Linux.com

Everything Linux and Open Source

OOo Basic crash course: Creating a simple application launcher

May 28, 2008 (4:00:00 PM) - 5 months, 1 week ago

By: **Dmitri Popov**

In previous installments of the crash course, you've learned how to build a **simple basket tool**, a **task manager**, and even a **word game**. This time, let's take a look at how you can use the skills you picked up from those exercises to create a simple application launcher, which will allow you to start virtually any application without leaving the convenience of OpenOffice.org. While working on this project, you'll learn how to create and use functions, handle errors, and how to populate list boxes using records from a database table.

The first order of business is to create a simple OpenOffice.org Base database with a table called applaunch containing three fields: ID (primary key), AppName to store applications' names, and AppPath to store paths to applications' binaries (e.g. C:\Program Files\Mozilla Firefox\firefox.exe on Windows, or simply Firefox on Linux). Since the application launcher pulls the data from the created database, the first thing the macro has to do is to create a connection to it. By now, you can probably do this with your eyes closed, but this time let's try something different. As any programming language worth its salt, OpenOffice.org Basic allows you to create functions that can save you a lot of typing and make your programs more efficient and easy to manage. A function is basically a mini program that usually performs just one action. For example, when called, the following function displays a "Hello world!" message box:

```
Function SayHello
MsgBox "Hello world!"
End Function
```

You can call this function in your macro by using its name:

```
Sub Main
SayHello
End Sub
```

In the example above, the "Hello world!" message is hard-wired into the function, but you can also use arguments, i.e. variables that are passed to the main program. For example, the SayHello function below uses the Message argument, and the main program assigns the "Hello world!" value to it:

```
Sub Main
SayHello("Hello world!")
End Sub


Function SayHello (Message)
MsgBox Message
End Function
```

In a similar manner, you can create a function that connects to a database, so you don't have to write the same code every time you need to establish a database connection. This function should look something like this:

```
Function ConnectToDatabase(DBName As String) As Object
DBContext=createUnoService("com.sun.star.sdb.DatabaseContext")
DataSource=DBContext.getByName(DBName)
ConnectToDatabase=DataSource.GetConnection ("","")
End Function
```

In previous examples, we used the If ... End If condition to display an error message if the macro fails to establish a connection:

```
If not DBContext.hasByName("BaseDB") then
   MsgBox ("Connection failed!" , "Error!") : End
End If
```

While this simple trick does the job, OpenOffice.org Basic offers a more powerful and flexible mechanism for catching and managing errors. Using the On Error Go To routine, you can instruct the macro to perform certain actions when an error occurs. In the following example, the execution is redirected to the code block labeled ErrorHandler, which displays an error message and halts the function:

```
Function ConnectToDatabase(DBName As String) As Object
DBContext=createUnoService("com.sun.star.sdb.DatabaseContext")


On Error GoTo ErrHandler


DataSource=DBContext.getByName(DBName)
ConnectToDatabase=DataSource.GetConnection ("","")


Exit Function
    ErrHandler:
    MsgBox("Database connection failed!", ,"Error"):End
End Function
```

Now you have a function that establishes a database connection and handles potential connection errors. Calling the created function from the main program is easy. The only thing you need to do is to specify the name of the database (in this case, it's BaseDB) you want to connect to:

```
Database=ConnectToDatabase("BaseDB")
```

Using an SQL query, you can then retrieve all records from the applaunch database:

```
SQLQuery="SELECT ""AppName"" FROM ""applaunch"""
SQLResult.activeConnection = Database
SQLResult.Command = SQLQuery
SQLResult.execute
```

To display a list of applications in the applaunch table, the macro uses the simple AppLauncherDialog dialog consisting of the ListBox1 list box and the Launcher button (you have to create the dialog beforehand using OpenOffice.org IDE):

```
exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
OpenDialog("AppLauncherDialog")
Dialog=CreateUnoDialog(TheDialog)
DialogField=Dialog.GetControl("ListBox1")
```

To populate the list box in the dialog with the names of the applications, the macro uses a While ... Wend loop, which retrieves the value of the first column in the database table as a string using the SQLResult.getString(1) statement, and then adds it to the dialog field ListBox1. The loop exits when the macro reaches the last record in the database:

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

Next step is to make the macro launch the application selected by the user. To do this, the macro assigns the value of the selected list box item to the CurrentItemName variable:

```
If Dialog.Execute=exitOK Then
CurrentItemName=DialogField.SelectedItem
End If
```

It then uses the value of the CurrentItemName variable to retrieve the application's path via the following SQL query:

```
SQLQuery="SELECT ""AppPath"" FROM ""applaunch"" WHERE ""AppName""=" & "'" & CurrentItemName & "'"
SQLResult=Database.createStatement()
QueryResult=SQLResult.executeQuery(SQLQuery)
```

The retrieved path is then assigned to the LaunchApp variable, which is used in the Shell() routine to launch the application:

```
QueryResult.next
LaunchApp = QueryResult.getString(1)
Shell(LaunchApp,1, "")
```

That's all there is to it. To round things up, here is the full listing of the application launcher macro:

```
Sub LaunchApp()


Dim LaunchApp As String
Database=ConnectToDatabase("BaseDB")

SQLResult=createUnoService("com.sun.star.sdb.RowSet")
SQLQuery="SELECT ""AppName"" FROM ""applaunch"""
SQLResult.activeConnection = Database
SQLResult.Command = SQLQuery
SQLResult.execute

exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
OpenDialog("AppLauncherDialog")
Dialog=CreateUnoDialog(TheDialog)

DialogField=Dialog.GetControl("ListBox1")

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

If Dialog.Execute=exitOK Then
CurrentItemName=DialogField.SelectedItem
End If

SQLQuery="SELECT ""AppPath"" FROM ""applaunch"" WHERE ""AppName""=" & "'" & CurrentItemName & "'"

SQLResult=Database.createStatement()
QueryResult=SQLResult.executeQuery(SQLQuery)
QueryResult.next
LaunchApp = QueryResult.getString(1)

Shell(LaunchApp,1, "")

Database.close
Database.dispose()

End Sub
```

With a little bit of tweaking, you can put the created macro to other uses. For example, if you are learning a foreign language, you can turn the created macro into a tool that can help you to look up verb forms. Here is what the modified version of the macro looks like (note that it uses the same function you've created earlier):

```
Sub Conjugate()
```

```
Dim ThisDialog as Object

Database=ConnectToDatabase("BaseDB")

SQLResult=createUnoService("com.sun.star.sdb.RowSet")
SQLQuery="SELECT ""Infinitive"" FROM ""verbs"""
SQLResult.activeConnection = Database
SQLResult.Command = SQLQuery
SQLResult.execute

exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
OpenDialog("VerbsDialog")
Dialog=CreateUnoDialog(TheDialog)

DialogField=Dialog.GetControl("ListBox1")

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

If Dialog.Execute=exitOK Then
CurrentItemName=DialogField.SelectedItem
Else
 End
End If

SQLQuery="SELECT ""Infinitive"", ""Definition"", ""Present"", ""SimplePast"", ""PastPerfect"" FROM ""verbs"" WHERE ""Infinitive""=" & "'" & CurrentItemName &"'"


SQLResult=Database.createStatement()
QueryResult=SQLResult.executeQuery(SQLQuery)

QueryResult.next
InfinitiveResult=QueryResult.getString(1) & "(" & QueryResult.getString(2) & ")"
PresentResult=QueryResult.getString(3)
SimplePastResult=QueryResult.getString(4)
PastPerfectResult=QueryResult.getString(5)

exitOK=com.sun.star.ui.dialogs.ExecutableDialogResults.OK
OpenDialog("ConjugationDialog")
Dialog=CreateUnoDialog(TheDialog)

DialogField1=Dialog.getControl("TextField1").setText(InfinitiveResult)
DialogField1=Dialog.getControl("TextField2").setText(PresentResult)
DialogField2=Dialog.getControl("TextField3").setText(SimplePastResult)
DialogField3=Dialog.getControl("TextField4").setText(PastPerfectResult)

Dialog.execute()

Database.close
Database.dispose()

End Sub
```

Read in the original layout at: **http://www.linux.com/feature/135700**