Managing reminders and microblogs in OpenOffice

CREATURE COMFORTS

If you spend a lot of time in OpenOffice, you can use it to remind you of deadlines or to update your microblog. **BY DMITRI POPOV**

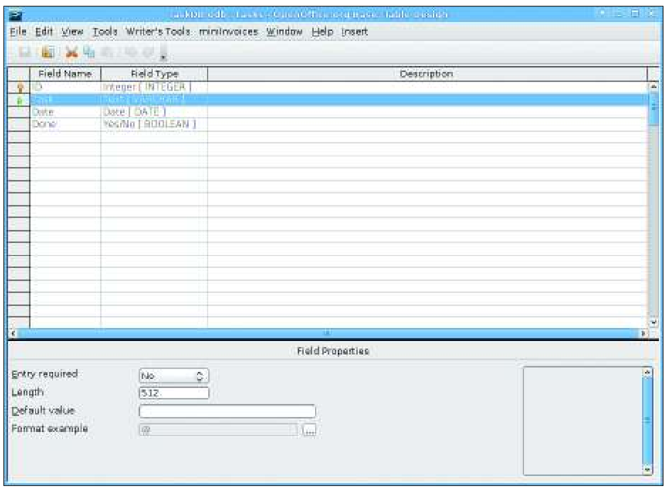
Although OpenOffice Basic is not the most powerful and flexible programming language out there, you can still create some nifty solutions with it. For example, with a simple macro and a database, you can add a reminder feature to help you keep tabs on your deadlines. How about a macro that lets you update your Identi.ca or Twitter status directly from within OpenOffice? If this sound good, then read on.

Adding a Simple Reminder Feature

As a busy professional, you might already have a calendaring solution that helps you keep tabs on your tasks. However, if you spend most of your time in OpenOffice, you might want to add a simple reminder feature that alerts you to upcoming events and overdue tasks every time you launch the productivity suite. To do this, you need two things: a simple database for storing tasks and a macro that pulls the data from the database and displays it in a window.

To start, create a new OpenOffice Base database called TaskDB. When you create the database, make sure that the *Yes, register the database for me* option is selected. Once the database is created, switch to the *Tables* section and create a new table in the design view. Now add at least three fields: *ID* (*INTEGER* primary key), *Task* (*VARCHAR* to store task descriptions), *Date* (*DATE* to store deadlines), and *Done* (*BOOLEAN* to mark tasks as done). Then save the table under the *tasks* name (Figure 1).

Once the database is in place, you can start working on the macro. In OpenOffice, choose *Tools | Macros | Organize Macros | OpenOffice Basic*, press the *Organizer* button, and switch to the *Dialogs* section. Now select an existing library (*Standard* is a good choice), press *New*, give the dialog a name (e.g., "Dialog1"), and press the *Edit* button to open the created dialog for editing. To add a list box to the dialog window, use the List box control.


File Edit View Tools Writer's Tools miniInvoices Window Help Insert

Field Name	Field Type	Description
ID	Integer [INTEGER]	
Task	Text [VARCHAR]	
Date	Date [DATE]	
Done	Yes/No [BOOLEAN]	

Field Properties

Entry required	No
Length	512
Default value	
Format example	@

Figure 1: Tasks table.

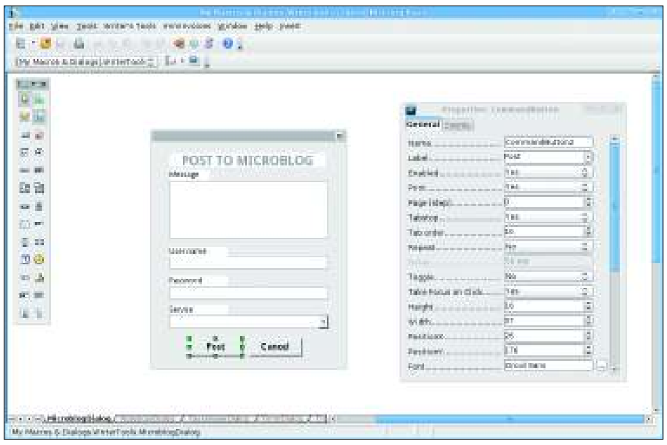


Figure 2: Creating the microblogging dialog.

Now switch to the BASIC module and enter the macro in Listing 1. First, the macro establishes a connection to the TaskDB database (lines 3–5). The macro then initiates the *Dialog1* dialog window and the *ListBox1* field (lines 7–12). Next, the macro pulls data from the tasks table with the SQL query (line 13).

This query selects the *Task* and *Date* (*SELECT ""Task"", ""Date""*) columns and obtains records that are not marked as *Done* (*WHERE ""Done""* = *'No'*). For each record, the query compares the value of the *Date* field with the current date. If the value is higher than the current date, the *Status* field remains blank, but if the value is lower than the current date, then the *Status* field is set to *OVERDUE*. Finally, the query sorts the data by date in ascending order (*ORDER BY ""Date"" ASC*). Next, the macro executes the SQL query and populates the list box with the fetched data (lines 14–19).

To run the macro every time you launch OpenOffice, choose *Tools* | *Customize*, click on the *Events* tab, and select *OpenOffice* from the *Save in* drop-down list. Select the *Open Document* event, press the *Macro* button, and select the *ShowTasks* macro. Press *OK* to save the settings and close the window.

That's all there is to it. Now you can view the list of upcoming events and overdue tasks every time you launch OpenOffice.

Microblogging with OpenOffice

Microblogging services like Twitter and its open source alternative, Identi.ca, are all the rage these days, and a slew of high-quality clients are available for both services. However, leaving the convenience of OpenOffice every time you want to update your status can become an annoyance. A simple microblogging tool within OpenOffice provides a solution to this problem.

Both Identi.ca and Twitter statuses can be updated with the use of a specific API (Application Programming Interface) call. Although you can't do this directly from OpenOffice Basic, cURL [1], which comes with almost every Linux distribution, is a perfect tool for the job. cURL

Listing 1: *ShowTasks* Macro

```
01 Sub ShowTasks()
02 Dim RowSetObj, SQLStatement As Object
03 DBContext=createUnoService("com.sun.star.sdb.
   DatabaseContext")
04 DataSource=DBContext.getByName("TaskDB")
05 ConnectToDB=DataSource.GetConnection ("","")
06
07 exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
08 DialogLibraries.LoadLibrary("Standard")
09 Library=DialogLibraries.GetByName("Standard")
10 TheDialog=Library.GetByName("Dialog1")
11 ShowTasksDlg=CreateUnoDialog(TheDialog)
12 DialogField1=ShowTasksDlg.GetControl("ListBox1")
13 SQLQuery= "SELECT ""Task"", ""Date"", CASEWHEN( ""Date"" >
   CURRENT_DATE, '', 'OVERDUE' ) AS ""Status"" FROM ""tasks""
   WHERE ""Done""='No' ORDER BY ""Date"" ASC"
14 SQLStatement=ConnectToDB.createStatement
15 RowSetObj=SQLStatement.executeQuery (SQLQuery)
16   While RowSetObj.next
17     ListBox1Item=RowSetObj.getString(1) & " [" & RowSetObj.
       getString(2) & "]" & " " & RowSetObj.getString(3)
18     DialogField1.additem(ListBox1Item, DialogField1.
       ItemCount)
19   Wend
20 ShowTasksDlg.Execute()
21 End Sub
```

Listing 2: *UpdateStatus* Macro

```
01 Sub UpdateStatus()
02 Dim Username, Password as String
03 ServiceURL="http://identi.ca/api/statuses/update.xml"
04 exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
05 DialogLibraries.LoadLibrary("Standard")
06 Library=DialogLibraries.GetByName("Standard")
07 TheDialog=Library.GetByName("Dialog1")
08 Dlg=CreateUnoDialog(TheDialog)
09   If Dlg.Execute=exitOK Then
10     DialogField1=Dlg.getControl("TextField1")
11     MessageTxt=DialogField1.Text
12     StatusMsg=Join((Split((Join((Split((Join((Split((Join(
       (Split((Join((Split((Join(Split(MessageTxt, " "), "%20
       ")), "'")), "%27")), "@")), "%40")), "+")), "%2B")),
       """")), "%22")), "&")), "%26")
13     DialogField2=Dlg.GetControl("TextField2")
14     Username=DialogField2.Text
15     DialogField3=Dlg.GetControl("TextField3")
16     Password=DialogField3.Text
17     StatusUpdate=" -u " + Username + ":" & Password + " -d
       status=" + "" & StatusMsg + "" + " " + ServiceURL
18     Dlg.dispose
19     Shell("curl",1, StatusUpdate)
20   Else :End
21   End If
22 End Sub
```

can submit status messages to Identi.ca or Twitter with the following command:

```
curl -u username:password -d status=⮐
"Status message goes here" http: ⮐
//identi.ca/api/statuses/update.xml
```

Using cURL, you can create an OpenOffice Basic macro to obtain the required data from the user. Then you can construct a command string and pass it to cURL with the *Shell* routine. First you have to create a dialog containing three text boxes for entering a status message, username, and password (Figure 2). Also, you need a button that triggers the rest of the macro. When adding the button to the dialog, make sure to set its type to *OK* in the Properties panel. Now use an existing module or create a new one and enter the macro in Listing 2.

Similar to the previous example, the macro starts by initializing and displaying the dialog. Once the user has entered the data and pressed the *OK* button, the macro obtains the content of the dialog fields. The interesting part here is the *StatusMsg* statement (line 12).

This statement creates a URL-encoded version of the status message so that cURL can submit any message containing special characters, such as &, +, @, and spaces. The statement uses a series of *Split* and *Join* routines to replace these characters with their URL-encoded equivalents. To better understand how this works, I'll look at how the *Split* and *Join* routines are used to replace spaces with the %20 string. First, the *Split* routine chops the string into an array using the space as a delimiter:

```
SplitStr= Split(MessageTxt, " ")
```

For example, the *Split* command above would turn the "Weather is good today" string into the following *SplitStr* array:

```
Weather
is
good
today
```

The *Join* routine does the exact opposite of *Split*: It glues pieces in the array into a string with the use of a specified delimiter. So, the *StatusMsg = Join(SplitStr, "%20")* statement turns the *SplitStr* array into the following string: "Weather%20is%20good%20today".

Similarly, the statement converts other special characters into their URL-encoded versions. The macro then puts all the pieces together and constructs the *StatusUpdate* string that is then passed to cURL by the *Shell* routine (line 19).

Your microblogging tool is ready to go, but you can still improve a few things. For example, you can tweak the macro so that it saves your status messages and dates in a OpenOffice Base database. This effectively turns your microblogging macro into a simple backup tool. Start by creating a simple database called MicroblogDB that contains a table with three fields: *ID* (*INTEGER*, primary key), *Status* (*VARCHAR* to store status messages), and *Date* (*DATE* to store the current date). The *Date* field should use the YYYY-MM-DD format (the ISO format). Now save the table under the file name *microblog*. Next, modify the macro (Listing 3) so that it now consists of three additional steps.

First, it converts the current date value into the YYYY-MM-DD format (line 20). Next, the macro establishes a connection to the registered MicroblogDB database (line 23). Finally, it constructs an *INSERT* SQL query that inserts the status message and the formatted date in the database (line 24). ■

THE AUTHOR

Dmitri Popov holds a degree in Russian language and computer linguistics. He has been writing exclusively about Linux and open source software for several years, and his articles have appeared in Danish, British, North American, German, and Russian magazines and websites.

INFO

[1] cURL tool: *http://curl.haxx.se/*

Listing 3: Improved Version of the *UpdateStatus* Macro

```
01 Sub UpdateStatus()
02 Dim Username, Password as String
03 Dim SQLStatement As Object
04 ServiceURL="http://identi.ca/api/statuses/update.xml"
05 exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
06 DialogLibraries.LoadLibrary("Standard")
07 Library=DialogLibraries.GetByName("Standard")
08 TheDialog=Library.GetByName("Dialog1")
09 Dlg=CreateUnoDialog(TheDialog)
10   If Dlg.Execute=exitOK Then
11     CurrentItemPos=DialogField.SelectedItemPos.
12     DialogField1=Dlg.getControl("TextField1")
13     MessageTxt=DialogField1.Text
14     StatusMsg=Join((Split((Join((Split((Join((Split((Join(
       (Split((Join((Split((Join(Split(MessageTxt, " "), "%20
       ")), "'")), "%27")), "@")), "%40")), "+")), "%2B")),
       """")), "%22")), "&")), "%26")
15     DialogField2=Dlg.GetControl("TextField2")
16     Username=DialogField2.Text
17     DialogField3=Dlg.GetControl("TextField3")
18     Password=DialogField3.Text
19     StatusUpdate=" -u " + Username + ":" & Password + " -d
       status=" + "" & StatusMsg + "" + " " + ServiceURL
20     DateToday=Format(Year(Now), "0000") & "-" &
       Format(Month(Now), "00") & "-"  & Format(Day(Now),"00")
21     DBContext=createUnoService("com.sun.star.sdb.
       DatabaseContext")
22     DataSource=DBContext.getByName("MicroblogDB")
23     Database=DataSource.GetConnection ("","")
24     SQLQuery="INSERT INTO ""microblog"" " + "(""Status"",
       ""Date"") VALUES " + "('" + StatusMsg + "','" +
       DateToday + "')"
25     SQLStatement=Database.createStatement
26     Result=SQLStatement.executeQuery (SQLQuery)
27     Database.close
28     Database.dispose()
29   Else :End
30   End If
31 Dlg.dispose
32 Shell("curl",1, StatusUpdate)
33 End Sub
```