LibreOffice
The Document Foundation

LibreOffice
Conference 2021

ScriptForge

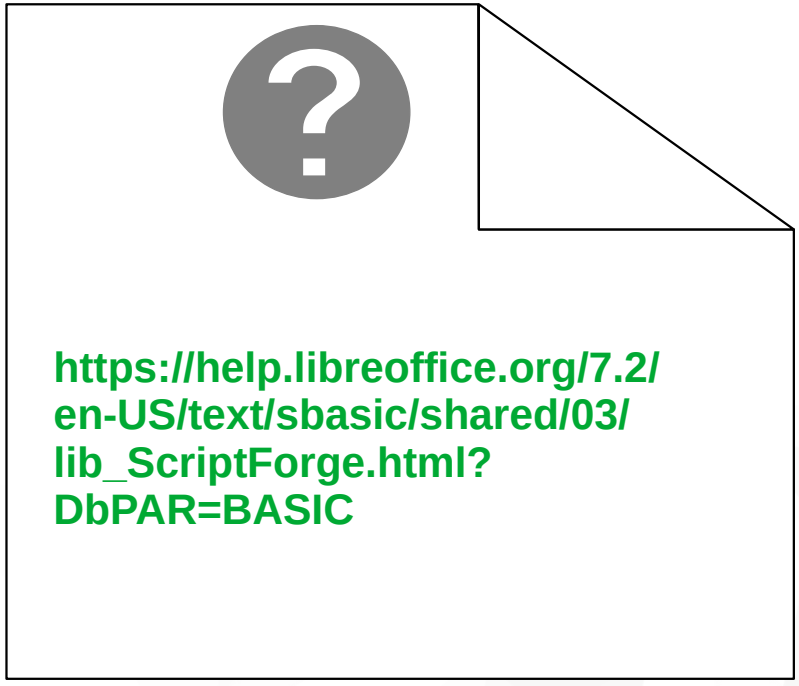
Scripting resources for Basic & Python coders

Jean-Pierre Ledure

List of services (7.2)

- **ScriptForge libraries** build up an extensible collection of macro scripting resources for LibreOffice to be invoked from **Basic macros** or **Python scripts**.
- Those libraries were **integrated in LO 7.1** and **upgraded in 7.2**.

Category	Services	
LibreOffice Basic	Array Dictionary	Exception String
Documents content	Base Calc	Database Document
User interface	Dialog DialogControl UI	Form FormControl
Utilities	Basic FileSystem L10N Platform	Services Session TextStream Timer



https://help.libreoffice.org/7.2/en-US/text/sbasic/shared/03/lib_ScriptForge.html?DbPAR=BASIC

ScriptForge – Release 7.2

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

- **Services in associated libraries**
  - ***SFDocuments***
    - Document*
    - Calc*
    - **Base**
    - **Form**
    - **FormControl**
  - ***SFDialogs***
    - Dialog*
    - DialogControl**
  - ***SFDatabases***
    - Database

(*) Upgraded in 7.2

Built within LO 7.2

- 4 Basic libraries
- 1 Python module: helper functions
- 1 Python module for Python scripts
- 23 Help pages
- 1 (english) POT file + PT translation

Additionally

- A complete unit-tests suite
- A coding conventions charter

Not walls, bridges ...

Targets of the project

- Propose the same comprehensive set of services to Basic and Python users
  (except when the language offers them natively)
- Unify the user interfaces for exceptions, user interactions, debugging
- Hide UNO complexity
  Give fast access to UNO
- Apply best practices from similar API's
- Propose well-written inspiring examples

Implement in Basic or Python ?

Coding 2X not being an option, either

- Code in Basic + Interfaces in Python or …
- Code in Python + Interfaces in Basic ?

	Basic	Python	
Object orientation	⚠	☑	No inheritance / subtyping Class instance creation inside its library
GUI	☑	☒	
Dynamic call	☑	☑	ScriptProvider *CallByName()* in Basic *hasattr(), getattr()* in Python
Persistent memory	☑	☒	*Global* variables
Namespaces	☒	☑	...

Implement in Basic or Python ?

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(...)
```

Implement in Basic or Python ?

Coding 2X not being an option, either

- **Code in Basic + Interfaces in Python** or …
- Code in Python + Interfaces in Basic ?

	Basic	Python	
Object orientation	⚠	☑	No inheritance / subtyping Class instance creation inside its library
GUI	☑	☒	
Dynamic call	☑	☑	ScriptProvider *CallByName()* in Basic *hasattr(), getattr()* in Python
Persistent memory	☑	☒	*Global* variables
Namespaces	☑	☑	

Service orientation of ScriptForge

ScriptForge service invocation

```
GlobalScope.BasicLibraries.loadLibrary("ScriptForge")
Dim f, t
    f = CreateScriptService("FileSystem")
    f.CopyFile( … )
    f.DeleteFolder( … )
    t = f.TemporaryFolder
```

Basic code

CreateScriptService() returns either

1) a Basic/Python object referring to a Basic (singleton) **module**

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

```
from scriptforge import CreateScriptService
f = CreateScriptService("FileSystem")
f.CopyFile( … )
f.DeleteFolder( … )
t = f.TemporaryFolder
```

Python code

Service – Behind the scenes

User script

```
Set s = CreateScriptService("SFLib.Service1")
```

ScriptForge core

```
Function CreateScriptService(...) As Object
    If [not yet done] Then
        GlobalScope.BasicLibraries.loadLibrary("SFLib")
        Set oModule = [FindModule]("RegisterScriptServices", "SFLib")
        oModule.RegisterScriptServices()
    End If
    CreateScriptService = [GetService]("SFLib.Service1")
```

SFLib

```
Sub RegisterScriptServices()
    ScriptForge.SF_Services.RegisterService("Service1", _
        "SFLib.aModuleName.NewService1")
            '   passes as a string the function to invoke
            '   to get an instance of the service
    ScriptForge.SF_Services.RegisterService("Service2", _
        GlobalScope.SFLib.Mod_Service2)    '   passes a module
Function NewService1() As Object
    NewService1 = New Service1
```

Service1

```
Option ClassModule
```

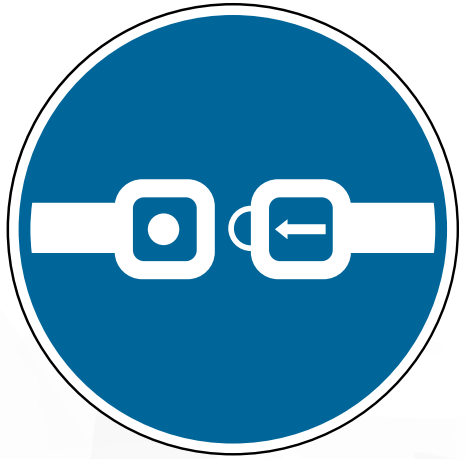
Very low differentiation between Basic and Python in ScriptForge

ScriptProvider.getScript().invoke()

Inter-language dynamic call

Incompatible data types between Python and Basic

- 2D arrays
- Dates
- Native objects
- Null, Empty, Nothing, Missing vs. None
- Arrays containing items with above types



Call applicable on module-level functions only

A specific protocol is required

Architecture – Identical interfaces

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Database(SFServices):
  serviceimplementation = ‘basic’
  serviceproperties = dict(Queries = False, …)
  def GetRows(self, sqlcommand, directsql = False):
    return self.ExecMethod(self.vbMethod, \
           'GetRows', sqlcommand, directsql)

def CreateScriptService(service, *args):
```

```
from scriptforge import CreateScriptService
a = CreateScriptService('database', '', 'Bibliography')
b = a.getrows('SELECT something FROM anything')
```

Python code

Interface definition

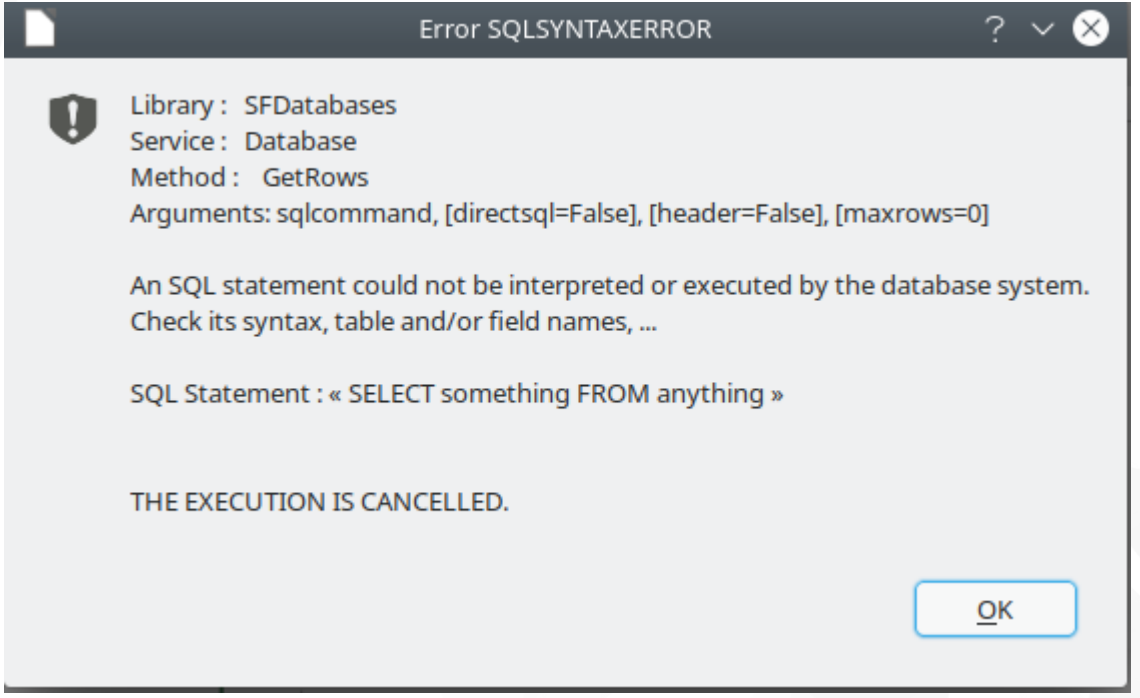
- Property names : lowercase, ProperCase or camelCase
- Method names : lowercase, ProperCase or camelCase
- Arguments : lowercase only

Architecture – Common error handling

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Database(SFServices):
  serviceimplementation = ‘basic’
  serviceproperties = dict(Queries = False, …)
  def GetRows(self, sqlcommand, directsql = False):
    return self.ExecMethod(self.vbMethod, \
           'GetRows', sqlcommand, directsql)

def CreateScriptService(service, *args):
```

```
from scriptforge import CreateScriptService
a = CreateScriptService('database', ‘’, 'Bibliography')
b = a.getrows('SELECT something FROM anything')
```

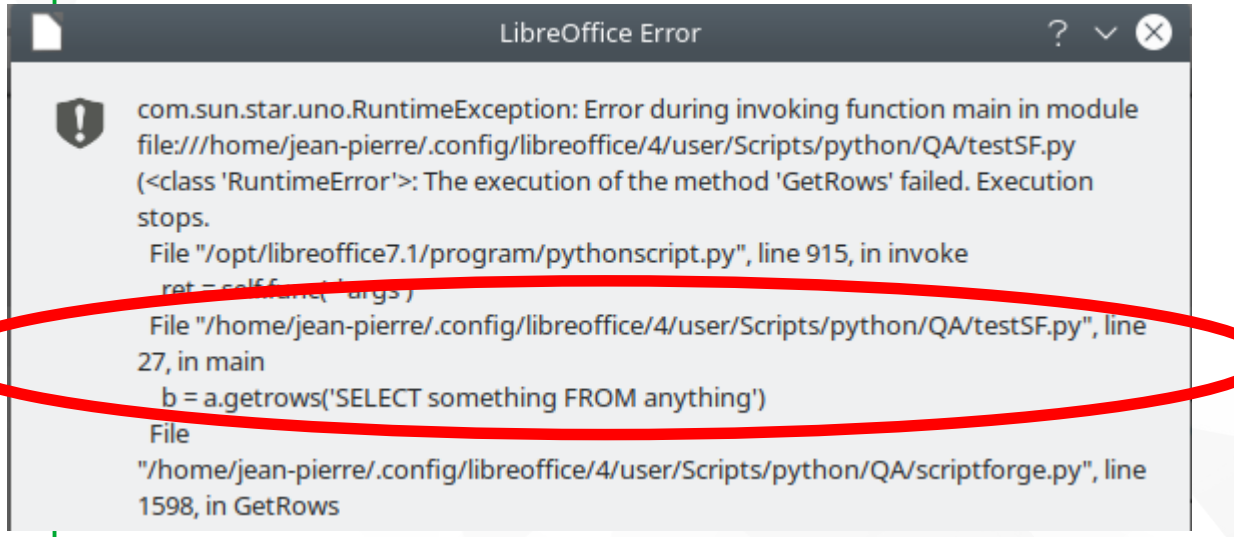


Architecture – Python error handling

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Database(SFServices):
  serviceimplementation = ‘basic’
  serviceproperties = dict(Queries = False, …)
  def GetRows(self, sqlcommand, directsql = False)
    return self.ExecMethod(self.vbMethod, \
           'GetRows', sqlcommand, directsql)

def CreateScriptService(service, *args):
```

```
from scriptforge import CreateScriptService
a = CreateScriptService('database', ‘’, 'Bibliography')
b = a.getrows('SELECT something FROM anything')
```



Architecture – Python calls dispatcher

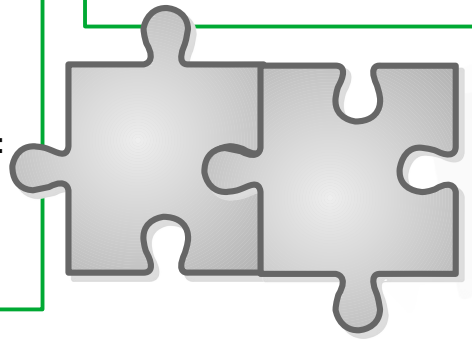
```
from scriptforge import CreateScriptService
a = CreateScriptService('database', '', 'Bibliography')
b = a.getrows('SELECT something FROM anything')
```

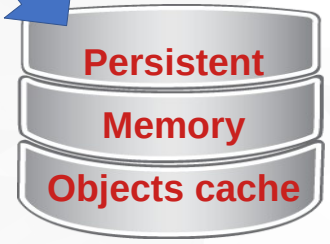
scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Database(SFServices):
  serviceimplementation = 'basic'
  serviceproperties = dict(Queries = False, …)
  def GetRows(self, sqlcommand, directsql = False):
    return self.ExecMethod(self.vbMethod, \
           'GetRows', sqlcommand, directsql)

def CreateScriptService(service, *args):
```

SF_PythonHelper.xba

```
Function _PythonDispatcher(…)
  ' Process input arguments
  ' Execute Basic routine
      vReturn = CallByName(object, "GetRows", ...)
  ' Process returned value
  '    (dates, arrays, ..)
  _PythonDispatcher = Array(vReturn, ..
End Function
```





Architecture – Common debugging

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Exception(SFServices):


  def DebugPrint(self, *args):



def CreateScriptService(service, *args):
```

SF_Exception.xba

```
Function DebugPrint(ParamArray Args)
  ...
End Function
```

Architecture – Common debugging/console

When mouse hovers ScrollBar1, next script is triggered

```
def ControlEvent(poEvent):
  a = CreateScriptService('SFDialogs.DialogEvent', poEvent)
  if a is not None:
    exc = CreateScriptService('exception')
    exc.DebugPrint('Event triggered from Python in control', \
             a.Name, 'of dialog', a.Parent.Name)
```

Python code

When mouse hovers ScrollBar2, next script is triggered

```
Sub ControlEvent(Optional poEvent As Object)
Dim a
  Set a = CreateScriptService("SFDialogs.DialogEvent", poEvent)
  If Not IsNull(a) Then SF_Exception.DebugPrint( _
     "Event triggered from Basic in control", a.Name, _
     "of dialog", a.Parent.Name)
End Sub
```

Basic code

Architecture – Common debugging/console

When mouse hovers ScrollBar1, next script is triggered (Python)

```
def ControlEvent(poEvent):
  a = CreateScriptService('SFDialogs.DialogEvent', poEvent)
  if a is not None:
    exc = CreateScriptService('exception')
    exc.DebugPrint('Event triggered from Python in control', \
            a.Name, 'of dialog', a.Parent.Name)
```

When mouse hovers S

```
Sub ControlEvent(Opt
Dim a
  Set a = CreateScrip
  If Not IsNull(a) T
     "Event triggered
     "of dialog", a.P
End Sub
```

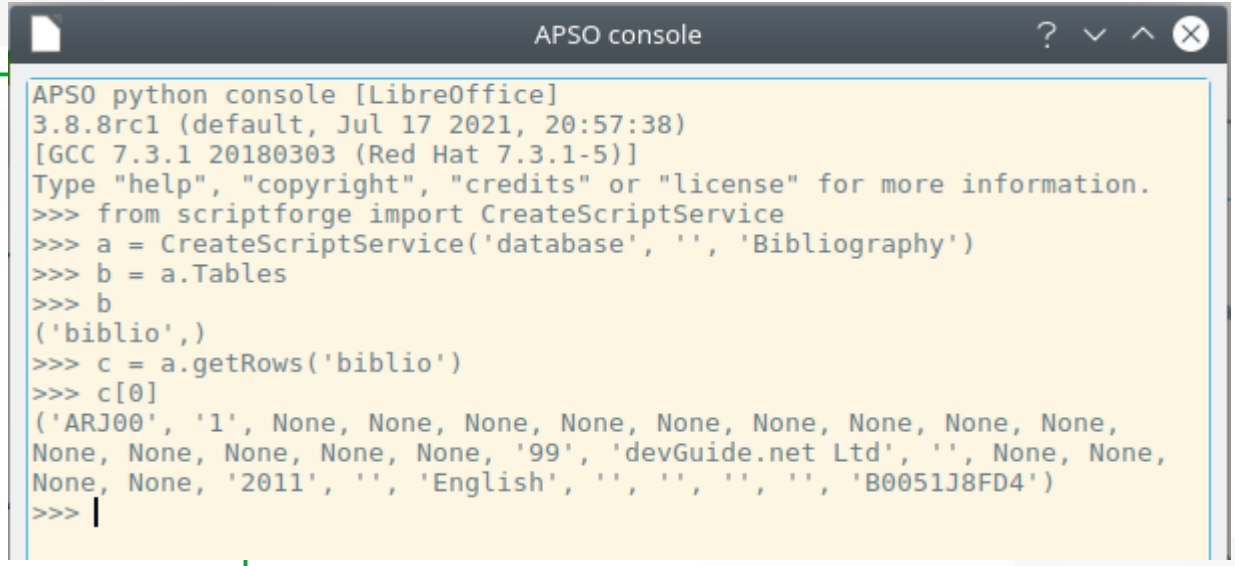
ScriptForge

```
15:00:52 -> Event triggered from Basic in control   ScrollBar2  of dialog   QADialog
15:00:52 -> Event triggered from Python in control  ScrollBar1  of dialog   QADialog
15:00:54 -> Event triggered from Basic in control   ScrollBar2  of dialog   QADialog
15:00:55 -> Event triggered from Python in control  ScrollBar1  of dialog   QADialog
15:00:57 -> Event triggered from Basic in control   ScrollBar2  of dialog   QADialog
15:00:58 -> Event triggered from Python in control  ScrollBar1  of dialog   QADialog
15:00:59 -> Event triggered from Basic in control   ScrollBar2  of dialog   QADialog
15:01:00 -> Event triggered from Python in control  ScrollBar1  of dialog   QADialog
```

Architecture – Common Python shell (the APSO console)

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Exception(SFServices):


  def PythonShell(self, ...):



def CreateScriptService(service, *args):
```



userscript.py

```
exc = CreateScriptService('exception')
exc.PythonShell()
print(arg1, arg2, ...)
```

Python code

userscript.xba

```
PythonPrint(arg1, arg2, ...)
```

Basic code

Architecture – Mutual helpers

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Platform(SFServices):


  def Architecture(self):



def CreateScriptService(service, *args):
```

ScriptForgeHelper.py

```
import platform
import hashlib
import filecmp
import webbrowser
import json
```

SF_String.xba

```
Function HashStr( … )
```

SF_Platform.xba

```
Function Architecture()
```

Basic code

Architecture – Mutual helpers

scriptforge.py

```
class ScriptForge:
  # ScriptProvider protocol

class SFServices:
  def __init__(objectreference, ...):
  def __getattr__(…):
  def __setattr__(…):
  def GetProperty(…):
  def Properties():
  def SetProperty(…):
  def Dispose():

class SF_Basic(SFServices):


  def MsgBox(self, ...):



def CreateScriptService(service, *args):
```

SF_PythonHelper.xba

```
Functions:
   Cdate, DateAdd, DateDiff, DatePart, Now*
   CdateFromUnoDateTime*, CDateToUnoDateTime*
   ConvertFromUrl, ConvertToUrl
   CreateUnoService
   Format
   GetGuiType, GetSystemTicks
   GlobalScope
   InputBox, MsgBox
   RGB*
   StarDesktop*
   ThisComponent*, ThisDatabaseDocument*
   Xray

(*) Python emulation
```

Help pages

Running Python scripts on LibreOffice

Depending on what you intend to achieve, you may choose one of the following approaches to running Python scripts in LibreOffice:

- **Run Scripts inside the current LibreOffice process:** Python scripts are executed from within the LibreOffice process by using the **Tools - Macros - Run Macro** menu or the APSO extension to call user scripts stored in the Python scripts folder. You can also use the APSO Python shell to interactively run Python scripts.
- **Run Scripts separately from the LibreOffice process:** Python scripts are executed from an external process that connects to an ongoing LibreOffice process using a socket.

If you plan to run scripts from inside the LibreOffice process, it is recommended to install the APSO (Alternative Script Organizer for Python) extension. However, to develop Python scripts from outside LibreOffice, you can choose your preferred Python IDE.

New in ScriptForge 7.2+

The “Dialog”, “DialogControl” services - TreeControl

Run dialogs designed with the Basic IDE

Properties

- *CurrentNode, RootNode, XTreeDataModel*

Methods

- *AddSubNode, AddSubTree, CreateRoot, FindNode*

Events

- *OnNodeExpanded, OnNodeSelected*

A treecontrol is good at visualizing tabular data

Node expansion can be done statically or dynamically

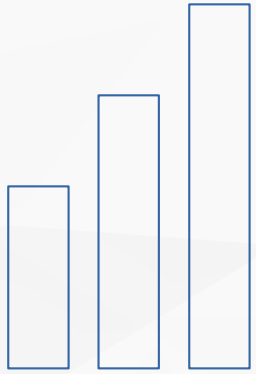
```
Flat data     >>>>     Resulting subtree
A1   B1   C1                |__   A1
A1   B1   C2                      |__   B1
A1   B2   C3                            |__   C1
A2   B3   C4                            |__   C2
A2   B3   C5                      |__   B2
A3   B4   C6                            |__   C3
                          |__   A2
                                |__   B3
                                      |__   C4
                                      |__   C5
                          |__   A3
                                |__   B4
                                      |__   C6
```

New in ScriptForge 7.3+ - Charts

- Chart service for Calc documents

Python code

```
# Get data
     db = CreateScriptService('database', …)
     data  = db.GetRows('SELECT …', header = True)
     db.CloseDatabase()
# Import data in Calc
     ui = CreateScriptService('UI')
     calc = ui.CreateDocument('Calc', hidden = True)
     datarange = calc.SetArray('Sheet1.A1', data)
# Make and export chart
     chart = calc.CreateChart('chart1', 'Sheet1', datarange)
     chart.ChartType = 'Bar'
     chart.Dim3D = 'cone'
     chart.Legend = True
     file = '/tmp/...'
     chart.ExportToFile(file, 'PNG')
     calc.CloseDocument(False)
# Display chart in dialog (or form …)
     dialog = CreateScriptService('dialog', …)
     dialog.Controls('aChart').Picture = file
     dialog.Execute()
```



New in ScriptForge 7.3+

- Chart service for Calc documents
- Printers
- PDF
- Dialogs translation, table controls in dialogs

In master

- SFWidgets library
  - Toolbar, ToolbarControl services
  - PopupMenu service
- Number service
  - Equality, roundings, unit conversions, number to text
- Region service
  - Locales, timezones, UTC, local languages, day/month names

Thank you …

LibreOffice
The Document Foundation

- The team
  - Jean-Pierre Ledure, Rafael Lima, Alain Romedenne
- Documentation
  - https://help.libreoffice.org/7.2/en-US/text/sbasic/shared/03/lib_ScriptForge.html?DbPAR=BASIC
- Sources
  - https://gitlab.com/LibreOfficiant/scriptforge
  - Gerrit
- Telegram groups
  - ScriptForge
  - LibreOffice Macros & Scripting