ScriptForge

Scripting resources for Basic [ & Python ] coders

Jean-Pierre Ledure

https://gitlab.com/LibreOfficiant/scriptforge

2020
Agenda

- “ScriptForge”, what is it, in a nutshell?
- Is it smart to code an API in Basic?
- Service orientation
- Extensibility, the framework
- Modules in release 1.0
- Perspectives
ScriptForge, what is it?

A service-oriented framework

- **Data** containers
- **Arrays**
- **Strings**
- A mapping class
- **Error** handling
- **Files** and folders, **text files** read & write
- **Context** information
- **Localization**
- Interconnection of **Basic** and **Python**
- Windows and **documents**
  - **Calc** sheets
- **Dialogs** and their controls
- **Databases**
- ... **MORE TO COME** ...

Have MORE people daring
MORE ambitious automation
An API for Basic written mainly in Basic.

Smart?

- Complex data structures?
  - Variants
  - User defined types
  - Arrays
  - Objects
  - All mixed

- Object orientation?
  - Encapsulation, polymorphism
  - Class instances, attributes, methods
  - No inheritance/subtyping (can be bypassed)
  - Class instance creation inside its library
  - Basic variable types are not objects

- Namespaces for variables?
  - Global, Local
  - Public, Private attributes ignored
  - Qualification to avoid homonyms
An API for Basic written mainly in Basic. Smart?

- Namespaces for Functions/Subs?
  - Private is ignored
  - Any Function present in a loaded library becomes callable from all libraries
  - A library cannot be unloaded
  - Only full qualification definitely prevents collisions

```
GlobalScope.Library.Module.Function()
```

- BUT ...

```
Dim f As Object, g As Object
Set f = GlobalScope.myLibrary.myModule
f.myFunction(...)
Set g = f
g.myFunction(...)
```
Service orientation of ScriptForge
ScriptForge service: the user’s point of view

Reserved words: (qualification is not forbidden ... :)

- CreateScriptService
- SF_Array, SF_Exception, SF_String (because of their presumed frequent use)
ScriptForge service: the user’s point of view

CreateScriptService() returns either

1) A Basic object referring to a Basic module

2) A Basic object referring to a Basic class instance
   Arguments may be passed to the constructor of the instance

RULE OF THUMB: Qualify ALL methodnames
Service – Behind the scenes

User script

Set myServ = CreateScriptService("myLibrary.myService1")

ScriptForge core

Function CreateScriptService(...) As Object
    If [not yet done] Then
        GlobalScope.BasicLibraries.loadLibrary("myLibrary")
        oModule = [FindModule]("RegisterScriptServices", "myLibrary")
        oModule.RegisterScriptServices()
    End If
    CreateScriptService = [GetService]("myLibrary.myService1")

myLibrary

Sub RegisterScriptServices()
    ScriptForge.SF_Services.RegisterService(“myService1”, _
        GlobalScope.myLibrary.Mod_Service1)” passes a module
    ScriptForge.SF_Services.RegisterService(“myService2”, _
        “myLibrary.aModuleName.NewService2”)’ passes as a string the function to invoke
    ’ to get an instance of the service
Service – The framework

ScriptForge core implements
- CreateScriptService()
- RegisterService()
- RegisterEventManager()

Each associated/invited Library implements
- RegisterScriptServices()

Each module implements
- ServiceName
- Properties()
- Methods()
- _Repr()

Each class module implements
- GetProperty()
- SetProperty()

User Basic script
CreateScriptService()

User Python script
CreateScriptService()
The framework

The CORE library
- Data containers
- Localization
- Exception handling
- User interface
- Technical context
- File management

Associated libraries
- Documents
- Dialogs
- Databases

Guest libraries

Extensions
The individual modules

Release 1.0
ScriptForge – Release 1.0

Services in *ScriptForge* library
- Array
- Dictionary
- Exception
- FileSystem
- L10N
- Platform
- Session
- String
- TextStream
- Timer
- UI

Services in associated libraries
- **SFDocuments**
  - Document
  - Calc
  - Base
- **SFDialogs**
  - Dialog
  - DialogControl
- **SFDataBases**
  - Database

Built with LO 7.1
- 4 Basic libraries + Python helper functions
- 1 Help page by service
- 1 (english) POT file

Additionally
- A complete unit-tests suite
- A coding conventions charter
**SF Databases.Database service**

The **Database service** provides for the access to databases either embedded or described in Base documents. Each instance of the current class represents a single database, with essentially its tables, queries and data (not its forms or reports).

The exchanges with the database are done in **SQL only**.

To make SQL statements more readable, use optionally square brackets to surround table/query/field names instead of the (RDBMS-dependent) normal surrounding character, (usually double-quote, single-quote or other).

SQL statements may be run in **direct** or **indirect mode**. In direct mode the statement is transferred literally without syntax checking nor review to the database engine.

The provided interfaces include simple tables, queries and fields lists, and access to database data.

**Service invocation and usage**

1. To access any database at anytime

```vba
Dim myDatabase As Object
Set myDatabase = CreateScriptService("SF Databases.Database", [FileName], [RegistrationName], [ReadOnly], [User], [Password]"
' Arguments:
' FileName: the name of the Base file compliant with the SF FileSystem.Filename notations
' RegistrationName: the name of a registered database (mutually exclusive with FileName)
' ReadOnly: Default = True
' User, Password: additional connection arguments to the database server
' ... Run queries, SQL statements, ...
myDatabase.CloseDatabase()
```

2. To access the database related to the current Base document
The “Array” service

30 methods to

- **Add data**
  - Append, AppendRow, AppendColumn

- **Transform**
  - Flatten, Transpose, Reverse

- **Sort**
  - Sort, SortRows, SortColumns, InsertSorted

- **Search**
  - Contains, IndexOf

- **Operate sets**
  - Unique, Intersection, Union, Difference

- **Import/Export**
  - ImportFromCSVFile, ExportToTextFile, Join2D, ConvertToDictionary

Inspired by Python lists and PHP arrays
The “Dictionary” service

**Benefit: enhance Collections**
Any item type + keys do not need to be known in advance

```vbnet
Dim myDict
myDict = CreateScriptService(“Dictionary”)```

- **Update data**
  - Add, ReplaceKey, ReplaceItem, Remove, RemoveAll

- **Search**
  - Exists, Item, Items, Keys

- **Import/Export**
  - ConvertToJson, ImportFromJson, ConvertToPropertyValues, ImportFromPropertyValues, ConvertToArray

As an example, the management of services is implemented thru a dictionary of dictionaries.

Inspired by the VBA Dictionary class
The “Exception” service (1)

- Basic runtime error in ScriptForge itself
  
  `RaiseAbort()`

- Error detected by ScriptForge
  
  `RaiseFatal()`
The “Exception” service (2)

Similar to the VBA Err object

- Basic runtime error in a user script
- Raise(), Clear()

```
Sub Example.Raise()
Dim a, b, c
On Local Error GoTo Catch
Try:
    a = 10 : b = 0
    c = a / b ' Error #11
'...
Exit Sub
Catch:
    Variants =>>
End Sub
```

- Error detected by a user script
- Raise(), RaiseWarning()

```vba
' Standard behaviour
Catch:
    SF_Exception.Raise()

' Ignore the error
Catch:
    If SF_Exception.Number = 11 Then SF_Exception.Clear()

' Simulate another error
Catch:
    SF_Exception.Raise(12)

' Replace the usual message
Catch:
    SF_Exception.Raise(, , "It is not a good idea to divide by zero")
```
The “Exception” service (3)

- Debugging and logging
  - `DebugPrint()`

- Console management
  - `Console(Modal)`, `ConsoleClear()`, `ConsoleToFile()`
The “FileSystem” service

8 Properties
- `FileNaming = “ANY | URL | SYS”`
- `ConfigFolder, ExtensionsFolder, InstallFolder`

25 methods to
- **Add**
  - `CreateFolder, CreateTextFile`
- **Copy/Move/Delete** (with wildcards)
  - `CopyFolder, CopyFile, MoveFolder, MoveFile, DeleteFolder, DeleteFile`
- **Explore**
  - `FolderExists, FileExists, SubFolders, Files`
- **Interact**
  - `PickFolder, PickFile`
- **Access to the “TextStream” service**
  - `CreateTextFile, OpenTextFile`
- **Examine**
  - `HashFile, CompareFiles`

Inspired by the VBA FileSystemObject class
The “FileSystem” service – the TextStream class

Dim FSO As Object, oFile As Object
Set FSO = CreateScriptService("FileSystem")
Set oFile = FSO.OpenTextFile("C:\Temp\ThisFile.txt", IOMode := FSO.ForReading)

6 Properties

- Encoding, NewLine
- AtEndOfStream

Methods to

- **Read**
  - ReadLine, ReadAll, SkipLine
- **Write**
  - WriteLine, WriteBlankLines
- **Close**
  - CloseFile

Inspired by the VBA Textstream class
The “L10N” service

PO-Files dissociate the two very different profiles involved in the process, i.e. the programmer and the translator(s)

Methods

- for the programmer to **build a set of words or sentences** in a reference language
  
  *AddText*

- To export all the above texts into a pristine POT-file
  
  *ExportToPOTFile*
  
  The generated file should pass successfully the "msgfmt --check" GNU command.

- To **get** at runtime the text in the **user language**
  
  *GetText*

Inspired by GNU’s “gentle art of editing PO files”

ScriptForge is shipped with its own POT file
The “Platform” service

- Architecture
- ComputerName
- CPUCount
- CurrentUser
- Machine
- OfficeVersion
- OSName
- OSPlatform
- OSRelease
- OSVersion
- Processor

Inspired and executed by the Python platform.py standard library
The “Session” service

- **UNO objects introspection**
  `UnoObjectType`, `HasUnoProperty`, `HasUnoMethod`, `UnoProperties`, `UnoMethods`

- **Send file(s)**
  `SendMail`

- **Execute external programs**
  `ExecuteBasicScript`, `ExecutePythonScript`, `ExecuteCalcFunction`
  `RunApplication`

- **Web**
  `WebService`, `OpenUrlInBrowser`
The “String” service

42 advanced methods to

- Replace substrings
  \textit{ReplaceStr, ReplaceChar}

- Validate input
  \textit{IsAlpha, IsAlphanum, IsADate, IsEmail, IsFileName, IsHexDigit, IsIPV4, IsUrl, IsWhitespace, IsLike, IsRegex}

- Parse strings
  \textit{FindRegex, ReplaceRegex, Count, SplitNotQuoted}

- Handle quotes, linebreaks and special characters
  \textit{Quote, Unquote, Escape, Unescape, FilterNotPrintable, HtmlEncode, ExpandTabs, Wrap}

- Compare strings
  \textit{HashStr}

Inspired by Python and PHP string functions
The “Timer” service

A timer measures elapsed time (in milliseconds)

```vbnet
Dim oTimer As Object
Set oTimer = CreateScriptService("Timer", True)
' True => Starts immediately
```

Properties
- IsStarted, IsSuspended, Duration, TotalDuration, SuspendDuration

Methods
- Start, Terminate, Suspend, Continue, Restart
The “UI” service

Cfr. StarDesktop

**Properties**
- ActiveWindow, Documents

**Methods**
- **To manage windows**
  - Activate, MaximizeWindow, MinimizeWindow, Resize, WindowExists
- **To show progress bars**
  - ShowProgress, SetStatusBar
- **To get a Document instance**
The “SFDocuments.Document” service

Generic functions for all types of documents

Properties

- `DocumentType, IsBase, IsCalc, …`
- `DocumentProperties, CustomProperties => Dictionary`
  `XComponent` (shortcut to UNO)

Methods

- **To save**
  `Save, SaveAs, SaveCopyAs`

- **To run a command**
  `RunCommand`

- **Other**
  `Activate, CloseDocument`
The “SFDocuments.Document” service, the Calc subtype

The same methods and properties as “Document” +

- Smart range concept

```vba
oDocB.CopyToRange(oDocA.Range("SheetX.D4:F8"), "D2:F6")
```

Properties
- CurrentSelection, LastCell, LastRow, LastColumn
- XSpreadsheet, XCellRange (shortcuts to UNO)

Methods, a.o.
- Sheet management
  InsertSheet, CopySheet, CopySheetFromFile …
- Data exchange with Basic
  CopyToCell, CopyToRange, GetValue
- Import data
  ImportFromCSVFile, ImportFromDatabase
- Modify data
  SetArray, SetValue, SortRange
The “SFDatabase.Database” service

Easy access to database data via SQL

**Properties**
- *Tables, Queries*

**Methods**
- **Selective data**
  - Dlookup, Dmax, Dmin, Dsum, DCount
- **Massive data**
  - GetRows
- **Any SQL**
  - RunSql
- **Shortcuts to UNO**
  - XConnection, XMetadata
The “SFDialogs.Dialog” service

Run dialogs designed with the Basic IDE

Properties
- Caption, Height, Visible, Page, Modal
- XDialogModel, XDialogView => Shortcuts to UNO

Methods
- Activate, Execute(modal As Boolean), EndExecute, Terminate

```vba
Dim oDlg As Object, lButton As Long
Dim Container As String, Library As String, DialogName As String
Set oDlg = CreateScriptService("SFDialogs.Dialog", Container, Library, DialogName)
    '    ... Initialize controls ...
    lButton = oDlg.Execute() ' Default mode = Modal
    If lButton = oDlg.OKBUTTON Then
        '    ... Process controls and do what is needed
    End If
oDlg.Terminate()

Controls([ControlName])
```
The “SFDialogs.DialogControl” service

Get / set control properties

Property names are identical (but not necessarily applicable) whatever the control type

E.g. “Value”

Properties

- **Read-only**
  - ControlType, ListCount, Parent, Text

- **Updatable**
  - Cancel, Caption, Default, Enabled, Format, ListIndex, Locked, MultiSelect, Page, Picture, RowSource, TipText, TripleState, Value, Visible

- **UNO shortcuts**
  - XControlModel, XControlView

Methods

- SetFocus, WriteLine
Perspectives

- Data structures
  - Tree
- Dialog controls
  - Treeview, Grid
- Documents
  - Writer
  - Progress
- Forms
  - Form controls
- Charts
- Toolbars

- Services for Python scripts only
- Services common to Python & Basic
Thank you …

- … for your attention!
- … for supporting LibreOffice!
- … for soon supporting ScriptForge!