Guía de referencia LibreOffice

LibreOffice Basic n.º 6
Diálogos

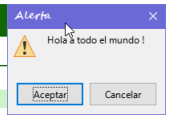
Avanzado

v. 1.05 – 02/12/2018

Redactado con LibreOffice v. 5.3.3 – Plataforma: Todas

Diálogos Basic

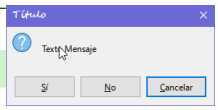
Mostrar un mensaje simple




```
Print "Hola a todo el mundo !"
```

☞ La elección de **Anular** provoca la detención del programa

Mostrar un mensaje de Información




```
MsgBox(TextoMensaje[, Código Diálogo[, Título]])
```

☞ Puede usar un salto de linea en el texto del mensaje con `Chr(10)` o `Chr(13)`.

Mostrar un mensaje y leer la respuesta

`Respuesta = MsgBox(TextoMensaje[, CódigoDiálogo[, Título]])` donde:

- `Respuesta` devuelve un entero según la elección del usuario.
- `CódigoDiálogo` : uno de los códigos de botones a mostrar + icono + botón predet.

Botones a mostrar

0	Aceptar	3	Sí, No, Cancelar
1	Aceptar, Anular	4	Si, No
2	Detener, Reintentar, Ignorar	5	Reintentar, Cancelar

Icono

0	(ninguno)	48	ⓘ Avertencia
16	⚠ Alerta	64	ℹ Información (sólo con Aceptar)
32	❓ Pregunta		

Botón predeterminado

0	primero	256	segundo	512	último

Valores de retorno (según elección del usuario)

1	Aceptar	3	Interrumpir	5	Ignorar	7	No
2	Cancelar	4	Reintentar	6	Si		

InputBox()

```
InputBox(Mensaje [, Título[, Valor predeterminado]])
```

devuelve una cadena de texto, si se cancela la cadena devuelta estará vacía.

Diálogos de la API

☞ Los tipos `FilePicker` y `FolderPicker` de abajo están relacionados con **Herramienas > Opciones > LibreOffice > General >usar los diálogos de L.O.**

Tipos de diálogo provistos por la API

Selección de archivo: objets FilePicker

`com.sun.star.ui.dialogs.FilePicker`	Según configuración indicada
`com.sun.star.ui.dialogs.OfficeFilePicker`	Fuerza el estilo de LibreOffice.
`com.sun.star.ui.dialogs.SystemFilePicker`	Fuerza el estilo del S.O.

Selección de directorio: objets FolderPicker

`com.sun.star.ui.dialogs.FolderPicker`	Según la configuración indicada
`com.sun.star.ui.dialogs.OfficeFolderPicker`	Fuerza el estilo de LibreOffice.
`com.sun.star.ui.dialogs.SystemFolderPicker`	Fuerza el estilo del S.O.

El Objeto FilePicker (u OfficeFilePicker o SystemFilePicker)

`oFilePicker = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")`

`AppendFilter()`	(2 arg.): `appendFilter("NombreLiteral", "*.xyz")` Ej.: `oFilePicker.appendFilter("Documents ODF", _ "*.odt;*.ods")`
`CurrentFilter`	El filtro predeterm., entre los filtros elegidos por `AppendFilter` (nombre litteral) o el filtro elegido por el usuario.
`DefaultName`	Nombre predeterm. para el archivo a guardad.
`DisplayDirectory`	El directorion inicial o elegido por el usuario.
`Execute`	Transfiere el flujo de ejecución al diálogo y provee de un codigo de retorno (ver constantes de códigos de retorno)
`Files`	El panel de archivos seleccionados.

`initialize()`	Elección del tipo de diálogo (ver constantes de tipo).
	`Dim FPType(0) As Integer`
	`FPType(0) = 'la constante de tipo (abajo)`
	`oFilePicker.initialize(FPType())`

`MultiSelectionMode`	(Des)Activa el modo de multi-selección (`False` predeterm.).
`Title`	El título del diálogo.

Constantes de types de FilePicker

`com.sun.star.ui.dialogs.TemplateDescription.XXX:`

`FILEOPEN_SIMPLE`	0 Apertura simple.
`FILESAVE_SIMPLE`	1 Guardado simple.
`FILESAVE_AUTOEXTENSION_PASSWORD`	2 Guardado compuesto: extensión automática + contraseña.
`FILESAVE_AUTOEXTENSION_PASSWORD_FILTEROPTIONS`	3 Guardado compuesto: extensión automática + contraseña + opciones filtro.
`FILESAVE_AUTOEXTENSION_SELECTION`	4 Guardado compuesto: extensión automática + selección.
`FILESAVE_AUTOEXTENSION_TEMPLATE`	5 Guardado compuesto: extensión automática + lista «Plantillas».
`FILEOPEN_LINK_PREVIEW_IMAGE_TEMPLATE`	6 Apertura compuesdo: inserción como sitio + previsualización + plantilla.
`FILEOPEN_PLAY`	7 Apertura compuesto: reproducir.
`FILEOPEN_READONLY_VERSION`	8 Apertura compuesto: sólo lectura + versión.
`FILEOPEN_LINK_PREVIEW`	9 Apertura compuesto: sitio + prev.
`FILESAVE_AUTOEXTENSION`	10 Guardado compuesto: extensión autom.
`FILEOPEN_PREVIEW`	11 Apertura compuesto: previsualizar.
`FILEOPEN_LINK_PLAY`	12 Apertura compuesto: inserción como sitio + reproducir.

Constantes de códigos de retorno

`com.sun.star.ui.dialogs.ExecutableDialogResults.XXX`

`CANCEL`	0	Cancelar	`OK`	1	Aceptar

El objeto FolderPicker (u OfficeFolderPicker o SystemFolderPicker)

`oFldrPicker = CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")`

`Description`	Texto de ayuda (sobre el diálogo, sobre un `OfficeFolderPicker`, no se visualiza).
`DisplayDirectory`	Directorio inicial.
`execute`	Transfiere el flujo de ejecución al diálogo y provee de un codigo de retorno (ver constantes de códigos de retorno)
`Title`	El título del diálogo.
`Directory`	Elegido por el usuario

Abrir un único archivo (FilePicker)

1. Crear un `FilePicker`. El tipo predeterm. normalmente es (`FILEOPEN_SIMPLE`),
2. inicializarlo (propiedades y métodos abajo)
3. ejecutarlo.
4. Lectura de la elección del usuario mediante las propiedades `CurrentFilter`, `DisplayDirectory` y `Files` (array) (`Files(0)` sólo contiene un valor).

```
Dim oFilePicker As Object, NombreArchivo As String
NombreArchivo = ""
'inicialización de FilePicker
oFilePicker = CreateUnoService("com.sun.star.ui.dialogs.FilePicker")
oFilePicker.DisplayDirectory = DirectorioInicio
oFilePicker.appendFilter("Hojas de cálculo", "*.ods")
oFilePicker.CurrentFilter = "Hojas de cálculo"
oFilePicker.Title = "Elige la hoja de cálculo"
'ejecución y verificación del código de retorno (OK)
If oFilePicker.execute = _
         com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
  NombreArchivo = oFilePicker.Files(0)
End If
```

Abrir varios archivos (FilePicker)

1. Como lo anterior,
2. initializarlo (en modo múltiple `MultiSelectionMode = True`),
3. ejecutar,
4. al retorno, el array `Files()` contiene la selección de archivos del usuario.

Guardar un archivo (FilePicker)

1. Crear un `FilePicker`,
2. inicializarlo (tipo `FILESAVE_XXX`), (y propiedades y métodos indicados),
3. ejecutar,
4. al retorno lectura de la elección de usuario en las propiedades `CurrentFilter`, `DisplayDirectory` y `Files` (array) (`Files(0)` sólo contiene un valor).

Elegir un directorio (FolderPicker)

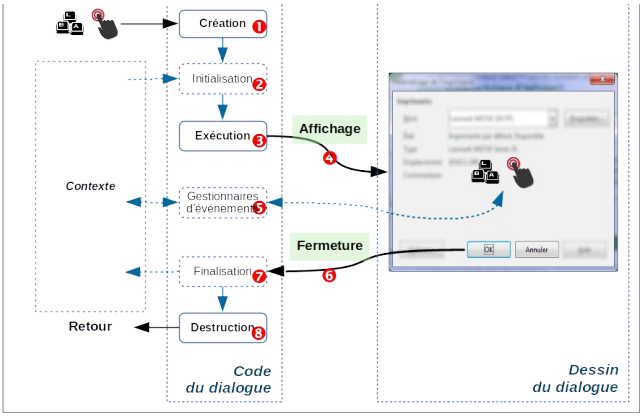
1. Crear un `FolderPicker`,
2. inicializarlo (propiedades y métodos indicados),
3. ejecutar,
4. al retorno, lectura de la elección del usuario en `Directory`.

```
Dim oFP As Object, NombreDir As String
NombreDir = ""
oFP = CreateUnoService("com.sun.star.ui.dialogs.FolderPicker")
oFP.DisplayDirectory = ConvertToURL("C:\Ruta\al\Directorio")
oFP.Description = "Cliquez sur un répertoire"
oFP.Title = "Elige el directorio de backup"
If oFP.execute = _
         com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
  NombreDir = oFP.Directory
End If
```

Diálogos personalizados - Principios

Diálogo Basic = un módulo **Diálogo** (diseño) + (al menos) un módulo de **código**

Secuencia de ejecución de un diálogo



La secuencia de ejecución se muestra en el gráfico:

1. En respuesta a un evento de la aplicación creamos el diálogo
2. (realizamos la **inicialización** de los controles desde el contexto de aplicación
3. **ejecutamos** el diálogo que recibe el flujo de la ejecución
   4. Se muestra en pantalla,
   5. (gestionamos de los **eventos** de los controles del diálogo),
   6. los eventos provocan **el cierre** del diálogo (**Aceptar, cancelar**);
7. (realizamos las operaciones de **finalización** hacia el contexto aplicativo si es preciso
8. el diálogo se **destruye** y **devuelve código** a la aplicación que hace la llamada

☞ Creación, inicialización, ejecución, finalización por nuestro código
Presentación, cierre: operaciones automáticas consecutivas a las precedentes

☠ Desarrollamos las respuestas a los **eventos** de los controles (Guía ref. n.º4).

Carga de las bibliotecas de diálogos

☞ Si usa mucho código o varios diálogos, piense en alojarlos en bibliotecas dedicadas (una por diálogo).

Las bibliotecas de diálogos no se cargan nunca automáticamente.

Los nombres de las bibliotecas a cargar son sensibles a las Mayúsculas !

Modal frente a No modal

Modal Un diálogo modal toma el control total del teclado, ratón y pantalla en espera de una acción del usuario, la aplicación permanece inaccesible

☞ predeterm. los diálogos son modales.

No modal Un diálogo no modal no bloquea el acceso a la aplicación.
Ej.: el diálogo **Buscar y reemplazar** de LibreOffice

Cuidado con las ejecuciones múltiples de diálogos no modales, ya que pueden bloquear la aplicación.

Diálogos personalizados comunes (Modales)

Tenemos un módulo de diálogo `MiDlg` y un módulo de código `MiCodigoDlg` en la biblioteca `MiLibDlg`. En una subrutina del módulo de código se crea una instancia de un objeto diálogo (`oDlg`) a partir del diálogo.

Creación/carga en memoria

```
DialogLibraries.loadLibrary("MilibDlg")
oLib = DialogLibraries.getByName("MiLibDlg")
oModulo = oLib.getByName("MiDlg")
oDlg = CreateUnoDialog(oModulo)
'y ahora se puede manipular oDlg
```

Sólo Llamada

`oDlg.execute` ☞ El flujo de ejecución se tranfiere al diálogo.

Llamada y comprobación del retono

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

☞ El flujo del ejecución se transfiere al diálogo y el valor de retorno comprobado (¿Ha pulsado el usuario el botón **Aceptar?**).

Finalización / destrucción del diálogo

```
oDlg.dispose
```

Ejemplo recapitulativo (módulo de código)

Este ejemplo no muestra la gestión de los eventos.

```
Sub MostrarDialogo()

  Dim oLib As Object, oModulo As Object, oDlg As Object

  DialogLibraries.loadLibrary("MiLibDlg")
  oLib = DialogLibraries.getByName("MiLibDlg")
  oModulo = oLib.getByName("MiDlg")
  oDlg = CreateUnoDialog(oModulo)
  'InicializaciarDlg() 'initializar el contenido del diálogo
  If oDlg.execute =
    com.sun.star.ui.dialogs.ExecutableDialogResults.OK Then
    'FinalizarDlg() 'Hacer algo con los datos obtenidos
  End If
oDlg.dispose
End Sub
```

Diálogos personalizados no modales

Tenemos un módulo de diálogo `MiDlgNM` y un módulo de código `MiCodigoDlgNM` en la biblioteca `MiLibDlgNM`. En una subrutina del módulo de código se crea una instancia de un objeto diálogo (`oDlg`) a partir del diálogo.
Aplicar los mismos principios que antes con las sustituciones:

1. **La presentación** del diálogo se asegura con `oDlg.SetVisible(True)` en lugar de `oDlg.execute`,
2. dos variables globales booleanas de control
   - `gCorriendo` (impide las ejecuciones múltiples).
   - `gMostrar` controla la presencia del diálogo en la pantalla
3. **las respuestas a los eventos** (controles) ponen `gMostrar a Falso cierre diálogo.`

Presentación del diálogo

`oDlg.SetVisible(True)` ☞ El diálogo se muestra.
El flujo de ejecución **NO** se transfiere al diálogo

Ejemplo recapitulativo (módulo de código)

```
'Control presentación del diálogo, declarado al inicio del módulo!
Dim gMostrar As Boolean
'control de ejecuciones múltiples no deseadas también al inicio
Dim gCorriendo As Boolean

Sub MostrarDialogoNoModal() 'gestiona creacción y present. del
    diálogo
  Dim oBibli As Object, oModule As Object, oDlg As Object

  'évitar ejecuciones múltiples
  If Not gCorriendo Then
    gCorriendo = True : gMostrar = True
    DialogLibraries.loadLibrary("MiLibDlgNM")
    oLib = DialogLibraries.getByName("MiLibDlgNM")
    oModulo = oLib.getByName("MiDlgNM")
    oDlg = CreateUnoDialog(oModulo)
    'IniializarDlg() 'inicializar el contenido del diálogo
    'mostrar el diálogo mientras que la vaiable sea True
    Do While gMostrar = True
      Wait 20       'permitir la ejecución de otros programas
      oDlg.SetVisible(True)  'mantenerlo en pantalla
    Loop

    'FinalizarDlg() 'hacer algo con los datos obtenidos si es preciso
    oDlg.dispose
    gEnCours = False
  End If
End Sub 'MostrarDialogoNoModal

Sub OnBtnOKClick(ByRef pEvt As Object)
'Réspuesta a un clic en Aceptar

  'realizar las acciones necesarias
  'Finalizar con :
  gMostrar = False '=> fin del bucle while
                          'por consiguiente cierre del diálogo
End Sub 'OnBtnOKClick
```

Asociar un evento a una macro

Nuestro diálogo se comunica con la aplicación a través de los **eventos** (❺ del esquema). Habrá que crear entonces las macros que respondan a las ocurrencias de estos eventos (extracto de Guía ref. n.° 4):

1. **Creamos la macro** a ejecutar siguiendo el modelo:
   ```
   Sub NombreDeLaMacro()
   End Sub
   ```

Consejo: El nombre de la macro se relaciona con el objeto, acción y el tipo de evento.
ejemplo: `Sub AlPulsarAceptar()`

La Subrutina puede incluir un parámetro Ver abajo «Obtener informaciones...»,

2. **seleccionamos el objeto** que va a interceptar el evento.
3. Accedemos a su configuración (método variable según el objeto),
4. **elegimos el evento** a interceptar,
5. **Asignamos la macro** a ejecutar en el desplegable del evento (punto 1.).

☞ Más informaciones sobre los eventos en el Guía ref. n.° 4.

Obtener informaciones sobre el evento desencadenante

La macro de procesamiento puede consultar el parámetro que recibe para obtener las informaciones complementarias sobre el evento:

```
Sub RespuestaEvento(ByRef Evento As Object)
End Sub
```

La estructura y las propiedades del objeto `Evento` dependerán del tipo de evento que desencadene la llamada al procedimiento

Información de los controles

Para acceder a	Consulta
(Objeto) **control** que hace la llamada	`Event.Source`
(Objeto) **modelo** del control	`Event.Source.Model`
(Objeto) **diálogo** contenedor del control	`Event.Source.Context`

Inicialización y finalización

Inicialización

(❷ esquema) El diálogo necesita a menudo informaciones procedentes del contexto de ejecución. La macro de inicialización configura el diálogo a partir de sus datos.

Finalización

(❼ esquema) Este proceso implica realizar la operación inversa a la anterior: actualizar los datos contextuales a partir de los ingresados o elegidos en el diálogo

Gestión de los módulos de diálogo

LibreOffice gestiona los módulos de diálogo independientemente del código (Guía de referencia. n.°1). Es posible copiar estos objetos de un documento a otro.

Copiar un módulo de diálogo de una biblioteca a otra

(en el mismo documento o entre documentos / contenedores)

1. En el IDE, abrimos los dos documentos/contenedores origen y destino.
2. Abrimos el **Organizador de macros** (botón),
3. pestaña **Diálogos**, arrastramos y soltamos desde el origen al destino.

☞ predeterm., los diálogos se mueven, para copiarlos pulsar **Ctrl** al desplazarlos.

Guardar una copia de un módulo de diálogo

1. En el IDE, abrimos el módulo Diálogo a guardar
2. Pulsamos el botón de la barra de herramientas **Diálogo de Exportación**
3. Ponemos nombre al archivo y lo guardamos

El documento se guarda en formato XML y con la extensión `.xdl`.

☞ La importación (acción inversa) de hace con el botón **Diálogo de importarción**

Creditos

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

Somos como enanos sentados sobre los hombros de gigantes. Si vemos más cosas y más lejanas que ellos, no es por la perspicacia de nuestra visión, ni por nuestra grandeza, sino porque son ellos los que nos elevan. (Bernard de Chartres [atribuido])

Historial

Versión	Fecha	Comentarios
1.01	01/10/2017	Primera versión.
1.05	02/12/2019	Pequeñas modificaciones.
	07/04/2021	Traducción al español y alguna modificación: B. Antonio Fernández

El documento original se puede obtener en la Wiki francesa de publicaciones de L.O.