BaseTools Extension

The goal of this extension is to provide a few methods of automating/simplifying certain tasks when working with Base forms (with Basic. Most of the tools have been grouped into the **RunCmd** module. Other tools have been included in the Forms, Databases and Utils Modules. The Query, Reports and Table modules are currently empty.

This tool is its early (very early) stages. I decided to post it now in case anyone wants to review it and provide feedback.

Using Base Tools

First, the BaseTools Basic library must be loaded (which is the entire extension).

```
BasicLibraries.LoadLibrary("BaseTools")
```

You can test to make sure the library is installed.

```
If BasicLibraries.hasByName("BaseTools") Then
    BasicLibraries.LoadLibrary("BaseTools")
End If
```

If the BaseLoader routine is bound to the Open Document OOo event, the library will be loaded automatically, otherwise load it before use (such as the *When Loading* event of a Base form).

> **NOTE:** There is an annoying bug related to the frame listener added by the BaseLoader subroutine. This causes the Basic IDE to open when the Base window is closed. I am still working on this issue.

Tools in the RunCmd Module

Functions in this module, as in any other module can be access directly, or via the module name and dot notation:

```
RunCmd.MoveToRecord bcNext
MoveToRecord bcNext
```

MoveToRecord

Moves the record pointer in a form to the specified record. The first parameter indicates the target record, which is a constant (see Constants module). The second parameter indicates a record number/index, if the target record constant **bcGoto** is used. The third parameter (optional) is the name of a form (the form document) or leave blank to use the current form.

```
RunCmd.MoveToRecord bcNext
```

SaveForm

Tries to save the data in the form as well as commit the database document to disk. The only parameter (optional) is a form name. Current form used if vomited

```
RunCmd.SaveForm
```

ApplyFilter

Applies the filter string passed as first parameter to the current form, or name form on second parameter.

```
RunCmd.ApplyFilter "FirstName='Charles'"
RunCmd.ApplyFilter "FirstName LIKE '*har*'"
```

ClearFilter

Clears the filter

```
RunCmd.ClearFilter
```

openFormDocument

Opens a form document. The first parameter is the form document name. The second parameter is an optional event object which makes finding a connection easier. The third parameter (optional) is a filter string to apply, and the fourth is the index of the form to filter. The final parameter, also optional, indicates a specific record to move to.

```
RunCmd.openFormDocument("Employees")
```

RunSQL

Executes the SQL string passed. Returns result set for select statements

```
RunCmd.RunSQL("UPDATE Employees Set Terminated=1")
```

PrintForm

Prints the current form. [needs work]. Only does a screen print of the visible area. Need to work on proper size conversions.

```
RunCmd.PrintForm
```

OpenQuery

Opens the named query.

```
RunCmd.OpenQuery("qryEmployees")
```

OpenTable

Opens the named table.

```
RunCmd.OpenTable(“Employees”)
```

OpenSQL

Similar to query, but can pass SQL select satement.

```
RunCmd.OpenSQL(“SELECT FirstName From Employees”)
```

ShowToolbar | HideToolbar |

Show/hide a toolbar. See list of available toolbar names in the constants module.

```
RunCmd.ShowToolbar bcFormStatusBar
RunCmd.ShowToolbar bcFormDesign
RunCmd.HideToolbar bcFormControls
```

SetFocus

Sets the focus to the named control

```
RunCmd.SetFocus “txtFirstName”
```

FindRecord

Searches for a record with matching criteria. This is fairly crude at the moment. Hope to improve in the future. The first parameter is the search criterion string. The second indicates where in the target field to search. Constant in: bcMatchStart, bcMatchEnd, bcMatchAnyWhere,bcMatchAll—see constants module. The third parameter indicates if seach is case sensitive (boolean). The forth parameter indicates which columns to target: use constant bcAllFields, or index of column. The 5th parameter indicates record to start search: bcFirstRecord, bcCurrentRecord, bcLastRecord. The 6th column indiates what direction to search : bcForward, bcBackward. And the final parameter indicates a form. All but the first parameter are optional.

```
RunCmd.FindRecord("dd",bcMatchAnyWhere,False,2,bcCurrentRecord)
RunCmd.FindRecord("pdf",bcMatchEnd)
RunCmd.FindRecord("dd",bcMatchStart)
RunCmd.FindRecord("Z:\ddf.pdf",bcMatchAll)
RunCmd.FindRecord("\dd",bcMatchAnyWhere,True,2,bcLastRecord,_
bcBackWard,"")
```

FindNext

The FindRecord method saves the all parameters when run. The FindNext method uses those parameters to find the next match.

```
RunCmd.FindNext
```

FindFirst and FindLast

These two functions are similar to the FindNext function (they actually just call the FindNext function). They find the first and last occurrence. These two functions DO NOT take the starting record and direction parameters. The FindFirst function starts at the first record and moves forward; and the FindLast function starts at the last and moves backward.

ShowControl and HideControl

Show and hide form controls. The first parameter is the control's name, and the second (optional) is the form name. The current form is used if parameter omited.

```
RunCmd.ShowControl("txtFirstName")
RunCmd.HideControl("txtFirstName")
```

The Forms Module

The Forms module has some features that help in working with forms/form controls. For example, if forms (form documents) *Orders* and *Clients* are opened. You can use the getForm(...) method to get a data form from either of these form documents. However, this is all contingent on one of two things:

1. the Sub BaseLoader has been set to the Open Document Event in the OOo main settings. Tracking what forms are open/active requires registering various event listeners. Which are done by this routine.
2. Otherwise you can call the subroutine InitDatabaseDocument and pass the database document model as parameter. This has to be done at an early stage, such as when a main menu type form loads.

Me and MeModel

These two functions return the active data form model. If either of the two previously mentioned functions are called, the event listeners will help in determining what form is active. Otherwise it is best to pass an event to these functions.

```
form=Me(Event)
form=MeModel(Event)
form=Me
form=MeModel
```

```
form.next()
```

MeController

This function returns the controller for the current form model. Also dependent on event listeners as Me/MeModel

InitFromEvent

From a given event (passed as parameter) sets the active form model/controller.

getFormController

For a given form model, get the controller.

IsDocumentLoaded

Returns true/false if form document is/is not loaded. Depends on event listeners. Parameter is the for name.

findDocument

Returns the form document model if loaded. Parameter is the for name.

getForm

Returns the form model for given name. If only the name of the form document is passed the top form, if any, is returned. To specify a specific form, include full path using dot notation.

```
emps=Forms.getForm("Employees") 'for doc
emps=Form.getForm("Employees.Standard") 'form doc with top form
emps=Form.getForm("Employees.Employees") 'ditto
empsContact=Form.getform("Employees.Employees.Contact") 'sub
```

getFormDocument

Get document model for a given data form model

getControl

Gets a control (view controller) for a given control model

Databases Module

For some time I keep reading that the runtime variables ThisDatabaseDocument, ThisConnection, and ThisDataSource are available (as well as storing macros in Base docs and assigning events). However I cannot seem to find them in any of my installations.

So, I have created a function that get the current db doc, connection and data source service—in case it is not just me, and they are not yet available. These functions also depend on the event listeners set by the BaseLoaded routine. Otherwise you can use the InitFromEvent routine to initialize from a form/form control event.

You can pass an event object to any of these functions to make sure it succeeds—faster also. Otherwise it will go putzing around trying to find the correct objects.

getDatabase

This function takes a database name (name of file without extension) as parameter and searches the desktop for open databases with that name. If found, returns the db file model, else null.

Well, that wraps her all up. I will keep working as time permits, and hopefully easy some of the pain of programming with Base forms. There are various other functions in the modules, but are auxiliary functions, so I will not cover them in this brief description.