Programmation d'OpenOffice (Macro-commandes).

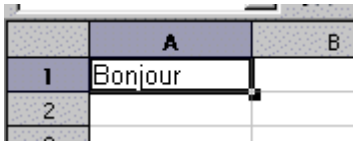
Comme on l'a vu par ailleurs, OpenOffice est un tableur très puissant qui dispose d'un très grand nombre de fonctions (mathématiques, manipulation de texte, mise en forme ...); il peut aussi être programmé pour réaliser des tâches répétitives, ce qui est l'objet de cette étude.

Ce sont des commandes regroupant plusieurs actions élémentaires; si, par exemple, on utilise souvent des titres écrits en police Times New Roman, de taille 20, caractères gras et de couleur rouge, il est fastidieux de devoir répéter la même séquence d'opérations à chaque fois. On va voir sur cet exemple comment créer une nouvelle commande et comment l'utiliser.

I Création de la macro (enregistrement).

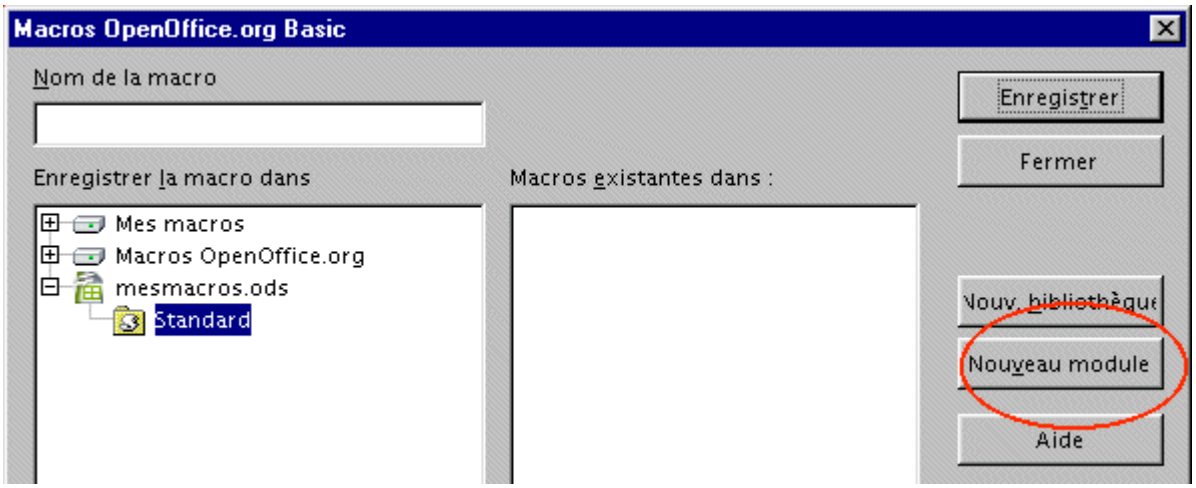
Ouvrir une nouvelle feuille de calcul et l'enregistrer sous le nom "mesmacros".

Ecrire un texte quelconque en A1 puis sélectionner cette case :

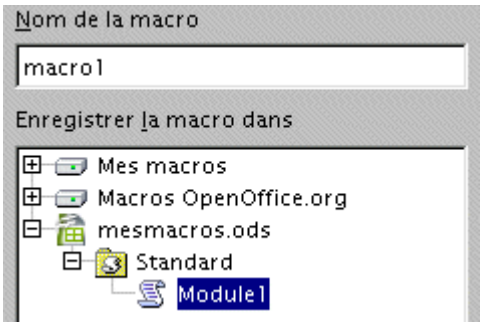


Aller au menu et choisir ***Outils/Macros/Enregistrer une macro...*** : un nouveau bouton apparaît qui servira à indiquer que l'enregistrement est terminé; en attendant, tout ce qu'on fera sera transformé en programme.

- Réaliser les opérations permettant d'obtenir le résultat voulu (Times New Roman gras 20 rouge) puis cliquer sur le bouton "Terminer l'enregistrement".
- L'écran suivant apparaît :

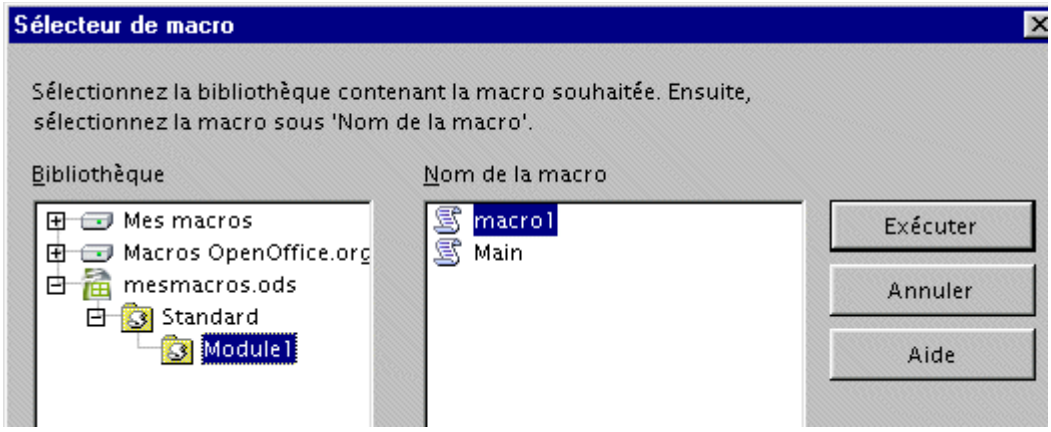


La "bibliothèque" Standard de notre document (mesmacros.ods) étant sélectionnée, cliquer sur **nouveau module**, (garder ou changer le nom proposé)



Puis donner un nom à cette macro (ici macro1) et cliquer sur "**Enregister**". On a ainsi créé une macro nommée **macro1** rangée dans le module **module1** de la bibliothèque **Standard** de **mesmacros.ods** !

- Tester la macro : écrire une phrase dans une cellule, valider, sélectionner cette cellule puis exécuter ***Outils/Macros/Exécuter la macro...***,



Aller chercher Macro1 en déroulant l'arborescence fichier/bibliothèque/module et cliquer sur le bouton **Exécuter**. Cette nouvelle commande fonctionne-t-elle ? ..... .

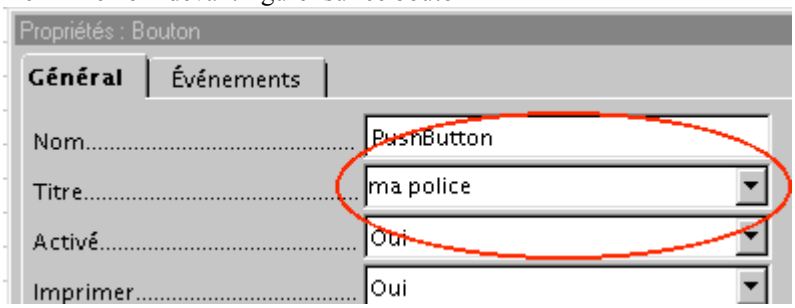
II On peut maintenant créer un bouton qui lancera cette macro :

- Si elle n'est pas visible, afficher la barre d'outils **Formulaire** :

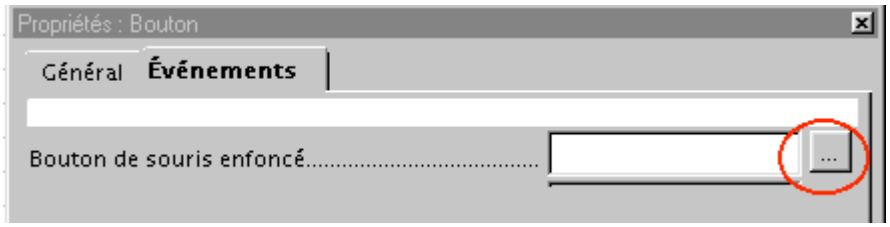
puis choisir **bouton** et en dessiner le contour : 

- D'un clic droit sur ce bouton choisir 

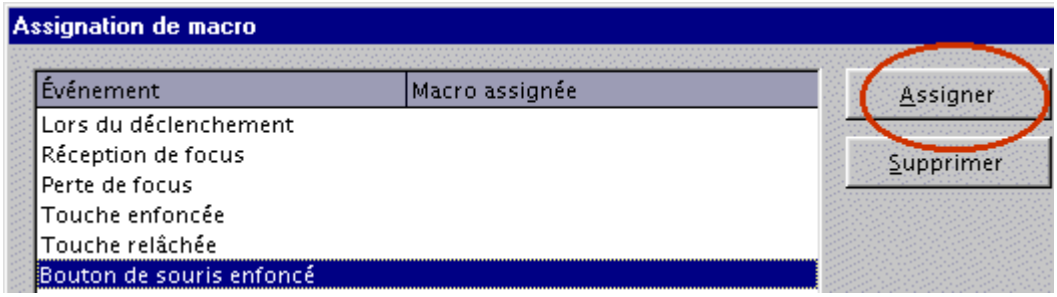
,
- Définir le nom devant figurer sur ce bouton



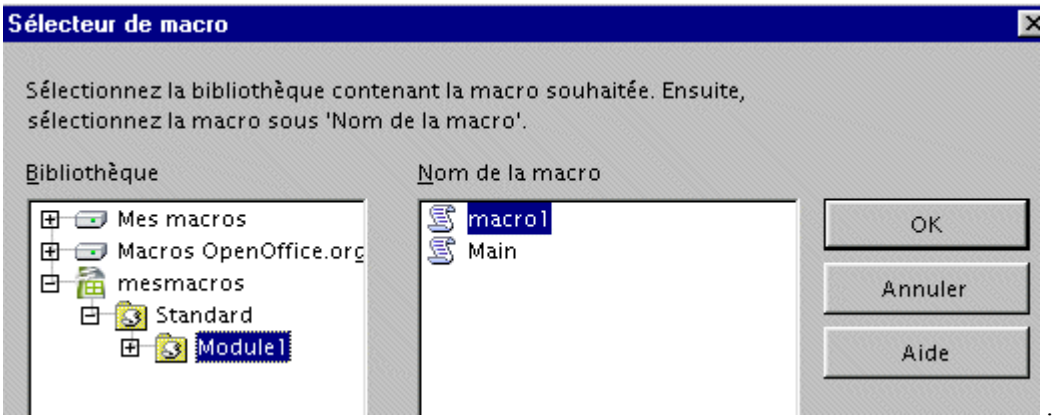
- Cliquer sur l'onglet Evénements pour lui affecter la macro (cette macro sera exécutée lorsqu'on enfoncera la souris sur ce bouton) :

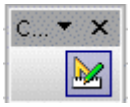


puis



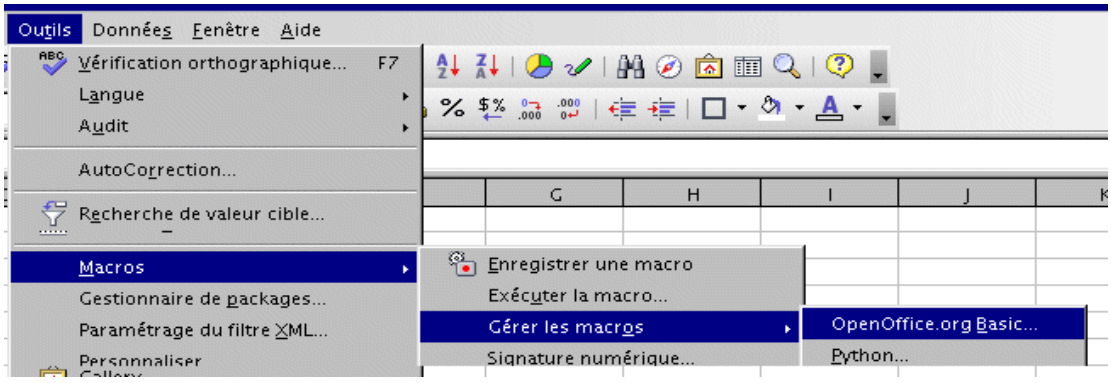
et enfin :



Lorsque tout est terminé, il suffit de cliquer sur le bouton **fin du mode conception**  (dans la barre d'outils **Formulaire**).

Analyse du programme.

exécuter ***Outils/ Macros/Gérer les macros/OO Basic*** :



choisir macro1 puis cliquer sur **Editer**.

Le programme correspondant aux macros apparaît alors. Il peut être ainsi amélioré, transformé... On remarque plusieurs points sur sa structure:

- Des commentaires peuvent être écrits, ils doivent être précédés d'une apostrophe ou du mot REM.
- La macro correspond à une **procedure** (mot-clé **sub**)

```
sub macro1
rem ----------------------------------------------------------------------
rem define variables
dim document as object
dim dispatcher as object
rem ----------------------------------------------------------------------
rem get access to the document
document = ThisComponent.CurrentController.Frame
dispatcher = createUnoService("com.sun.star.frame.DispatchHelper")

rem ----------------------------------------------------------------------
dim args1(2) as new com.sun.star.beans.PropertyValue
args1(0).Name = "FontHeight.Height"
args1(0).Value = 20
args1(1).Name = "FontHeight.Prop"
args1(1).Value = 100
args1(2).Name = "FontHeight.Diff"
args1(2).Value = 0

dispatcher.executeDispatch(document, ".uno:FontHeight", "", 0, args1())

rem ----------------------------------------------------------------------
dim args2(0) as new com.sun.star.beans.PropertyValue
args2(0).Name = "Bold"
args2(0).Value = true

dispatcher.executeDispatch(document, ".uno:Bold", "", 0, args2())

rem ----------------------------------------------------------------------
dim args3(0) as new com.sun.star.beans.PropertyValue
args3(0).Name = "Color"
args3(0).Value = 16711680

dispatcher.executeDispatch(document, ".uno:Color", "", 0, args3())


end sub
```

On remarque que la structure de ce programme est assez complexe mais avec quelques propriétés remarquables :

- Un entête définit l'objet classeur et le gestionnaire,
- chaque action enregistrée a provoqué l'écriture d'un bloc commençant par **rem ----------**,
- pour chacune, un tableau d'argument est créé (args2(0) par exemple) mot-clé : **dim**,
- ce tableau est rempli de noms et de valeurs
- ces arguments sont envoyés à un gestionnaire **executeDispatch**.

Création d'une macro (sans enregistrement).

Aller à la fin de la page Module1 et taper les lignes suivantes:

```
sub essai
rem ----------------------
 Dim Doc As Object
 Dim Sheet As Object
 Dim Cell As Object

  Doc = StarDesktop.CurrentComponent
  sheet = doc.getCurrentController.getactivesheet()
rem ----------------------
```

```
dim i as integer

for i=0 to 10
  Cell = Sheet.getCellByPosition(i, 0)
  Cell.Value = i*i
  Cell.CellBackColor = RGB(110, 210, 110)
  Cell.HoriJustify = com.sun.star.table.CellHoriJustify.RIGHT
next i
end sub
```

Retourner à la feuille1 dans le tableur et tester cette nouvelle macro appelée **essai** ; que fait-elle ?.

Explications :

La première partie (entre les deux rem ----------) est une introduction qui définit le document de travail, la feuille ouverte et la cellule que l'on va remplir. *Il faudra toujours écrire ce bloc lorsqu'on écrira de petits programmes agissant sur les cellules du tableur*.

Ensuite vient le travail de programmation, on y trouve :

une **boucle** (itération) :

for i=0 to 10 (l'indice i variant de 0 à 10)

,,,

,,,

next i (fin de boucle, augmenter i d'une unité et recommencer)

la **désignation d'une cellule** :

Cell = Sheet.getCellByPosition(i, 0) (cell est la cellule colonne n° i, ligne n° 0)

les **modifications** de cette cellule :

Cell.Value = i*i le nombre i au carré est **affecté** à cette cellule
Cell.CellBackColor = RGB(110, 210, 110) la **couleur de fond** vaut 110 (rouge et bleu) 210 (vert)
Cell.HoriJustify = com.sun.star.table.CellHoriJustify.CENTER la **justification** horizontale est centrée.

Exercice : écrire un programme permettant de réaliser une table de multiplication :

x	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100