

Listes de validation dans OpenOffice Calc

Révision [n° 0.2] – 05/08/04

Réalisé avec : OOo 1.1.1

Plate-forme / Os : **Toutes**

Distribué par le projet Fr.OpenOffice.org

Table des Matières

1	But de ce how-to	3
2	Méthode n°1 : la fonction « Liste de sélection »	3
2.1	Principe	3
2.2	Mise au point	3
2.3	Comment empêcher les saisies directes ?	4
2.4	Conclusion	5
3	Méthode des champs de formulaire	5
3.1	Et pourquoi pas une base de données ?	5
3.2	Les avancées de la version 1.1.1 d'OpenOffice	6
3.3	Le complément proposé	6
3.4	Conclusion	8
4	Troisième méthode : une macro et c'est (presque) tout !	8
4.1	Principe	8
4.2	La macro	10
4.3	Conclusion	13
5	Crédits	14
6	Licence	14

1 But de ce how-to

Ce how-to est destiné à fournir une illustration d'ensemble des possibilités d'utilisation de listes de validation dans OpenOffice.org Calc.

Excel dispose au sein de son menu Données/Validation, de la possibilité bien pratique de définir une liste de validation, dont on indique les coordonnées sur la feuille. Les cellules ainsi « verrouillées » se trouvent alors munies d'une flèche déroulante encadrant et facilitant la saisie. Tout en aidant l'utilisateur, on s'assure du même coup de l'homogénéité des données entrées ce qui garantit leur bonne exploitation ultérieure (consolidation et traitement). Il n'est pas étonnant dès lors que les tableaux Excel soient largement employés dans les organisations pour faire remonter des données lorsque les 'serveurs de formulaire' ne sont pas trop développés.


Si Calc comporte, comme Excel, la possibilité de définir des critères de validation sur les données entrées dans une plage de cellules (menu Données/Validité), les listes de validation ne sont hélas pas prises en charge pour l'instant, du moins pas de façon aussi simple et transparente qu'avec Excel.

En attendant une intégration dans une prochaine version (il semble que la v2 constituera l'avancée décisive...), nous allons voir comment parvenir dès maintenant à un résultat proche avec OpenOffice.org en utilisant des options existantes ou des macros. On souhaite notamment pouvoir traiter des plages entières de données.

2 Méthode n°1 : la fonction « Liste de sélection »

2.1 Principe

Cette commande **OpenOffice.org** Calc est associée à la combinaison de touches CTRL+D mais peut bien sûr être réaffectée. Lorsqu'on la déclenche dans une cellule, elle propose dans une liste toutes les valeurs textuelles de la colonne, qui auront été placées par exemple dans les premières lignes de cette colonne.



	A	B	C
1	TITRE		
2			
3	Un		
4	Deux		
5	Trois		
6	Quatre		
7	Cinq		
8	Six		
9			
10	TITRE		
11	Cinq		
12	Deux		
13	Quatre		
14	Six		
15	Trois		
16	Un		
17			
18			
19			
20			
21			

2.2 Mise au point

La commande comporte un certain nombre d'inconvénients, qui pour la plupart, peuvent être contournés. Le tableau suivant les présente.

Inconvénients relevés	Contournements proposés
La liste ne présente que des valeurs textuelles	Formater les nombres et les dates en tant que texte en les précédant de ' (apostrophe) Attention : il faut désactiver la correction automatique des guillemets simples ! On notera que ces valeurs de liste textuelles seront bien transformées en nombres ou dates à l'utilisation.
La liste est encombrante	Masquer les lignes concernées
Le titre de la colonne est pris comme valeur dans la liste	Remplacer les titres par des 'zones de texte' ou mieux, afin de conserver la notion de titre, les saisir en tant que formule par : ="TITRE"
La liste est triée par ordre alphabétique ce qui peut être gênant (ex : liste des jours...)	Pas de solution véritable ici mais on peut toujours forcer un ordre déterminé en faisant précéder les valeurs de la liste par un numéro, au prix cependant d'un embarquement de ces numéros dans les saisies.

2.3 Comment empêcher les saisies directes ?

Il faut empêcher l'utilisateur de contourner le cadrage établi en saisissant directement n'importe quelle valeur. C'est de plus vital ici car toute valeur saisie différente de celles présentes dans une liste s'ajoutera à cette liste dès la saisie suivante !

Pour établir ce blocage, on utilisera astucieusement les fonctionnalités offertes par le menu « Données/Validité ». On peut ainsi n'admettre par exemple que des valeurs de longueur de texte égale à 0 c'est à dire en fait aucune ! Ne pas omettre d'établir l'action 'STOP' et le message d'erreur. On notera cependant que la protection n'est pas parfaite puisqu'elle n'empêche pas les écritures par copier/coller ou recopie de série ; on ne peut pas en effet utiliser la protection de feuille qui bloquerait aussi...l'emploi de CTRL+D !

On peut améliorer le dispositif en prévoyant dans le même menu un message d'aide utile et du plus bel effet (affichage d'une infobulle).





On notera en passant que la rubrique « Action » peut référencer une macro ce qui permet d'envisager des développements intéressants sur le thème des procédures événementielles de cellules...

2.4 Conclusion

Cette méthode un peu 'artisanale' paraît recevable en première approche à condition de préparer le tableau avec soin grâce aux astuces présentées plus haut.

3 Méthode des champs de formulaire

Des connaissances au niveau de la manipulation des champs de formulaire sont requises pour la mise en œuvre de cette méthode.

3.1 Et pourquoi pas une base de données ?

Compte tenu de la nature du problème posé, il est légitime de penser à l'utilisation d'une base de données couplée à un formulaire de saisie d'autant que cette voie est bien dans la philosophie d'OpenOffice. La base contiendrait à la fois la table recueillant les données saisies ainsi que la ou les tables servant de sources aux listes.

Toutefois cette voie est loin d'être simple tant à la conception qu'à l'utilisation :

- ➔ connaissances requises en bases de données et en interface de formulaire, y compris pour contourner des anomalies existantes à ce niveau ;
- ➔ installation d'un driver ODBC ou autre dans Windows ;
- ➔ déclaration d'une source de données dans OpenOffice ;
- ➔ problème du transport de la base d'un poste à l'autre, aggravé lorsque la base est répartie en plusieurs fichiers (cas de MySQL) ;
- ➔ traitement ultérieur des données plus compliqué
- ➔ rappelons également que l'emploi d'une feuille Calc comme source de données n'est pas possible car les données sont en lecture seule au travers du formulaire.

En définitive ce système semble davantage correspondre à des environnements dans lesquels la bureautique est étroitement intégrée à l'informatique générale quoique même dans ce cas, il serait sans doute aujourd'hui plus pertinent de s'orienter vers des solutions Web à base par exemple de serveurs de formulaires.

En conséquence, cette option ne sera pas présentée dans le cadre de cet How-to.

Pour la méthode exposée dans cette partie, les champs de formulaire ne seront donc pas liés à une source de données ; c'est pourquoi lors de leur création à l'aide de la barre d'outils de formulaire, il faudra, si l'autopilote de contrôle de formulaire est actif (cas général), faire 'Annuler' au premier écran.

3.2 Les avancées de la version 1.1.1 d'OpenOffice

La possibilité de lier un champ de formulaire (ListBox ou ComboBox), en entrée avec une plage de cellules, et en sortie avec une cellule synchronisé au choix dans la liste, dont la perspective avait été annoncée dans le How-to « Lier un contrôle de formulaire à une cellule de classeur », a été curieusement implémentée à l'occasion de la version mineure 1.1.1.

	A	B	C	D
1	3			1
2				2
3				3
4				4
5				

The screenshot shows a spreadsheet with columns A-F and rows 1-5. Cell A1 contains the value '3'. Cell B1 contains a list box with the value '3' selected. The list box is linked to the range D1:D4. The 'Propriétés : Zone de liste' dialog is open, showing the following settings:

- Général: Données
- Cellule liée: A1
- Contenu de la cellule liée: L'entrée sélectionnée
- Plage de cellules source: D1:D4
- Champ de données: (empty)
- Champ lié: 1
- Type du contenu de liste: Liste de valeurs
- Contenu de liste: (empty)

At the bottom, the 'Fonctions de formulaire' toolbar is visible, showing various form control icons.

NB : La plage de cellules source peut être placée sur une autre feuille du classeur ; il suffit de préfixer l'adresse par le nom de la feuille (ex : Feuille2.B2:B5).

Hélas toutes ces possibilités, si elles peuvent convenir pour quelques cellules à servir, ne sont plus du tout utilisables dès qu'il s'agit de plages importantes (il faudrait mettre en place des centaines de champs de formulaire !). Sauf à compléter le dispositif avec une macro...

3.3 Le complément proposé

Une macro sera chargée de déterminer la cellule active pour y injecter la valeur choisie dans la liste. Les champs de formulaire pourront alors servir comme cartouche unique de saisie. On les positionnera par exemple en tête des colonnes dans un volet figé.

	A	B	C	D
1				
2	DeuxBis	10000		
3		10001		
4		10002		
5		10003		
6		10004		
7				
8	Trois	10003		
9	Deux	10004		Un
10	DeuxBis	10003		Deux
11	Quatre	10000		DeuxBis
12	Trois	10004		Trois
13				Quatre
14	Saisie:			
15	Pour saisir dans la cellule, utiliser la liste en tête de colonne.			
16				
17				

Volet figé

La propriété 'Cellule liée' sera laissée vide. Par contre, l'évènement 'Souris relevée' sera lié à la macro 'SertCellule' dont voici un code possible :

```

sub SertCellules (Event as object)

2  dim oCell as object, oCoord as object, oControl as object, oControlM as object
    dim numChoixListe as integer, noCol as integer

4  oControl = Event.Source 'qui m'appelle ?
    oControlM = oControl.Model

6  'calcul num de colonne (limité à 26 col...)
    noCol = asc(ucase(oControlM.tag)) - 65 'récupérée de propriété "Info complémentaire"

8  if CalcQuelleTypeSelection() <> 1 then
    msgbox ("Une seule cellule doit être sélectionnée !",48,"Attention:")
10  exit sub
end if

12 oCell = thisComponent.currentSelection
    oCoord = oCell.cellAddress

14 if oCoord.column <> noCol then
    msgbox ("Utiliser la bonne liste !",48,"Attention:")
16  exit sub
end if

18 if oControlM.supportsService("com.sun.star.form.component.ListBox") then
    oCell.formula = oControl.selectedItem 'LIST SEUL
20 elseif oControlM.supportsService("com.sun.star.form.component.ComboBox") then
    oCell.formula = oControl.selectedText 'COMBO
22 end if
'numChoixListe = oControl.selectedItems(0) ' le n° de l'item choisi LIST
24 'oCell.formula = oControl.stringItemList(numChoixListe) 'LIST

end sub

26 function CalcQuelleTypeSelection as integer
    dim oSelection as object, nReturn as integer
28 oSelection = thisComponent.currentSelection
    If oSelection.supportsService("com.sun.star.sheet.SheetCell") Then
30     nReturn = 1 ' une cellule
    ElseIf oSelection.supportsService("com.sun.star.sheet.SheetCellRange") Then
32     nReturn = 2 ' une zone
    ElseIf oSelection.supportsService("com.sun.star.sheet.SheetCellRanges") Then
34     nReturn = 3 ' plusieurs zones
    End If
36 CalcQuelleTypeSelection = nReturn
end function

```

Ce programme vérifie qu'il n'y a pas de sélection dans la feuille (une seule cellule active !) et contrôle également qu'on utilise la bonne liste dans la bonne colonne. C'est pour permettre ce dernier contrôle qu'il faut indiquer le numéro de colonne (en lettre) dans la propriété

« Info complémentaire » de chaque champ liste.

Le choix de l'évènement « souris relevée » permet d'injecter la valeur en seul clic.

Le verrouillage de la zone de saisie est assuré par la méthode expliquée au 2.3 Comment empêcher les saisies directes ?.

3.4 Conclusion

Pour le concepteur, cette méthode a l'inconvénient de requérir des manipulations sur les champs de formulaires pour être mis en place, en plus de la macro. La place occupée en haut de la feuille peut gêner mais sera réduite en employant des ComboBox au lieu de ListBox.

Pour l'utilisateur, le principe du cartouche de saisie peut dérouter mais la vitesse de saisie est très rapide si on s'organise bien (sélection de la zone avec la cellule active en haut à gauche, utilisation de la touche TAB dans la main gauche et souris dans la main droite sur les listes, le contraire pour un gaucher avec la touche ENTER).

Rappelons également que cette méthode ne fonctionne que depuis la version 1.1.1 d'OpenOffice. Avec une version antérieure, il faudrait également gérer par macro le remplissage des listes (voir le How-to « Lier un contrôle de formulaire à une cellule de classeur »).

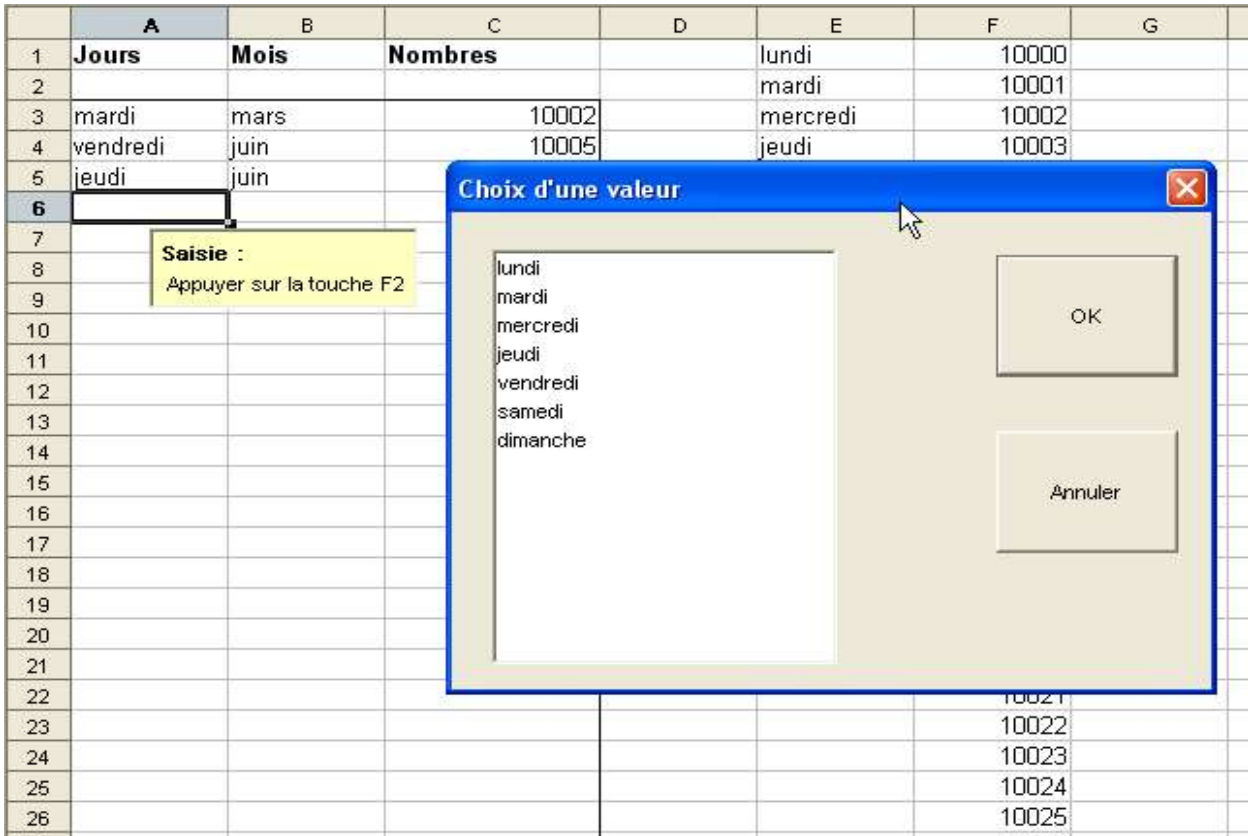
La méthode des champs de formulaire ne sera donc pas conseillée d'autant qu'on va le voir, le système exposé au point suivant combine tous les avantages. Mais outre son caractère illustratif des outils d'OpenOffice, elle pourra servir de point de départ à des développements alternatifs : on citera comme exemple la génération de champs de formulaires par programmation et leur positionnement sur la cellule active.

4 Troisième méthode : une macro et c'est (presque) tout !

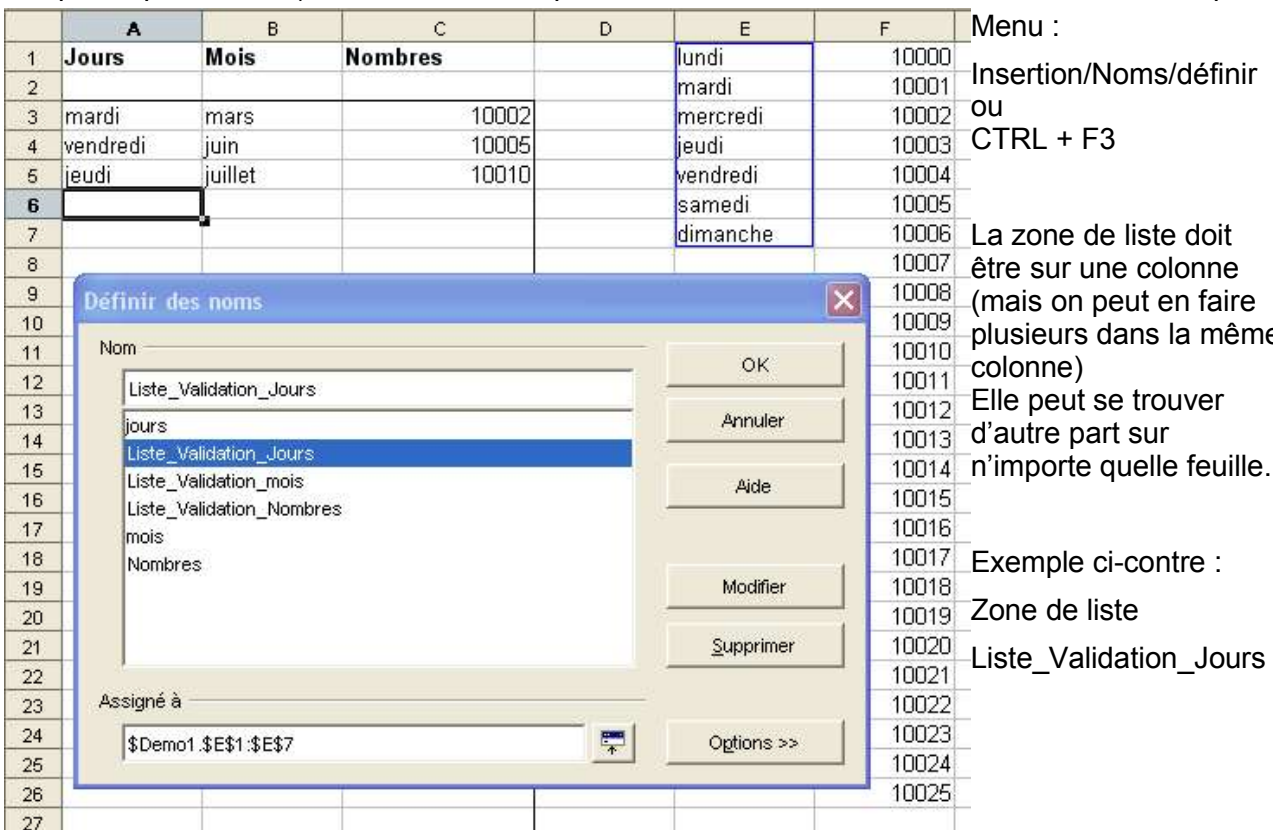
4.1 Principe

La macro, appelée par une touche, va se charger de présenter la liste adéquate dans un « pop-up » et d'injecter la valeur choisie dans la cellule active.

Un seul clic sur un item de liste suffit à injecter la valeur ce qui procure la même rapidité que la méthode précédente. Les boutons OK a cependant été maintenu par sécurité (dans ce cas, on emploiera le clavier pour sélectionner l'item).



On nommera 'Liste_Validation_XXX' chaque zone abritant une liste (qui peut se trouver sur n'importe quelle feuille), le « XXX » correspondant au nom de la zone à saisir. Voici un exemple :



	A	B	C	D	E	F	G
1	Jours	Mois	Nombres		lundi	10000	...qui correspond à la
2					mardi	10001	zone de saisie
3	mardi	mars	10002		mercredi	10002	nommée :
4	vendredi	juin	10005		jeudi	10003	Jours
5	jeudi	juillet	10010		vendredi	10004	
6					samedi	10005	
7					dimanche	10006	
8						10007	
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							
21							
22							
23							
24							
25							
26						10025	
27							
28							
29							
30							
31							

Définir des noms

Nom

- jours
- jours
- Liste_Validation_Jours
- Liste_Validation_mois
- Liste_Validation_Nombres
- mois
- Nombres

Assigné à

\$Demo1.\$A\$3:\$A\$30

OK
Annuler
Aide
Modifier
Supprimer
Options >>

La casse des noms donnés n'a pas d'importance.

Le verrouillage de la zone de saisie est assuré cette fois par la protection de feuille (sans mot de passe), que la macro va gérer si elle existe (protection retirée avant l'écriture et replacée après). On obtient ainsi une sécurité renforcée contre les saisies directes. Notez qu'il peut s'avérer alors utile d'inclure une cellule vide (ou munie d'un libellé ayant cette signification) dans les zones de liste.

Le menu Données/Validité conserve cependant une utilité pour disposer d'une infobulle d'aide sur la zone de saisie (onglet [Aide à la saisie]).

4.2 La macro :

```

38 '*****
39 'Copyright (C) 2004 MINEFI DGDDI
40 'Ecrit par Bruno Moutouh
41 'bruno.moutouh@douane.finances.gouv.fr
42
43 'This library is free software; you can redistribute it and/or
44 'modify it under the terms of the GNU Lesser General Public
45 'License as published by the Free Software Foundation; either
46 'version 2.1 of the License, or (at your option) any later version.
47
48 'This library is distributed in the hope that it will be useful,
49 'but WITHOUT ANY WARRANTY; without even the implied warranty of
50 'MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
51 'Lesser General Public License for more details.
52
53 'You should have received a copy of the GNU Lesser General Public
54 'License along with this library; if not, write to the Free Software
55 'Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA
56 '*****
57
58 REM ***** BASIC *****

```

```

Option Explicit

56 ' variables publiques
dim oDlgListeValidation as object
58 dim bClicDansListe as boolean

Sub SaisieListe

60 ' tous les noms de zones de liste commencent par :
Const NOM_COMMUN_LISTES = "Liste_Validation_"
62 ' suivi du nom de la zone de saisie (casse indifférente)

Const MOT_PASSE_FEUILLE = ""

64 dim oCellSaisie as object, oCellListe as object
dim oListeBox as object
66 dim oNames as object, oName as object, cName as string
dim oNameSaisie as object, cNameSaisie as string
68 dim oNameListe as object, cNameListe as string
dim oZoneSaisie as object, oSheetSaisie as object
70 dim oZoneListe as object, oSheetListe as object
dim oCoordSaisie as object, oCoordListe as object
72 dim numChoixListe as integer, noCol as integer
dim sRef as string, aRef() as string
74 dim sRefSheet as string, sRefZone as string
dim oCoordZoneListe as object, nbLignesZoneListe as integer
76 dim aListe(), i as integer
dim nm as integer, cNomListe as string
78 dim nRetourDlg as integer
dim bProtected as boolean

80 if CalcQuelleTypeSelection() <> 1 then
    msgbox ("Une seule cellule doit être sélectionnée !",48,"Attention:")
82     exit sub
end if

84 oCellSaisie = thisComponent.currentSelection ' où est-on ?

' Recherche sur les noms normalisés de liste
86 ' et récup du nom de zone de saisie correspondant
' puis on vérifie que la cellule active est bien dans cette zone de saisie
88 ' si c'est le cas le popup se déclenche

oNames = thisComponent.namedRanges
90 for nm=0 to oNames.getCount()-1
    oName = oNames.getByIndex(nm)
92     cName = oName.getName
    if left(cName,len(NOM_COMMUN_LISTES))= NOM_COMMUN_LISTES then
94         cNameListe = cName
        oNameListe = oName
96         cNameSaisie = mid(cNameListe,len(NOM_COMMUN_LISTES)+1)
        if not oNames.hasByName(cNameSaisie) then
98             msgbox("Le nom de zone de saisie '"+cNameSaisie+"' est inconnu !",48,"Attention :
problème potentiel")
100             exit sub
        end if
102         oNameSaisie = oNames.getByName(cNameSaisie)
        ' recherche coord zone saisie
104         sRef = oNameSaisie.getContent()
        aRef = split(sRef, ".")
106         sRefSheet = right(aRef(0),len(aRef(0))-1) 'enlever le $ devant pour la suite !!!
        sRefZone = aRef(1)
108         oSheetSaisie = thisComponent.sheets.getByName(sRefSheet)
        oZoneSaisie = oSheetSaisie.getCellRangeByName(sRefZone)
110         oCoordSaisie = oZoneSaisie.RangeAddress
        if CalcIsInZone(oCellSaisie,oZoneSaisie) then
112             ' déclencher le popup
            ' recherche coord zone liste
114             sRef = oNameListe.GetContent()
            aRef = split(sRef, ".")
116             sRefSheet = right(aRef(0),len(aRef(0))-1)
            sRefZone = aRef(1)
118             oSheetListe = thisComponent.sheets.getByName(sRefSheet)
            oZoneListe = oSheetListe.getCellRangeByName(sRefZone)
120             oCoordListe = oZoneListe.RangeAddress

            if oZoneListe.columns.count >1 then

```

```

122     MsgBox("Zone mal définie: ne doit contenir qu'une seule colonne !",48,"Attention:")
123     exit sub
124 end if

    redim aListe(oCoordListe.startRow to oCoordListe.endRow)

126 for i = oCoordListe.startRow to oCoordListe.endRow
    oCellListe = oSheetListe.getCellByPosition(oCoordListe.startColumn, i)
128     aListe(i) = oCellListe.formula
next i

130 ' appel du dialog DLG ListeValidation et remplissage de la liste LB_ListeValidation
DialogLibraries.LoadLibrary("Standard")
132 oDlgListeValidation = createUnoDialog(DialogLibraries.Standard.DLG_ListeValidation)
oListBox = oDlgListeValidation.GetControl("LB_ListeValidation")
134 oListBox.addItem(aListe(),0)

    bClicDansListe = false 'init
136 nRetourDlg = oDlgListeValidation.execute()

    if bClicDansListe or nRetourDlg=1 then
138         ' on écrit
        if oSheetSaisie.isProtected() then bProtected = true
140         if bProtected then oSheetSaisie.unprotect(MOT_PASSE_FEUILLE)
        oCellSaisie.formula = oListBox.selectedItem
142         if bProtected then oSheetSaisie.protect(MOT_PASSE_FEUILLE)
    endif

144     oDlgListeValidation.dispose()
    exit for 'popup actionné, terminé
146 end if 'cell in zone saisie

    endif 'nom de liste à nous repérée

148 next 'autre nom à examiner
end Sub

150 function CalcQuelleTypeSelection as integer
    dim oSelection as object, nReturn as integer

152 oSelection = thisComponent.currentSelection
If oSelection.supportsService("com.sun.star.sheet.SheetCell") Then
154     nReturn = 1 ' une cellule
ElseIf oSelection.supportsService("com.sun.star.sheet.SheetCellRange") Then
156     nReturn = 2 ' une zone
ElseIf oSelection.supportsService("com.sun.star.sheet.SheetCellRanges") Then
158     nReturn = 3 ' plusieurs zones
End If
160 CalcQuelleTypeSelection = nReturn

end function

162 function CalcIsInZone (cell as object, zone as object) as boolean
' détermine si une cellule fait partie d'une zone nommée

164 dim oCoordCell as object, oCoordZone as object

    oCoordCell = cell.cellAddress
166 oCoordZone = zone.rangeAddress

    if oCoordCell.sheet = oCoordZone.sheet and _
168 oCoordCell.column >= oCoordZone.startColumn and _
oCoordCell.column <= oCoordZone.endColumn and _
170 oCoordCell.row >= oCoordZone.startRow and _
oCoordCell.row <= oCoordZone.endRow then
172     CalcIsInZone = true
    else
174     CalcIsInZone = false
    endif

176 end function

sub ChoixListeDirect (oEvent as object)
178 ' lancée par evenement mouseUp du control de liste LB_ListeValidation

```

```
180 if oEvent.source.selectedItemPos >=0 then
    oDlgListeValidation.endExecute()
    bClicDansListe = true
182 endif
end sub
```

La boîte de dialogue « DLG_ListeValidation » avec une zone de liste « LB_ListeValidation » doit être construite au niveau du module abritant la macro.

La macro sera enfin associée à une touche, par exemple F2 (associer SaisieListe) ou F12 pour un gaucher.

4.3 Conclusion

C'est une méthode :

- ➔ transparente : on est libéré de la gestion des champs de formulaire ou des « bricolages » du CTRL+D ;
- ➔ simple à mettre en oeuvre : si on utilise un modèle (comme CalcListesValidationMacro.stc mis en ligne sur l'espace documentaire fr.openoffice.org), les éléments techniques sont automatiquement mis en place ; tout ce qui est requis du concepteur est ensuite de savoir nommer des zones, activer la protection de feuille et l'infobulle d'aide à la saisie ;
- ➔ universelle puisqu'on est libre de définir les zones qui conviennent ;
- ➔ rapide : grâce à l'injection sur simple clic, deux actions élémentaires seulement sont requises (une frappe clavier et un clic de souris) ;
- ➔ la plus puissante potentiellement : on peut disposer de toutes les variantes imaginable en retouchant la macro (par exemple, pour distinguer ce qui est présenté dans le « pop-up » et ce qui sera injecté dans la cellule)

C'est donc cette voie qui a largement ma préférence ce qui n'empêche pas d'espérer par ailleurs une intégration totalement 'Excel-like' dans la prochaine version 2 de votre suite bureautique préférée, qui semble devoir nous réserver une très bonne surprise à ce niveau...

5 Crédits

Auteur : **Bruno Moutouh**

Remerciement : **tous les contributeurs du projet documentation, qui par leurs remarques pertinentes, m'ont permis d'améliorer sensiblement le modèle « CalcListesValidationMacro.stc ».**

Intégré par : **Sophie Gautier**

Dernière modification : **[date]**

Contacts : **Projet Documentation OpenOffice.org - Fr.OpenOffice.org**

Traduction :

6 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 Listes de validation dans OpenOffice Calc The Initial Writer of the Original Documentation is Bruno Moutouh Copyright (C) 2004 MINEFI DGDDI. All Rights Reserved. (Initial Writer contact(s): bruno.moutouh@douane.finances.gouv.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.