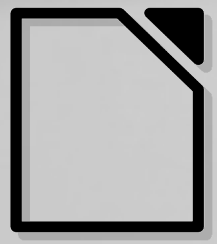
API Open Office (Tondeur Hervé)



LibreOffice
The Document Foundation

Introduction

OpenOffice dispose d'un puissant langage de macro, Basic, qui est fourni avec le produit. En plus de ses possibilités classiques de programmation, il permet de manipuler le contenu des documents Writer, Calc, etc. Ceci implique d'utiliser l'interface de programmation (API) de OpenOffice.

Télécharger LibreOffice : http://fr.libreoffice.org/telecharger/

Télécharger OpenOffice : http://fr.openoffice.org/about-downloads.html

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 (mais pas Draw, ni Impress, ni Base).

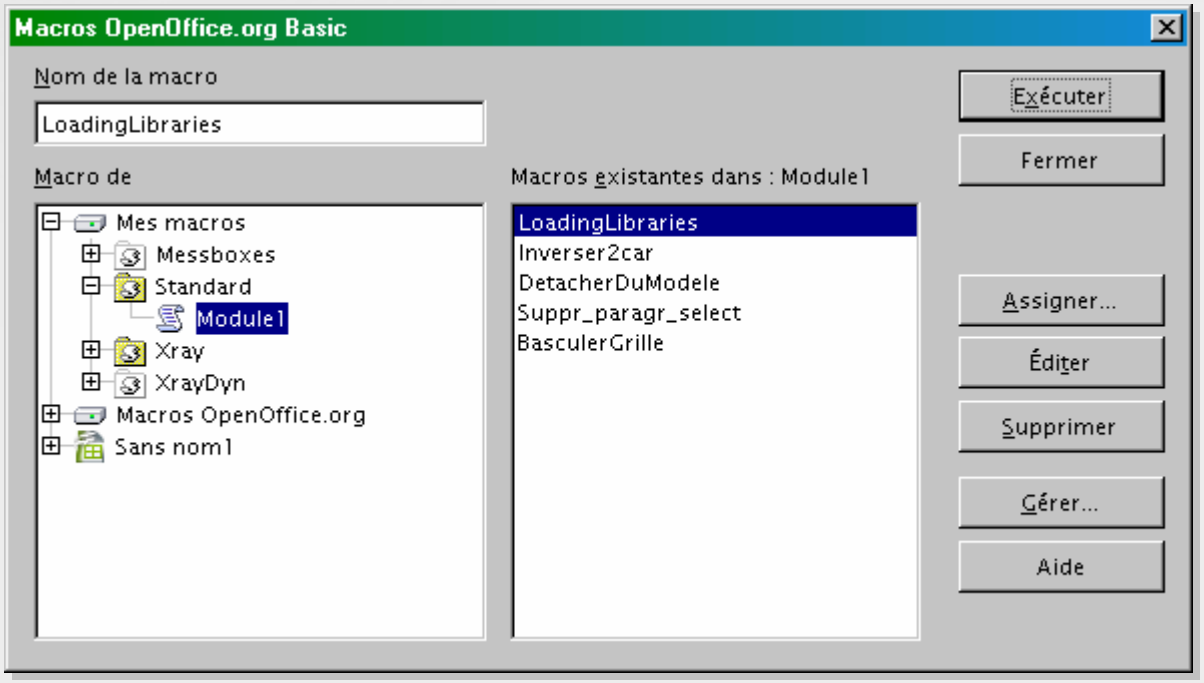
.

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. Pour tout projet autre qu'une simple séquence de touches, il est nécessaire de programmer en utilisant Basic et l'API OpenOffice.org.

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.

Accès aux macros - OpenOffice version 3.x

L'accès et le contenu sont un peu plus complexes. Par le menu **Outils > Macros > Gestionnaire de Macros > OpenOffice.org Basic**... 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.



La racine Mes Macros contient les macros que vous avez créées ou importées et qui sont disponibles pour tous les documents.

La racine Macros OpenOffice.org contient les macros livrées avec OpenOffice. En principe vous ne pouvez pas les modifier, mais vous pouvez les lire ou les utiliser.

En réalité ces deux racines ne sont là que pour vos yeux. Toutes les deux sont un seul et même conteneur, qu'on appelle **soffice** en version 1.x. Ce terme est actuellement encore utilisé dans certaines documentations de la version 2 & 3 ainsi que dans le titre par défaut des panneaux MsgBox.

Obtenir 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
```

Accéder à un autre document existant

Pour ouvrir un document existant :

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

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

MonDocument = StarDesktop.LoadComponentFromURL(AdresseDoc,"_blank",0,
PropFich)
```

PropFich est un tableau de propriétés, ici on passe un tableau vide

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)
```

ThisComponent

L'instruction Basic ThisComponent simplifie les choses. Mémorisez en début d'exécution de votre macro la valeur de ThisComponent

MonDocument = ThisComponent

Pourquoi utiliser une variable intermédiaire ? Parce que ThisComponent n'est pas un objet fixe, c'est une sorte de sous-programme qui renvoie l'objet document actuel.
Le résultat dépendra des conditions régnant au moment exact où vous appelez ThisComponent.

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.

```
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
```

Propriété	Signification
AsTemplate	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. Remarque Avec *AsTemplate = true* et une *AdresseDoc* correspondant à un document ordinaire (pas un modèle) on obtient un nouveau document basé sur celui en référence.
Hidden	booleen ; valeur true pour charger le document sans le rendre visible
Password	chaîne de caractères ; mot de passe pour ouvrir le document encrypté
ReadOnly	booléen ; valeur true pour ouvrir le document en lecture seule (pour l'usager, pas pour l'API)

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.

Créer un nouveau document

La méthode est celle de la diapo précédente, 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*

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.

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()

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() )
```

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

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

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.

Fermer le document

La méthode Close 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.Close(True)
```

Dans tous les cas pratiques, il faut mettre en argument de Close la valeur booleenne True.
Si OpenOffice n'a plus d'autre fenêtre active, et si le Démarrage rapide n'est pas utilisé, l'application OpenOffice.org se ferme.

LES MACROS SOUS CALC

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

Dim MonDocument As Object
MonDocument = ThisComponent.

Voir les diapos 6, 7, 8 & 9.

Travailler au niveau des feuilles (sheets)

Trouver 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
```

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

```
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
```

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)
```

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")
```

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*

Travailler au niveau des cellules (cells)

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.

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")
```

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.

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)
```

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
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
```

Valeur numérique dans la cellule

Lire et écrire une valeur numérique dans la cellule :

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

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.

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

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 :

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.

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 :

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))+")"

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
```

Couleur de fond de la cellule

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

Couleur de caractère

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

Taille de caractère

La taille des caractères est définie en points
UneCellule.CharHeight = 12

Justification horizontale

Encore des horribles constantes...

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

Principales valeurs possibles :

Constante	Signification
STANDARD	alignement par défaut selon le type du contenu (texte, nombre)
LEFT	Alignement à gauche
CENTER	Alignement centré
RIGHT	Alignement à droite

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
STANDARD	alignement par défaut
TOP	alignement sur le haut de la cellule
CENTER	alignement centré en hauteur
BOTTOM	alignement sur le bas de la cellule

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.

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).

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
```

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 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)

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

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)
```

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

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
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
```

Les Images dans Calc

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
```

Positionner une image

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)
```

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")
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
```

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
```

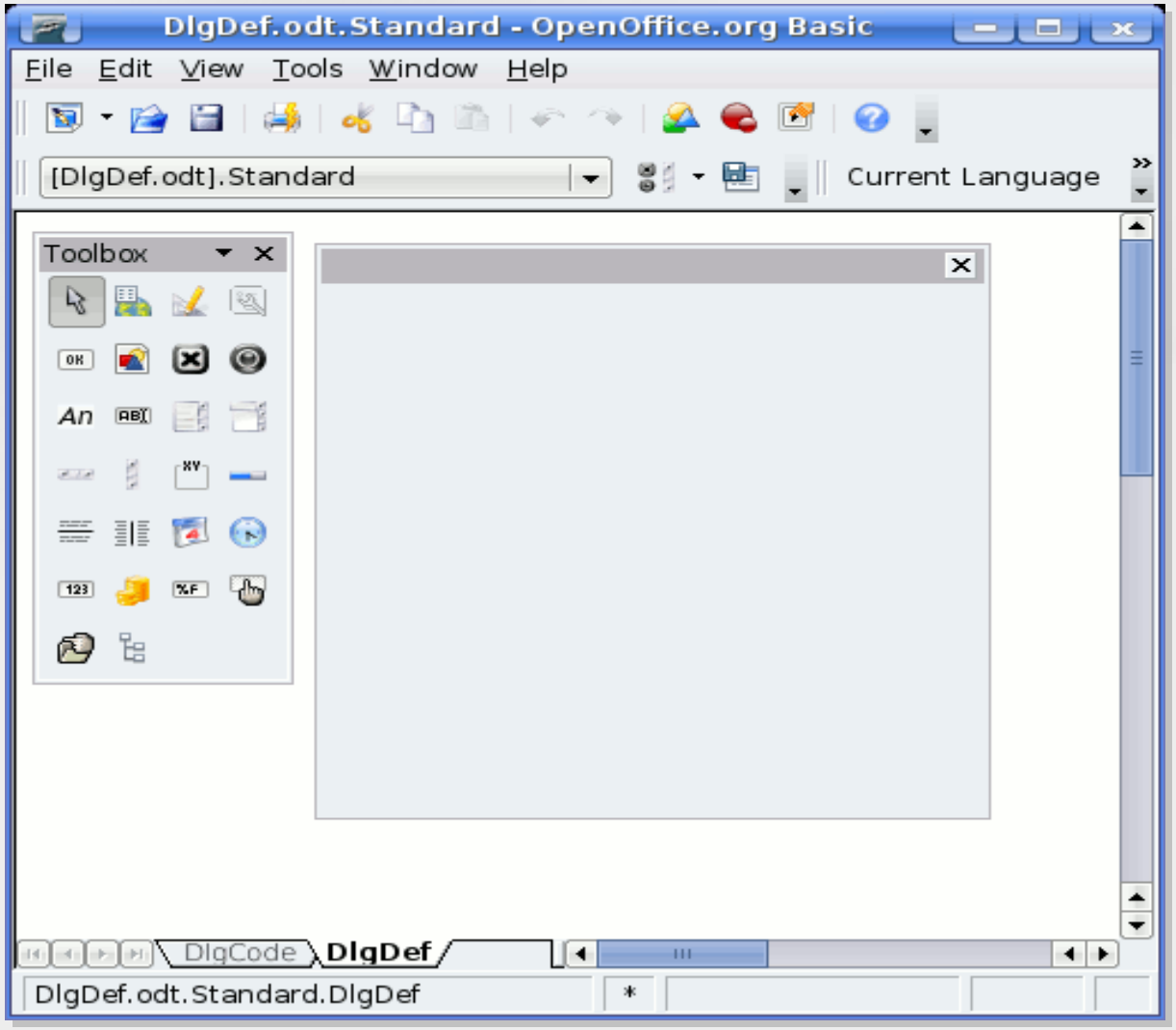
LES BOITES DE DIALOGUES

Vous pouvez ajouter des fenêtres de dialogue et des formulaires aux documents OpenOffice.org.
Celles ci à leur tour peuvent être liés à des macros Basic OpenOffice.org et étendre considérablement la gamme d'utilisation d' OpenOffice.org.

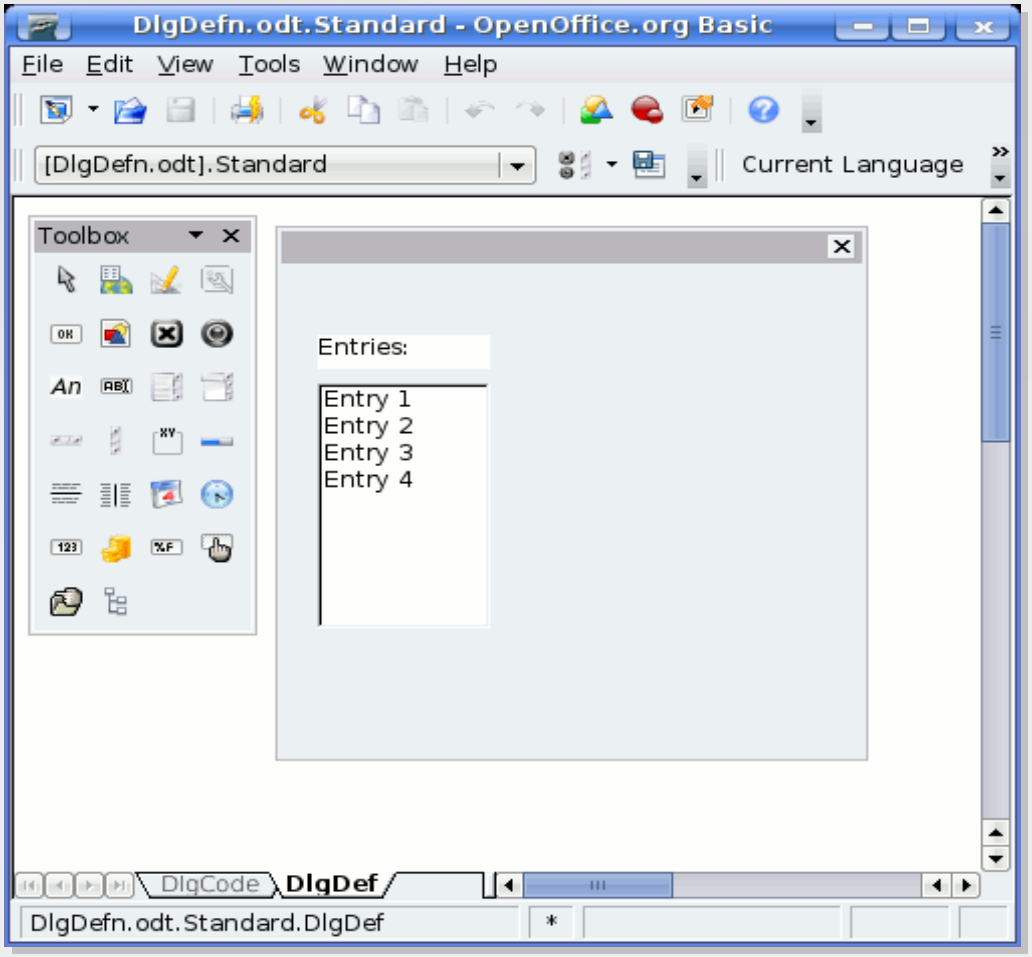
Les dialogues peuvent, par exemple, afficher des informations de bases de données ou de guider les utilisateurs à travers un processus étape par étape lors de la création d'un nouveau document sous la forme d'un assistant.

Vous pouvez créer des boites de dialogues en utilisant l'éditeur de dialogues OpenOffice.org

Vous pouvez glisser les elements de controles à partir de la palette des outils(droite) sur la feuille de dialogue, ainsi que définir leur positions et leur tailles.



Un exemple de boite de dialogue qui contient un label et une boite « liste déroulante ».



Vous pouvez ouvrir un dialog avec le code suivant :

```
Dim Dlg As Object
DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.DlgDef)
Dlg.Execute()
Dlg.dispose()
```

CreateUnoDialog crée un objet appelé **Dlg** qui référence le dialogue associé. Avant de pouvoir créer le dialogue,
vous devez vous assurer que la librairie est chargée. la méthode **LoadLibrary** permet ceci.
Une fois que le dialogue **Dlg** a été initialisé, vous pouvez exécuter la méthode pour afficher le dialogue. Ce type de dialogues
est dit modal parce qu'ils ne permettent pas aux autres programmes d'agir, jusqu'à ce qu'ils soit fermé.
Tant que ce dialogue est ouvert, le programme le rappel dans l'appel à **Execute**.
La méthode dispose à la fin du code libère les ressources utilisé par le dialogue.

Fermer avec OK ou Cancel

Si un dialogue contient un bouton "OK" ou "Cancel", le dialogues est automatiquement fermé quand vous cliquez sur l'un d'eux

Si vous fermez un dialogue en cliquant sur le bouton OK, la méthode **Execute** retourne la valeur 1, sinon la valeur 0 est retournée.

```
Dim Dlg As Object
DialogLibraries.LoadLibrary("Standard")
Dlg = CreateUnoDialog(DialogLibraries.Standard.MyDialog)
Select Case Dlg.Execute()
Case 1
MsgBox "Ok pressed"
Case 0
MsgBox "Cancel pressed"
End Select
```

Fermer avec le bouton fermer de la barre de titre

Vous pouvez fermer un dialogue en cliquant la croix rouge sur la barre de titre de la fenêtre du dialogue.
La méthode **Execute** du dialogue retourne la valeur 0.

Fermer de manière explicit par un appel programme

Vous pouvez fermer un dialogue ouvert avec la méthode **endExecute**:

Dlg.endExecute()

La méthode **Execute** du dialogue retourne la valeur 0.

Accéder aux éléments de contrôle individuellement

Un dialogue plusieurs éléments de contrôles Vous pouvez accéder a ces éléments au travers de la méthode **getControl** qui retourne le nom de l'élément de contrôle

```
Dim Ctl As Object
Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

Ce code détermine pour le contrôle **MyButton** et initialise la variable objet **Ctl** avec une référence à l'élément. Finalement le code affecte la propriété Label du contrôle avec la nouvelle valeur du label

Note – contrairement aux identifiants du basic OpenOffice.org, le nom des contrôles est sensible à la case.

LES PROPRIETES DE L'OBJET « MODEL »

Nom et Titre

Chaque élément de contrôle a son propre nom qui peut être interrogé en utilisant la propriété de modèle suivants:

Model.Name (String)
nom de l'élément de contrôle

Vous pouvez spécifier le titre qui apparaît dans la barre de titre d'un dialogue avec la propriété de modèle suivants:

Model.Title (String)
Titre de dialogue (s'applique uniquement aux dialogues)

Position et taille

Vous pouvez interroger la taille et la position d'un élément de contrôle en utilisant les propriétés suivantes de l'objet modèle:

Model.Height (long)
hauteur de l'élément de commande (en unités ma)

Model.Width (long)
largeur de l'élément de commande (en unités ma)

Model.PositionX (long)
X-position de l'élément de contrôle, mesurée à partir du bord gauche intérieur de la boîte de dialogue (en unités ma)

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

Focus et séquence de tabulation

Vous pouvez naviguer à travers les éléments de contrôle dans tout dialogue en appuyant sur la touche Tab. Les propriétés suivantes sont disponibles dans ce contexte dans le contrôle des éléments du modèle:

Model.Enabled (Boolean)
actionne l'élément de commande

Model.Tabstop (Boolean)
permet à l'élément de contrôle à atteindre par la touche Tab

Model.TabIndex (Long)
position de l'élément de contrôle dans l'ordre d'activation

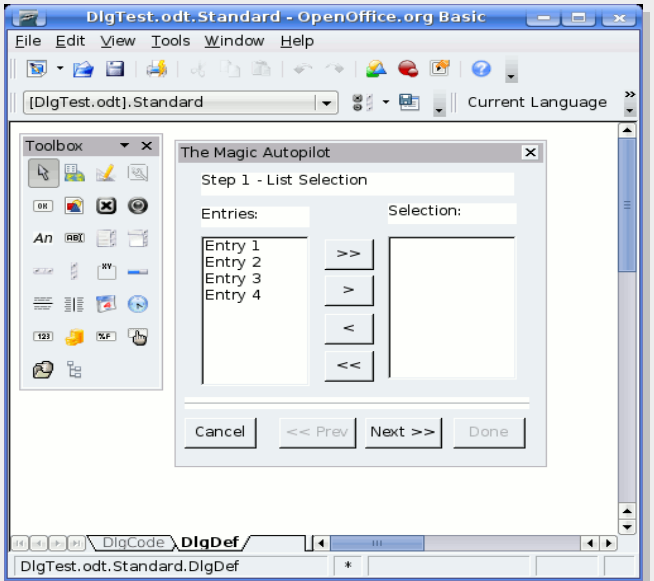
Enfin, l'élément de contrôle fournit une méthode getFocus qui garantit que l'élément de contrôle reçoit le focus sous-jacente:

getFocus
élément de contrôle reçoit le focus (uniquement pour les dialogues)

Multi-Page Dialogues

Une boîte de dialogue dans OpenOffice.org peut avoir plus d'une page d'onglets. La propriété Step d'une boîte de dialogue définit la page de l'onglet courant de la boîte de dialogue alors que la propriété Step d'un élément de contrôle spécifie la page de l'onglet où l'élément de contrôle doit être affichée.
Step-value à 0 est un cas particulier. Si vous définissez cette valeur à zéro dans une boîte de dialogue, tous les éléments de contrôle sont visibles quel que soit leur valeur Step. De même, si vous définissez cette valeur à zéro pour un élément de contrôle, l'élément est affiché sur toutes les pages d'onglets dans une fenêtre de dialogue.

Dans l'exemple précédent, vous pouvez également assigner la valeur Step de 0 à la ligne de démarcation, ainsi que la Annuler, Précédent, Suivant, et des boutons Terminé pour afficher ces éléments sur toutes les pages. Vous pouvez également affecter les éléments d'une page de l'onglet individuel.



Le code du programme suivant montre comment la valeur Step des gestionnaires d'événements des boutons Next et Prev peut être augmenté ou réduit et change l'état des boutons.

```
Sub cmdNext_Initiated
Dim cmdNext As Object
Dim cmdPrev As Object
cmdPrev = Dlg.getControl("cmdPrev")
cmdNext = Dlg.getControl("cmdNext")
cmdPrev.Model.Enabled = Not cmdPrev.Model.Enabled
cmdNext.Model.Enabled = False
Dlg.Model.Step = Dlg.Model.Step + 1
End Sub
Sub cmdPrev_Initiated
Dim cmdNext As Object
Dim cmdPrev As Object
cmdPrev = Dlg.getControl("cmdPrev")
cmdNext = Dlg.getControl("cmdNext")
cmdPrev.Model.Enabled = False
cmdNext.Model.Enabled = True
Dlg.Model.Step = Dlg.Model.Step - 1
End Sub
```

LES EVENEMENTS DES CONTROLES

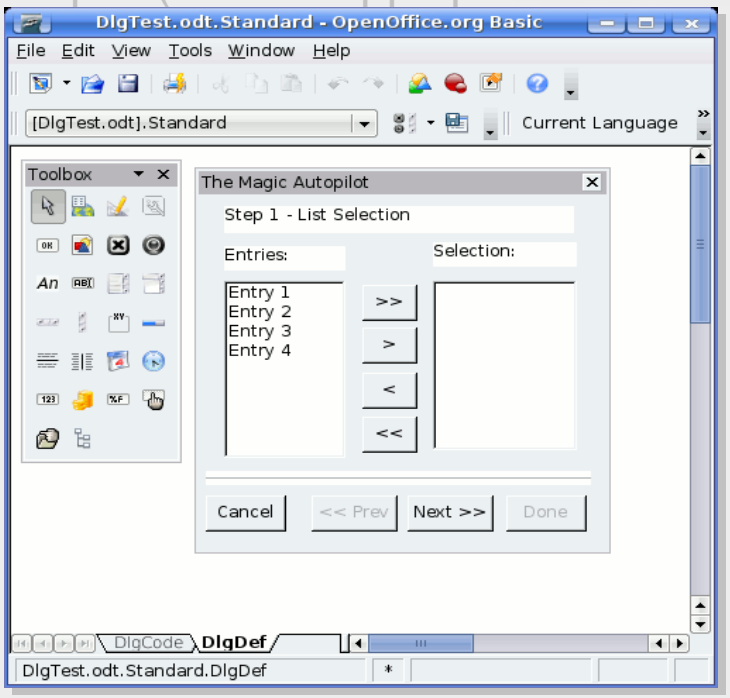
Les dialogues et les formes OpenOffice.org sont basées sur un modèle de programmation orienté événement dans lequel vous pouvez assigner des gestionnaires d'événements aux éléments de contrôle. Un gestionnaire d'événement exécute une procédure prédéfinie quand une action particulière survient. Vous pouvez également éditer des documents ou des bases de données ouvertes avec la gestion des événements ainsi que le contrôle d'accès d'autres éléments.
Les éléments de commande OpenOffice.org permettent de reconnaître différents types d'événements qui peuvent être déclenchés dans différentes situations.

Ces types d'événements peuvent être divisés en quatre groupes:

contrôle de la souris: Les événements qui correspondent à des actions de la souris (par exemple, les mouvements de souris ou d'un clic sur un emplacement de l'écran notamment).
Clavier de contrôle : Les événements qui sont déclenchés par des touches de clavier.
Modification Focus : Les événements qu'exécute OpenOffice.org lorsque des éléments de contrôle sont activés ou désactivée.

Contrôle des éléments spécifiques à des événements: Les événements qui se produisent par rapport à certains éléments de contrôle.

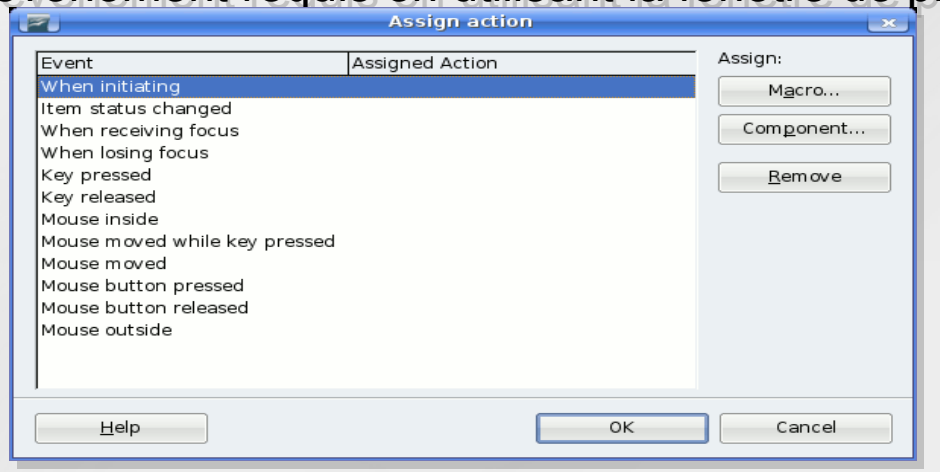
Lorsque vous travaillez avec des événements, assurez-vous que vous créez le dialogue associé dans l'environnement de développement et qu'il contient les éléments de commande ou les documents requis (si vous appliquez les événements à un formulaire).



Le code dans l'exemple suivant déplace une entrée de la boite de liste de gauche vers la boîte de liste de droite du dialogue.

```
Sub cmdSelect_Initiated
Dim lstEntries As Object
Dim lstSelection As Object
lstEntries = Dlg.getControl("lstEntries")
lstSelection = Dlg.getControl("lstSelection")
If lstEntries.SelectedItem > 0 Then
lstSelection.AddItem(lstEntries.SelectedItem, 0)
lstEntries.removeItems(lstEntries.SelectItemPos, 1)
Else
Beep
End If
End Sub
```

Si cette procédure a été créée dans OpenOffice.org Basic, vous pouvez l'assigner à un événement requis en utilisant la fenêtre de propriété de l'éditeur de dialogue.



La boîte de dialogue Assigne une action répertorie tous les événements disponibles. Pour assigner une macro à un événement:

1. Sélectionnez l'événement
2. Cliquez Macro ...
3. Recherchez et sélectionnez la macro que vous souhaitez attribuer
4. Cliquez sur OK

Paramètres

La survenu d'un événement particulier n'est pas toujours suffisante pour une réponse appropriée. Des renseignements supplémentaires peuvent être nécessaires. Par exemple, pour traiter un clic de souris, vous pouvez avoir besoin de la position de l'écran où le bouton de la souris a été pressé.
Dans OpenOffice.org Basic, vous pouvez utiliser les paramètres dans le but de fournir plus d'informations sur un événement à une procédure, par exemple:

Sub ProcessEvent (Event As Object)
End Sub

La structure et les propriétés de l'objet événement dépendra du type d'événement qui déclenche l'appel de procédure.
Peu importe le type d'événement, tous les objets donnent accès à l'élément de contrôle pertinent et son modèle. L'élément de contrôle peut être atteint en utilisant ***Event.Source*** et son modèle en utilisant ***Event.Source.Model***.
Vous pouvez utiliser ces propriétés pour déclencher un événement dans un gestionnaire d'événement.

L'exemple suivant renvoie la position de la souris ainsi que le bouton de la souris qui a été pressé:

```
Sub MouseUp(Event As Object)
Dim Msg As String
Msg = "Keys: "
If Event.Buttons AND com.sun.star.awt.MouseButton.LEFT Then
Msg = Msg & "LEFT "
End If
If Event.Buttons AND com.sun.star.awt.MouseButton.RIGHT Then
Msg = Msg & "RIGHT "
End If
If Event.Buttons AND com.sun.star.awt.MouseButton.MIDDLE Then
Msg = Msg & "MIDDLE "
End If
Msg = Msg & Chr(13) & "Position: "
Msg = Msg & Event.X & "/" & Event.Y
MsgBox Msg
End Sub
```

L'exemple suivant utilise la propriété KeyCode pour établir si la touche Entrée, la touche Tab, ou l'une ou l'autre des touches de contrôle a été enfoncée. Si une de ces touches a été enfoncée, le nom de la clé est retourné, sinon le caractère qui a été tapé est retournée:

```
Sub KeyPressed(Event As Object)
Dim Msg As String
Select Case Event.KeyCode
Case com.sun.star.awt.Key.RETURN
Msg = "Return pressed"
Case com.sun.star.awt.Key.TAB
Msg = "Tab pressed"
Case com.sun.star.awt.Key.DELETE
Msg = "Delete pressed"
Case com.sun.star.awt.Key.ESCAPE
Msg = "Escape pressed"
Case com.sun.star.awt.Key.DOWN
Msg = "Down pressed"
Case com.sun.star.awt.Key.UP
Msg = "Up pressed"
Case com.sun.star.awt.Key.LEFT
Msg = "Left pressed"
Case com.sun.star.awt.Key.RIGHT
Msg = "Right pressed"
Case Else
Msg = "Character " & Event.KeyChar & " entered"
End Select
MsgBox Msg
End Sub
```

Éléments de dialogue, les Contrôles

OpenOffice.org Basic reconnaît un éventail d'éléments de contrôle qui peuvent être divisés selon les groupes suivants:

Entry fields	Buttons	Selection lists	Other
Text fields	Standard buttons	List boxes	Scrollbars (horizontal and vertical)
Date fields	Checkboxes	Combo-boxes	Fields of groups
Time fields	Radio Buttons	Tree Control	Progress bars
Numerical fields			Dividing lines (horizontal and vertical)
Currency fields			Graphics
Fields adopting any format			File selection fields

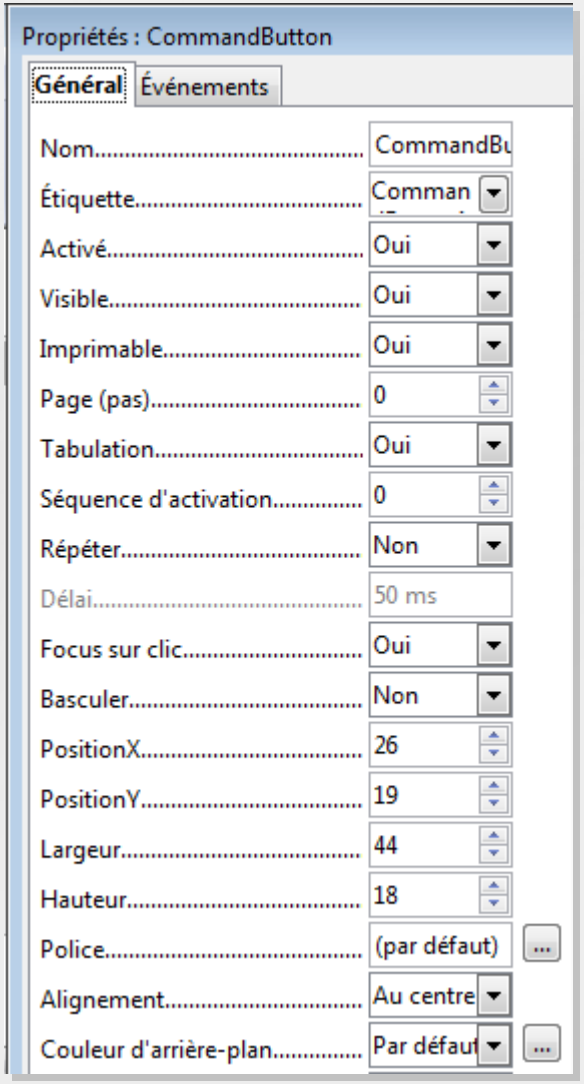
Accéder aux éléments de contrôle individuellement

Un dialogue plusieurs éléments de contrôles Vous pouvez accéder a ces éléments au travers de la méthode **getControl** qui retourne le nom de l'élément de contrôle

```
Dim Ctl As Object
Ctl = Dlg.getControl("MyButton")
Ctl.Label = "New Label"
```

Ce code détermine pour le contrôle **MyButton** et initialise la variable objet **Ctl** avec une référence à l'élément. Finalement le code affecte la propriété Label du contrôle avec la nouvelle valeur du label

Note – contrairement aux identifiants du basic OpenOffice.org, le nom des contrôles est sensible à la case.



L'Objet Model permet d'accéder a chacune dés propriétés du contrôle, ces propriétés peuvent être listées en ouvrant la fenêtre de propriété du contrôle Dans l'EDI (édition des boites de dialogues).

```
Sub cmdNext_Initiated

Dim cmdNext As Object
Dim cmdPrev As Object
cmdPrev = Dlg.getControl("cmdPrev")
cmdNext = Dlg.getControl("cmdNext")
cmdPrev.Model.Enabled = Not cmdPrev.Model.Enabled
cmdNext.Model.Enabled = False
Dlg.Model.Step = Dlg.Model.Step + 1

End Sub
```

Utilisation de graphiques dans les feuilles de calcul

Les graphiques dans des tableurs peuvent afficher les données d'une plage de cellules affectées au tableur.

Toute modification apportée aux données contenues dans la feuille de calcul sera également reflétée dans le tableau attribué.

L'exemple suivant montre comment créer un tableau attribué à certaines plages de cellules dans un document de tableur:

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object
Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress
Doc = ThisComponent
Charts = Doc.Sheets(0).Charts
Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000
RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12
Charts.addNewByName("MyChart", Rect, RangeAddress(), True,
True)
```

L'exemple suivant crée un tableau avec un titre "String Main Title", un sous-titre "String sous-titres" et une légende. La légende a un fond de couleur grise, est placé au bas du tableau, et a une taille de caractères de 7 points.

```
Dim Doc As Object
Dim Charts As Object
Dim Chart as Object
Dim Rect As New com.sun.star.awt.Rectangle
Dim RangeAddress(0) As New com.sun.star.table.CellRangeAddress
Rect.X = 8000
Rect.Y = 1000
Rect.Width = 10000
Rect.Height = 7000
RangeAddress(0).Sheet = 0
RangeAddress(0).StartColumn = 0
RangeAddress(0).StartRow = 0
RangeAddress(0).EndColumn = 2
RangeAddress(0).EndRow = 12
Doc = ThisComponent
Charts = Doc.Sheets(0).Charts
Charts.addNewByName("MyChart", Rect, RangeAddress(), True, True)
Chart = Charts.getByName("MyChart").EmbeddedObject
Chart.HasMainTitle = True
Chart.Title.String = "Main Title String"
Chart.HasSubTitle = True
Chart.Subtitle.String = "Subtitle String"
Chart.HasLegend = True
Chart.Legend.Alignment = com.sun.star.chart.ChartLegendPosition.BOTTOM
Chart.Legend.FillStyle = com.sun.star.drawing.FillStyle.SOLID
Chart.Legend.FillColor = RGB(210, 210, 210)
Chart.Legend.CharHeight = 7
```

Documentation :

Les documents cité ci dessous peuvent êtres librement téléchargeable sur Internet ou à l'adresse suivante http://tondeurh.fr | onglet OpenOffice & LibreOffice

- Basic Guide Open Office (Oracle) : **BasicGuide_OOo3.2.0.pdf**
- Comment Utiliser des Macros Basic dans OpenOffice.org de Daniel Strome : **ht01_basic.pdf**

OpenOffice.org 3.1 Developer's Guide **: DevelopersGuide_OOo3.1.0.odt**