différents éléments, et un bon navigateur est nécessaire (par exemple [Opera](#), qui facilite l'ouverture de fenêtres simultanées). Les pages disponibles en ligne ont une présentation un peu différente de celles du fichier compressé.

Utilisez les pages d'index pour sauter assez rapidement à ce qui vous intéresse, et pour repérer des éléments nouveaux.

On retrouvera les mêmes noms de procédures ou fonctions dans différents contextes, avec des fonctionnalités variant légèrement.

Le tableau de la page suivant fournit quelques repères.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

184/186

Programmation Macro OpenOffice

Statut : En cours

Formation OpenOffice

Révision : 1.1

Sujet	Où trouver l'information
Document	com > sun > star > Document
Texte	com > sun > star > text
Tableur	com > sun > star > sheet
Diagramme de données	com > sun > star > chart
Dessin	com > sun > star > drawing
Présentation (Impress)	com > sun > star > presentation
Styles de paragraphe et de cellule	com > sun > star > style
Tableaux, en général (Calc, Writer)	com > sun > star > table
Propriété de ligne de tableau (Writer)	index > TextTableRow
Contenu de Cellule	index > XCell
Formatage de cellule	index > CellProperties (table pour Calc) index > CellProperties (text pour Writer)
Méthodes pour zone de cellules	index > XCellRange
Zone de texte	index > XTextRange TextRangeCompare XTextRangeMover
Curseur pour aller de cellule en cellule	index > XCellCursor (pour Calc) index > XTextTableCursor (pour Writer)
Propriétés d'un cadre de texte	index > TextFrame > BaseFrame
Propriétés de paragraphe et de caractère dans un texte	com > sun > star > text > Paragraph
Propriétés de paragraphe de texte	index > ParagraphProperties
Forme (tableau, cadre, dessins)	index > Shape
Propriétés de trait (dessin)	index > LineProperties
Remplissage d'une forme (couleur de fond)	index > FillProperties

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

185/186

Programmation Macro OpenOffice

Statut : En cours

**Formation
OpenOffice**

Révision : 1.1

VIII Licence

Appendix

Public Documentation License Notice

The contents of this Documentation are subject to the Public Documentation License Version 1.0 (the "License"); you may only use this Documentation if you comply with the terms of this License. A copy of the License is available at <http://www.openoffice.org/licenses/PDL.html>.

The Original Documentation is : L'API OpenOffice.org (presque) sans peine
The Initial Writer of the Original Documentation is Bernard Marcelly
Copyright (C) 2003-2004. All Rights Reserved. (Initial Writer contact(s): marcelly@club-internet.fr)

The Original Documentation is Comment faire une macro pour choisir un modèle via une boîte de dialogue.
The Initial Writer of the Original Documentation is Vincent Morlot Copyright (C) 2003. All Rights Reserved. (Initial Writer contact(s): vincent.morlot@free.fr).

Contributor(s): _____.

Portions created by _____ are Copyright (C) _____ [Insert year(s)].

All Rights Reserved. (Contributor contact(s): _____ [Insert hyperlink/alias]).

NOTE: The text of this **Appendix** may differ slightly from the text of the notices in the files of the Original Documentation. You should use the text of this **Appendix** rather than the text found in the Original Documentation for Your Modifications.

Préparé par - Société Guidella - confidentiel

coursMacro.sxw

Imprimé le 28 décembre 2004

186/186