

Automatización de tareas en OpenOffice

MACRO COMIDA

No hace falta ser un experto para iniciarse en la interfaz de programación de BASIC de OpenOffice. **POR DMITRI POPOV**

Dmitry Pechugin, Fotolia

OpenOffice viene con su propio lenguaje de programación basado en Basic. A pesar de que OOO Basic no es el lenguaje de programación más difícil, sí requiere tiempo y esfuerzo, especialmente si no se es programador. Aún así, ni la lectura de documentación ni tener que jugar con código deberían ser necesarios si sólo necesitamos automatizar tareas esporádicamente. En este artículo proporcionaré algunos consejos y fragmentos de código de manera que permitan un uso práctico de OOOBasic sin tener que aprender el lenguaje desde cero.

Iniciando Aplicaciones Externas

La capacidad de iniciar aplicaciones externas y pasarles los datos es una de las funcionalidades más útiles de OpenOffice.org. Usando el comando *Shell* podemos arrancar virtualmente cualquier aplicación instalada en nuestra máquina. El comando tiene el siguiente formato: *Shell (Ruta, Estilo-ventana, Parámetro)*, donde *Ruta* define la ruta al programa, *Estilo-ventana* la ventana en la que se inicia, mientras que *Parámetro* especifica el parámetro de la línea de comandos. Por ejemplo, el enunciado de *Shell* de abajo abre la URL <http://wordnet.princeton.edu/perl/webwn> en el navegador Firefox y lo trae hasta el primer plano:

```
Shell ("firefox", 1, "
http://wordnet.princeton.edu
/perl/webwn")
```

Si deseamos abrir otra URL en Firefox, deberemos cambiarla manualmente, lo cual no es práctico. Afortunadamente, OOO Basic ofrece un modo de *pillar* un texto seleccionado desde el documento abierto actual:

```
ThisDoc=ThisComponent
TextSelection=
ThisDoc.getCurrentController()
.getSelection().getByIndex(0)
.getString()
```

De esta manera, podemos seleccionar una URL en el texto y usarla con el comando *Shell*:

```
Shell ("firefox", 1,
TextSelection)
```

Otra forma de adquirir una URL es a través de un cuadro de entrada en el que el usuario puede introducirla. En este caso, el código que abre la URL en Firefox se asemeja a

```
InputText=InputBox(
("Campo de Entrada:",
"Titulo de Ventana")
Shell ("firefox", 1, InputText)
```

Incluso con este simple comando podemos crear macros útiles, como la que se puede ver en el Listado 1, la cual nos permite enviar mensajes a Twitter directamente desde OpenOffice.org.

Para enviar mensajes, la macro usa la utilidad *curl*, que deberá encontrarse instalada en nuestra máquina. Esta utilidad usa el

comando siguiente para enviar mensajes a Twitter:

```
curl -u nombreusuario:
contraseña -d status="Tu
tweet va aquí."
http://twitter.com/statuses/
update.xml
```

De este modo la macro tiene que solicitar al usuario que introduzca un mensaje (lo que se llama un *tweet* en la jerga Twitter), el cual se pasa luego como argumento al comando *curl* con el uso de una instrucción *Input-Text = Input-Box("Tu mensaje:", "Enviar a Twitter")*.

Las dos líneas siguientes de la macro merecen un análisis más profundo. El problema es que los espacios en el mensaje deben convertirse a formato URL, en caso contrario, cada palabra se envía en el mensaje como un tweet separado.

Para convertir el mensaje en URL deben sustituirse todos los espacios por la cadena *%20*, así *"Tu tweet va aquí"* se convierte en *Tu%20tweet%20va%20aquí*. Las rutinas de la cadena *Split* y *Join* hacen exactamente esto. La rutina *Split* "corta" la cadena en segmentos de texto utilizando como referencia los espacios y un separador, mientras que *Join* los "pega" usando la cadena *%20*.

Listado 1: Macro PostToTwitter

```
01 Sub PostToTwitter()
02 InputText=InputBox("
mensaje:", "Enviar a
Twitter")
03 SplitStr = Split(InputText, "
")
04 Tweet = Join(SplitStr, "%20")
05 TwMessage=" -u
nombreusuario:contraseña -d
status=" & "" & Tweet & "" &
" http://twitter.com/
statuses/update.xml"
06 Shell("curl", 1, TwMessage)
07 End Sub
```

A continuación, la macro debe construir el argumento del comando *curl*, lo cual se hace concatenando los parámetros de la línea de comandos requeridos y el tweet (sustituimos *nombreusuario:contraseña* por nuestro nombre de usuario y contraseña Twitter reales):

```
TwMessage=" -u nombreusuario:contraseña -d status=" & "" & Tweet & "" & "http://twitter.com/statuses/update.xml"
```

Por último, la macro usa el comando *Shell* para iniciar la utilidad *curl* y enviar el tweet. Como puede apreciarse, estos pocos comandos simples son suficientes para crear macros y hacer buen uso de ellas.

Trabajar con Documentos

La macro que aparece en el Listado 2 puede parecerse un poco complicada, pero introduce unas cuantas técnicas que nos permiten obtener el nombre del documento abierto en un momento dado y su ruta, comprobar su estado y guardarlo en un lugar específico. Puede dividirse en distintos pasos: Primero define la variable *FileProperties(0)*, que más tarde se usará para especificar las propiedades de los ficheros para una copia de seguridad del documento actual. El comando *ThisDoc = ThisComponent* ordena a la macro que use el documento actual, y el código siguiente carga la librería *Tools*:

```
If (Not GlobalScope.BasicLibraries.isLibraryLoaded("Tools")) Then GlobalScope.BasicLibraries.LoadLibrary("Tools") End If
```

La librería *Tools* contiene rutinas que le permiten a la macro obtener el directorio del documento actual. Pero antes de hacerlo debe comprobar si el documento ha sido guardado (es decir, asegurándose de que de hecho tiene una localización). Si no, la macro solicitará al usuario que lo guarde y después saldrá:

```
If ThisDoc.hasLocation=False Then MsgBox ("¡Has de guardar este documento primero!", , "¡Atención!") :End End If
```

Cuando guarda una copia del documento, añade la fecha actual y la hora al nombre del fichero para hacer que sea más fácil encontrar la versión del documento que desea. Usando las rutinas *CDateToISO* y *Format* consigue la fecha actual en formato ISO (es decir, AAAA-MM-DD) y la hora actual en HH-MM-SS:

```
DateToday=CDateToISO(Date) & "_" & Format(Hour(Now), "00") & "-" & Format(Minute(Now), "00") & "-" & Format(Second(Now), "00")
```

La macro obtiene la URL del documento y usa la rutina *DirectoryNameoutofPath* para obtener el directorio:

```
DocURL=ThisDoc.getURL() DocDir=DirectoryNameoutofPath(DocURL, GetPathSeparator())
```

La rutina *Dir* usa luego la URL para extraer el nombre del documento:

```
FileName=Dir(DocURL, 0)
```

La macro construye la ruta para la copia de seguridad del fichero, que consta de la ruta al directorio en el que se almacena el documento original y el nombre del fichero, que incluye el sello de la hora y la fecha creadas:

```
SaveFile=DocDir & GetPathSeparator() & FileName & "_" & DateToday
```

También guarda una copia de seguridad del documento en un lugar específico usando la propiedad del fichero *Overwrite*, la cual sobrescribe cualquier fichero con el mismo nombre:

```
FileProperties(0).Name="Overwrite" FileProperties(0).Value=True ThisDoc.storeToURL(SaveFile, FileProperties())
```

La macro puede modificarse para guardar una copia de seguridad del documento en un servidor FTP. Se añade un comando que solicita al usuario que introduzca una dirección FTP (por ejemplo, *ftp://usuario:contraseña@192.168.1.7/seguridad/*):

```
FTPServerPath=InputBox("Introducir ruta FTP", "Dirección FTP")
```

Luego se modifica el enunciado *SaveFile = DocDir & GetPathSeparator() & FileName & "_" & DateToday* de manera que sea algo como:

```
SaveFile=FTPServerPath & FileName & "_" & DateToday
```

Listado 2: Macro BackupDocument

```
01 Sub BackupDocument()
02 Dim FileProperties(0) As New com.sun.star.beans.PropertyValue
03 ThisDoc=ThisComponent
04
05 If (Not GlobalScope.BasicLibraries.isLibraryLoaded("Tools")) Then
06 GlobalScope.BasicLibraries.LoadLibrary("Tools")
07 End If
08
09 If ThisDoc.hasLocation=False Then
10 MsgBox ("¡Hay que guardar el documento primero!", , "¡Atención!") :End
11 End If
12
13 DateToday=CDateToISO(Date) & "_" & Format(Hour(Now), "00") & "-" & Format(Minute(Now), "00") & "-" & Format(Second(Now), "00")
14
15 DocURL=ThisDoc.getURL()
16 DocDir=DirectoryNameoutofPath(DocURL, GetPathSeparator())
17 FileName=Dir(DocURL, 0)
18 SaveFile=DocDir & GetPathSeparator() & FileName & "_" & DateToday
19 FileProperties(0).Name="Overwrite"
20 FileProperties(0).Value=True
21 ThisDoc.storeToURL(SaveFile, FileProperties())
22 End Sub
```

Entonces podemos personalizar esta macro según nuestras necesidades de acuerdo a nuestro entorno.

Diálogos

La rutina *InputDialog* nos permite presentar cuadros de entrada simples, aunque OOo Basic también nos permite crear cuadros de diálogo que contengan múltiples campos de entrada, listas desplegables, botones y otros widgets ofrecidos por cualquier GUI que se precie. Como ejemplo de cómo crear un cuadro de diálogo, consideremos la macro del Listado 3, la cual crea una herramienta que convierte la Temperatura de grados Fahrenheit a Celsius.

La macro presenta un cuadro de diálogo que consta de cuatro elementos: un cuadro de entrada, en el que los usuarios introducen el valor que deseen; un cuadro de lista desplegable, que contiene direcciones de conversión (por ejemplo, "Celsius - > Fahrenheit" y "Fahrenheit - > Celsius"); otro campo de entrada, que muestra el resultado de la conversión; y un botón que muestra la conversión y cierra el diálogo.

Antes de comenzar a codificar la macro debemos crear el diálogo. Para ello, seleccionamos *Herramientas | Macros | Organizar diálogos*, y pulsamos el botón *Nuevo*. Le damos al diálogo un nombre descriptivo (por ejemplo, *SimpleConverterDialog*) y pulsamos el botón *Aceptar*. Pulsamos en el botón *Editar*, y con las herramientas disponibles en la barra *Cuadro de Herramientas* añadimos los campos de resultados y

las entradas (deben ser campos numéricos), el cuadro de la lista desplegable y un botón. Para definir las propiedades para cada elemento usamos la ventana *Propiedades*. Por ejemplo, tenemos que configurar el tipo de botón a *Aceptar*. Para hacerlo seleccionamos el botón y elegimos *Aceptar* de la lista desplegable *Tipo de Botón* en la ventana *Propiedades*. También tenemos que añadir las entradas "Celsius - > Fahrenheit" y "Fahrenheit - > Celsius" al cuadro de la lista desplegable. Esto lo podemos hacer seleccionando el campo cuadro de la lista desplegable y añadiendo las entradas en el campo *Entradas de la Lista* de la ventana *Propiedades*. Con el campo *Nombre*, en la misma ventana *Propiedades*, podemos dar a a cada elemento un nombre para hacer más fácil su identificación. Por ejemplo, nombraremos al campo de entrada como *InputField* y al resultante *ResultField*.

Una vez que el diálogo esté listo, podemos comenzar a programar la macro. Ésta inicializa en primer lugar el diálogo con las siguientes líneas:

```
exitOK=com.sun.star.ui.dialogs.
ExecutableDialogResults.OK
Library=DialogLibraries.
GetByName("GUI")
TheDialog=Library.GetByName(
("SimpleConverterDialog"))
```

Este código supone que el diálogo *SimpleConverterDialog* se almacena en la librería de

la GUI. A continuación, la macro tiene que inicializar los campos de diálogo:

```
DialogField1=
Dialog.getControl("InputField")
DialogField2=
Dialog.getControl("ListBox")
DialogField2.SelectItemPos(0,
True)
DialogField3=
Dialog.getControl("ResultField")
```

ejecutando entonces el diálogo mediante el uso del comando *Dialog.execute*, y asignando el valor de la variable *InputField* a la *InputValue*:

```
InputValue=DialogField1.value
```

El comando *Select Case* redirige la macro a la fórmula de conversión adecuada dependiendo de la entrada que el usuario seleccionó en el cuadro de la lista desplegable. Por ejemplo, si el usuario ha seleccionado "Fahrenheit - > Celsius", la macro ejecuta lo siguiente:

```
ConvertedValue=(InputValue-
32)*5/9
```

El resultado de la conversión se inserta luego en el campo resultado:

```
DialogField3=Dialog.getControl(
("ResultField")).setValue(
ConvertedValue)
```

Listado 3: Macro TemperatureConverter

01 Sub TemperatureConverter()	10 DialogField3=	(“CommandButton”)
02	Dialog.getControl(“ResultField	22 Button.Label = “Cerrar”
03 exitOK=	”))	23 Dialog.execute()
com.sun.star.ui.dialogs.Execut	11	24
ableDialogResults.OK	12 Dialog.execute()	25 Case “Celsius -> Fahrenheit”
04 Library=	13	26 ConvertedValue=
DialogLibraries.GetByName	14 InputValue=DialogField1.value	InputValue*9/5+32
(“GUI”)	15	27 DialogField3=Dialog.
05 TheDialog=Library.GetByName	16 Select Case	getControl(“ResultField”).setV
(“SimpleConverterDialog”)	DialogField2.SelectedItem	alue(ConvertedValue)
06	17	28 Button=Dialog.getControl
07 DialogField1=	18 Case “Fahrenheit -> Celsius”	(“CommandButton”)
Dialog.getControl	19 ConvertedValue=	29 Button.Label = “Cerrar”
(“InputField”)	(InputValue-32)*5/9	30 Dialog.execute()
08 DialogField2=	20 DialogField3=Dialog.	31
Dialog.getControl(“ListBox”)	getControl(“ResultField”).	32 End Select
09 DialogField2.SelectItemPos(0,	setValue(ConvertedValue)	33 End Sub
True)	21 Button=Dialog.getControl	

Finalmente, la macro cambia la etiqueta del botón a *Close*, y cuando el usuario lo pulsa, se cierra el diálogo y se detiene la macro:

```
Button=Dialog.getControl
("CommandButton")
Button.Label = "Close"
Dialog.execute()
```

La macro convertidora puede modificarse fácilmente para que lleve a cabo otros tipos de conversiones. Lo único que debemos hacer es especificar nuevas entradas en el cuadro de la lista desplegable y añadir un nuevo bloque de código *Case* a la macro con la fórmula de conversión apropiada.

OOo Basic contiene herramientas que nos permiten conectar a una base de datos y manipular los datos contenidos en ésta. La macro del Listado 4 demuestra cómo establecer una conexión a una base de datos local y cómo crear una consulta y presentar los resultados en una ventana de diálogo. *ShowWordlist* es una macro bastante sencilla que conecta a una base de datos llamada *TinyDB*. Busca todos los registros en la columna *Words* en la tabla *Worldlist* y puebla un cuadro de lista desplegable en el diá-

logo *Worldlist* con los resultados. Antes de poder establecer conexión con la base de datos se registra como fuente de datos en OpenOffice. Para hacerlo, seleccionamos *Herramientas | Opciones y luego OpenOffice.org Base | Bases de Datos*, y pulsamos el botón *Nuevo*. Seleccionamos la base de datos *TinyDB* y damos a la nueva conexión el nombre de *TinyDB*. Pulsamos *Aceptar | Aceptar*, y ya está. Establecer una conexión a la base de datos usando OOOBasic sólo requiere tres líneas de código:

```
DBContext=createUnoService
("com.sun.star.sdb.
DatabaseContext")
DataSource=
DBContext.getByname("TinyDB")
Database=
DataSource.getConnection ("", "")
```

Como OpenOffice.org Base utiliza el lenguaje SQL para manipular bases de datos, crear una consulta que recupere todos los registros de la tabla *worldlist* es sólo cuestión de usar el comando *SELECT FROM SQL*. Para recuperar los registros desde la base de datos usando la consulta SQL espe-

cificada, la macro usa el servicio *RowSet*, que primero debe iniciarse:

```
SQLResult=createUnoService
("com.sun.star.sdb.RowSet")
SQLQuery="SELECT ""Words""
FROM ""wordlist"""
```

Ejecuta entonces la consulta SQL usando el siguiente código:

```
SQLResult.activeConnection=
Database
SQLResult.Command=SQLQuery
SQLResult.execute
```

Ya sabemos cómo iniciar un cuadro de diálogo, así que la única parte que necesita una atención detallada es el siguiente bloque de código:

```
While SQLResult.next
ListBoxItem=SQLResult.
getString(1)
DialogField.additem
(ListBoxItem, DialogField.
ItemCount)
Wend
```

Para poblar el cuadro de la lista desplegable la macro usa el bucle *While ... Wend*, el cual recoge un registro (*SQLResult.next*), extrae la cadena de la primera columna: (*ListBoxItem = SQLResult.getString(1)*), y la inserta como un elemento del cuadro de la lista: (*DialogField.additem(ListBoxItem, DialogField.ItemCount)*). Una vez ha terminado, cierra la conexión a la base de datos:

```
Database.close
Database.dispose()
```

OpenOffice.org Extension Repository contiene extensiones útiles, muchas de las cuales se editan bajo licencias de código abierto, así que podemos usar el código en nuestras propias creaciones. Por ejemplo, encontraremos algunas de las técnicas y códigos descritos en este artículo en la extensión *Writer's Tools* [1], creada por servidor. ¡Feliz programación!

Listado 4: Macro ShowWordlist

```
01 Sub ShowWordlist()
02
03 DBContext=createUnoService
("com.sun.star.sdb.DatabaseCon
text")
04 DataSource=
DBContext.getByname("TinyDB")
05 Database=
DataSource.getConnection
("", "")
06
07 SQLResult=createUnoService
("com.sun.star.sdb.RowSet")
08 SQLQuery="SELECT ""Words""
FROM ""wordlist""
09 SQLResult.activeConnection=
Database
10 SQLResult.Command=SQLQuery
11 SQLResult.execute
12
13 exitOK=
com.sun.star.ui.dialogs.Execut
ableDialogResults.OK
14 Library=
DialogLibraries.GetByName
("Wordlist")
15 TheDialog=Library.GetByName
("WordlistDialog")
16
17 DialogField=Dialog.getControl
("ListBox")
18
19 While SQLResult.next
20 ListBoxItem=
SQLResult.getString(1)
21 DialogField.additem
(ListBoxItem,
DialogField.ItemCount)
22 Wend
23
24 If Dialog.Execute=exitOK Then
25 CurrentItemName=
DialogField.SelectedItem
26 End If
27
28 Database.close
29 Database.dispose()
30 End Sub
```

RECURSOS

- [1] *Writer's Tools*: [writertools.googlecode.com](http://www.googlecode.com)
- [2] Listados de este artículo: <http://www.linux-magazine.es/Magazine/Downloads/47/>