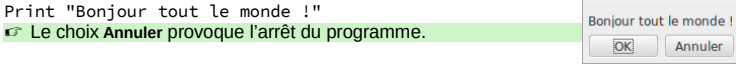
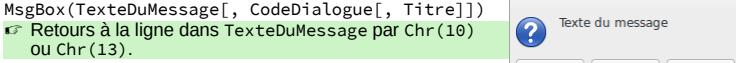


Dialogues Basic

Afficher un message simple



Afficher un message d'information



Afficher un message et lire la réponse

Reponse = MsgBox(TexteDuMessage[, CodeDialogue[, Titre]])
 où

- Reponse est un entier, selon le choix de l'utilisateur.
- CodeDialogue : somme des codes boutons à afficher + icône + bouton par défaut

Boutons à afficher			
0	OK	3	Oui, Non, Annuler
1	OK, Annuler	4	Oui, Non
2	Interrompre, Réessayer, Ignorer	5	Réessayer, Annuler

Icône	
0	(aucune) 48
16	Message critique 64
32	Question

Bouton par défaut			
0	premier	256	deuxième
512	dernier		

Valeurs de retour (choix de l'utilisateur)			
1	OK	3	Interrompre
2	Annuler	4	Réessayer
5	Ignorer	6	Oui
7	Non		

Fonction InputBox()

Fonction InputBox(Invite[, Titre[, ValeurDefault]])
 retourne une chaîne. Si abandon, la chaîne est vide.

Dialogues de l'API

Les types FilePicker et FolderPicker ci-dessous sont influencés par Outils > Options > LibreOffice > Général, Utiliser les boîtes de dialogue LibreOffice.

Types de dialogues proposés par l'API

Sélection de fichier : objets FilePicker
 com.sun.star.ui.dialogs.FilePicker Selon configuration ci-dessus.
 com.sun.star.ui.dialogs.OfficeFilePicker Force le style LibreOffice.
 com.sun.star.ui.dialogs.SystemFilePicker Force le style de l'OS.

Sélection de répertoire : objets FolderPicker
 com.sun.star.ui.dialogs.FolderPicker Selon configuration ci-dessus.
 com.sun.star.ui.dialogs.OfficeFolderPicker Force le style LibreOffice.
 com.sun.star.ui.dialogs.SystemFolderPicker Force le style de l'OS.

L'objet FilePicker (ou OfficeFilePicker ou SystemFilePicker)

oFilePicker = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")
 AppendFilter() Par paires: appendFilter("NomLitteral", "*.xyz")
 Ex: oFilePicker.appendFilter("Documents ODF", "*.odt;*.ods")

CurrentFilter Le filtre par défaut, parmi les filtres ajoutés par AppendFilter (nom littéral) ou le filtre choisi par l'utilisateur.

DefaultName Nom par défaut pour le fichier à enregistrer.

DisplayDirectory Le répertoire initial ou choisi par l'utilisateur.

Execute Transfère le flux d'exécution au dialogue et fournit un code de retour (voir constantes codes de retour)

Files Le tableau des fichiers sélectionnés.

initialize() Choix du type de dialogue (voir les constantes de type).
 Dim FPType(0) As Integer
 FPType(0) = 'la constante de type (ci-dessous)
 oFilePicker.initialize(FPType())

MultiSelectionMode (Dés)Active le mode multi-sélection (False par défaut).

Title Le titre de la fenêtre.

Constantes de types de FilePicker

com.sun.star.ui.dialogs.TemplateDescription.XXX :	
FILEOPEN_SIMPLE	0 Dialogue d'ouverture simple.
FILESAVE_SIMPLE	1 Dialogue d'enregistrement simple.
FILESAVE_AUTOEXTENSION_PASSWORD	2 Dialogue d'enregistrement enrichi : extension automatique + mot de passe.
FILESAVE_AUTOEXTENSION_PASSWORD_FILTEROPTIONS	3 Dialogue d'enregistrement enrichi : extension automatique + mot de passe + options de filtrage.
FILESAVE_AUTOEXTENSION_SELECTION	4 Dialogue d'enregistrement enrichi : extension automatique + sélection.
FILESAVE_AUTOEXTENSION_TEMPLATE	5 Dialogue d'enregistrement enrichi : extension automatique + liste « Modèles ».
FILEOPEN_LINK_PREVIEW_IMAGE_TEMPLATE	6 Dialogue d'ouverture enrichi : insertion comme lien + prévisualisation + modèle.
FILEOPEN_PLAY	7 Dialogue d'ouverture enrichi : jouer.
FILEOPEN_READONLY_VERSION	8 Dialogue d'ouverture enrichi : lecture seule + version.
FILEOPEN_LINK_PREVIEW	9 Dialogue ouverture enrichi : lien + prévisu.
FILESAVE_AUTOEXTENSION	10 Dialogue d'enregistrement enrichi : extension automatique.

FILEOPEN_PREVIEW	11 Dialogue d'ouverture enrichi : prévisu.
FILEOPEN_LINK_PLAY	12 Dialogue d'ouverture enrichi : insertion comme lien + jouer.

Constantes codes de retour

com.sun.star.ui.dialogs.ExecutableDialogResults.XXX	
CANCEL 0	Annulé
OK 1	Validé

L'objet FolderPicker (ou OfficeFolderPicker ou SystemFolderPicker)

oFldrService = CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
 Description Texte d'aide sur le dialogue. Sur un OfficeFolderPicker, ne fait rien.

DisplayDirectory Répertoire initial.

execute Transfère le flux d'exécution au dialogue et fournit un code de retour (voir constantes codes de retour).

Title Le titre du dialogue.

Directory Le choix de l'utilisateur.

Ouvrir un fichier unique (FilePicker)

1. Créez un FilePicker. Le type par défaut convient en général (FILEOPEN_SIMPLE),
2. initialisez-le (propriétés et méthodes ci-dessus),
3. exécutez-le,
4. au retour, lisez les choix de l'utilisateur dans les propriétés CurrentFilter, DisplayDirectory et Files (vecteur) (seul Files(0) contient une valeur).

```
Dim oFilePicker As Object, NomFichier As String
NomFichier = ""
'initialisation du FilePicker
oFilePicker = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")
oFilePicker.DisplayDirectory = pRepDepart
oFilePicker.appendFilter("Classeurs Calc", "*.ods")
oFilePicker.CurrentFilter = "Classeurs Calc"
oFilePicker.Title = "Choisissez le classeur"
'exécution et vérification du retour (OK)
If oFilePicker.execute = _
    com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
    NomFichier = oFilePicker.Files(0)
End If
```

Ouvrir plusieurs fichiers (FilePicker)

1. Comme pour le précédent,
2. initialisez-le (en part. avec MultiSelectionMode = True),
3. exécutez-le,
4. au retour, le vecteur Files() contient les choix de l'utilisateur.

Enregistrer un fichier (FilePicker)

1. Créez un FilePicker,
2. initialisez-le (le type FILESAVE_XXX), (et propriétés et méthodes ci-dessus),
3. exécutez-le,
4. au retour, lisez les choix de l'utilisateur dans les propriétés CurrentFilter, DisplayDirectory et Files (vecteur) (seul Files(0) contient une valeur).

Choisir un répertoire (FolderPicker)

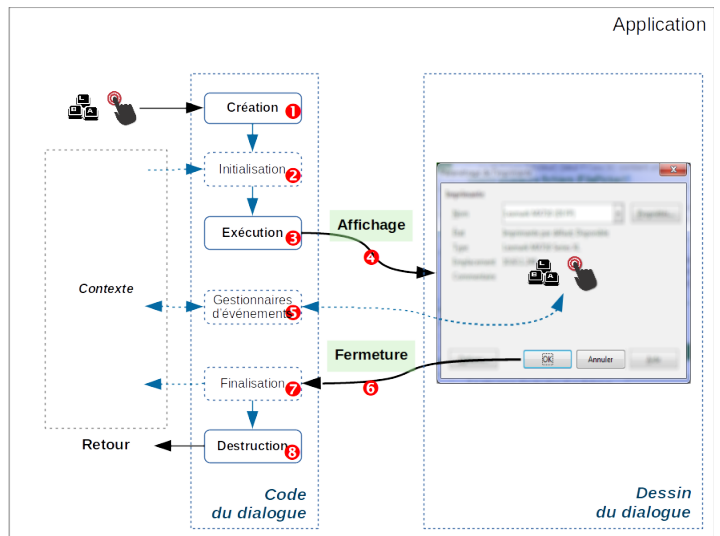
1. Créez un FolderPicker,
2. initialisez-le (propriétés et méthodes ci-dessus),
3. exécutez-le,
4. au retour, lisez le choix de l'utilisateur dans Directory.

```
Dim oFP As Object, NomRep As String
NomRep = ""
oFP = CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
oFP.DisplayDirectory = ConvertToURL("C:\Chemin\Vers\Répertoire")
oFP.Description = "Cliquez sur un répertoire"
oFP.Title = "Choisissez votre répertoire de sauvegarde"
If oFP.execute = _
    com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
    NomRep = oFP.Directory
End If
```

Dialogues personnalisés – Principes

Dialogue Basic = un module Dialogue (dessin) + (au moins) un module de code.

Séquence d'exécution d'un dialogue



- La séquence d'exécution d'un dialogue est illustrée ci-dessus :
1. En réponse à un événement de l'application, vous créez le dialogue ;
 2. (réalisez l'initialisation des contrôles depuis le contexte applicatif si nécessaire) ;
 3. exécutez le dialogue, qui reçoit le flux d'exécution :
 4. affichage,
 5. (gestion des événements sur les contrôles du dialogue),
 6. des événements provoquent la fermeture du dialogue (OK, Annuler) ;

- réalisez les opérations de **finalisation** vers le contexte applicatif si nécessaire) ;
- le dialogue est **détruit et retour** à l'application appelante.

☞ Création, initialisation, exécution, finalisation et destruction : dans votre code.
Affichage, fermeture : opérations automatiques consécutives aux précédentes.
⚠ Prévoyez des réponses aux **événements** sur les contrôles (aide-mémoire n°4).

Chargement des bibliothèques de dialogues

☞ Si beaucoup de code ou si plusieurs dialogues, pensez à les héberger dans des bibliothèques dédiées (une par dialogue).
⚠ Les bibliothèques de dialogues ne sont jamais chargées automatiquement.
⚠ Noms des bibliothèques à charger : respectez la casse !

Modal vs Non modal

Modal Un dialogue modal prend le contrôle total du clavier, de la souris et de l'écran dans l'attente d'une action de l'utilisateur. L'application sous-jacente devient inaccessible.
☞ Par défaut les dialogues sont modaux.

Non modal Un dialogue non modal ne bloque pas l'accès à l'application.
Ex : le dialogue **Rechercher & remplacer** dans LibreOffice
⚠ Attention aux exécutions multiples des dialogues non modaux qui peuvent bloquer l'application.

Dialogues personnalisés ordinaires (modaux)

C'est la situation majoritaire.

Soient un module de dialogue `MonDlg` et un module de code `MonDlgCode` dans la bibliothèque `MaBibliDlg`. Dans une Sub du module de code, on instancie un objet dialogue (`oDlg`) à partir du dialogue.

Création / chargement en mémoire

```
DialogLibraries.loadLibrary("MaBibliDlg")
oBibli = DialogLibraries.getByname("MaBibliDlg")
oModule = oBibli.getByname("MonDlg")
oDlg = CreateUnoDialog(oModule)
'on manipule ensuite oDlg
```

Appel seul

`oDlg.execute` ☞ Le flux d'exécution est transféré au dialogue.

Appel et test du retour

```
If oDlg.execute = com.sun.star.ui.dialogs.ExecutableDialogResults.OK
Then ...
```

☞ Le flux d'exécution est transféré au dialogue et la valeur de retour testée (l'utilisateur a-t-il cliqué OK ?).

Terminaison / destruction du dialogue

`oDlg.dispose`

Exemple récapitulatif (module de code)

Cet exemple ne montre pas de gestion d'événement.

```
Sub AfficherDialogue()
    Dim oBibli As Object, oModule As Object, oDlg As Object

    DialogLibraries.loadLibrary("MaBibliDlg")
    oBibli = DialogLibraries.getByname("MaBibliDlg")
    oModule = oBibli.getByname("MonDlg")
    oDlg = CreateUnoDialog(oModule)
    'InitialiserDlg() 'initialiser le contenu du dialogue
    If oDlg.execute = com.sun.star.ui.dialogs.ExecutableDialogResults.OK
    Then
        'FinaliserDlg() 'faire qqch avec les saisies
    End If
    oDlg.dispose
End Sub
```

Dialogues personnalisés non modaux

Soient un module de dialogue `MonDlgNM` et un module de code `MonDlgCodeNM` dans la bibliothèque `MaBibliDlgNM`. Dans une Sub du module de code, on instancie un objet (`oDlg`) à partir du dialogue.

Appliquez les mêmes principes que ci-dessus, avec des subtilités :

- L'**affichage** du dialogue est assuré par `oDlg.SetVisible(True)` au lieu de `oDlg.execute`,
- deux variables globales booléennes de contrôle :
 - `gEnCours` empêche les exécutions multiples,
 - `gAfficheMoi` contrôle la présence du dialogue à l'écran,
- les **réponses aux événements** (contrôles) doivent mettre `gAfficheMoi` à `False` pour fermer le dialogue.

Affichage du dialogue

`oDlg.SetVisible(True)` ☞ Le dialogue est **affiché**.
Le flux d'exécution n'est **pas** transféré au dialogue.

Exemple récapitulatif (module de code)

```
'contrôle de l'affichage du dialogue. Globale au module !
Dim gAfficheMoi As Boolean
'contrôle des exécutions multiples non désirées. Globale aussi.
Dim gEnCours As Boolean

Sub AfficherDialogueNonModal()
'gère la création et l'affichage du dialogue

    Dim oBibli As Object, oModule As Object, oDlg As Object

    'éviter les lancements multiples
    If Not gEnCours Then
        gEnCours = True
        gAfficheMoi = True
        DialogLibraries.loadLibrary("MaBibliDlgNM")
        oBibli = DialogLibraries.getByname("MaBibliDlgNM")
        oModule = oBibli.getByname("MonDlgNM")
        oDlg = CreateUnoDialog(oModule)
        'InitialiserDlg() 'initialiser le contenu du dialogue

        'afficher le dialogue tant que la var. est True
        Do While gAfficheMoi
            Wait 20 'permettre l'exécution d'autres logiciels
            oDlg.SetVisible(True) 'maintenir à l'écran
        Loop
```

```
'FinaliserDlg() 'faire qqch avec les saisies si besoin
oDlg.dispose
gEnCours = False
End If
End Sub 'AfficherDialogueNonModal

Sub OnBtnOKClick(ByRef pEvt As Object)
'Reponse à clic sur OK

    'réaliser les actions utiles
    'finir par :
    gAfficheMoi = False '=> fin de la boucle while
    'donc la fermeture du dialogue

End Sub 'OnBtnOKClick
```

Associer un événement à une macro

Votre dialogue communique avec l'application via des **événements** (6 du schéma). Vous allez donc créer des macros pour répondre aux occurrences de ces événements (extrait de l'aide-mémoire n°4) :

- Créer la macro** à exécuter selon le modèle ci-dessous :

```
Sub NomDeLaMacro()
End Sub
```

☞ Conseil : nommez la macro selon l'objet, l'action et le type d'événement.
exemple : `Sub OnBoutonOKClick()`

- La Sub peut comporter un paramètre. Voir plus bas « Obtenir des informations... »,
- sélectionnez l'objet** porteur de l'événement à intercepter,
- accédez à sa configuration (méthode variable selon l'objet),
- choisissez l'événement** à intercepter,
- pointez vers la macro** à exécuter sur déclenchement de l'événement (point 1).

☞ Plus d'informations sur les événements dans l'aide-mémoire n°4.

Obtenir des informations sur l'événement déclencheur

La macro de traitement peut interroger le paramètre entrant pour obtenir des informations complémentaires à propos de l'événement :

```
Sub ReponseEvenement(ByRef Event As Object)
End Sub
```

La structure et les propriétés de l'objet `Event` dépendent du type d'événement qui déclenche l'appel de procédure.

Cas fréquents pour les contrôles

Pour accéder à	Interrogez
Objet contrôle appelant	<code>Event.Source</code>
Objet modèle du contrôle	<code>Event.Source.Model</code>
Objet dialogue propriétaire du contrôle	<code>Event.Source.Context</code>

Initialisation et finalisation

Initialisation

(2 du schéma) Le dialogue nécessite souvent des informations provenant du contexte d'exécution. La macro d'initialisation configure le dialogue à partir de ces données.


Finalisation

(7 du schéma) Il s'agit ici de réaliser l'opération inverse de la précédente : actualiser les données contextuelles à partir de celles saisies ou choisies dans le dialogue.


Gérer les modules Dialogue

LibreOffice gère les modules Dialogue indépendamment du code (aide-mémoire n°1). Il est possible de recopier ces objets d'un document vers un autre.


Recopier des modules d'une bibliothèque vers une autre

- Dans l'IDE, ouvrez les deux documents/conteneurs source et cible,
 - ouvrez le **Gestionnaire de macros** (bouton ) ,
 - onglet **Boîtes de dialogue**, glissez/déposez depuis la source vers la cible.
- ☞ Par défaut les modules sont **déplacés**. Pour les **recopier** : **Ctrl** + glisser/déposer.

Sauvegarder une boîte de dialogue seule

- Dans l'IDE, ouvrez le module Dialogue à sauvegarder,
- cliquez le bouton de barre d'outils  **Export de boîte de dialogue**,
- nommez le fichier et enregistrez-le.

Le document est au format XML et doté de l'extension `.xdl`.

☞ L'importation est la réciproque de cette manipulation, au moyen du bouton  **Importer une boîte de dialogue**.

Crédits

Auteur : Jean-François Nifenecker – jean-francois.nifenecker@laposte.net

Nous sommes comme des nains assis sur des épaules de géants. Si nous voyons plus de choses et plus lointaines qu'eux, ce n'est pas à cause de la perspicacité de notre vue, ni de notre grandeur, c'est parce que nous sommes élevés par eux. (Bernard de Chartres [attr.])

Historique

Version	Date	Commentaires
1.01	01/10/2017	Première version
1.05	02/12/2018	Corrections mineures

Licence

Cet aide-mémoire est placé sous licence

Creative Commons BY-SA v3 (fr).

Informations

<https://creativecommons.org/licenses/by-sa/3.0/fr/>

