September 20 - 23, 2023

# Workshop: Introduction to LibreOffice Development

## *LibreOffice SDK development (Java / Python)*

**Hossein Nourikhah**
Developer Community Architect at TDF

# Topics

- LibreOffice SDK
  - SDK examples
    - Python
    - Java
    - C++
  - JLOP examples
- Writing extensions
  - Structure of an extension
    - In BASIC
    - In Java
    - In Python

# LibreOffice SDK

# LibreOffice SDK

- Purpose: Work with LibreOffice components in other programs
- Name: Office development kit (odk)
  - Usually referred to as LibreOffice SDK
- Built using --enable-odk
- Separate binaries
- Structure of SDK
  - Inside odk/ in sources
  - Is installed in instdir/sdk during build
- Containing loaders, makefiles, examples, etc.

# SDK examples

- SDK examples
  - https://api.libreoffice.org/examples/examples.html
  - Latest version alongside docs
    - In the sources → odk/examples/examples.html
    - In the binary installation → sdk/examples/examples.html
  - Examples from
    - DevGuide
    - Java
    - Python
    - C++
    - BASIC

# Running SDK examples (Python)

- No need to set anything / Just run the Python file with internal Python
  - Source build → instdir/program/python
  - Windows binary install → C:\Program Files\LibreOffice\program\python
  - Linux binary install → /opt/libreoffice7.6/program/python
- Installing additional packages using pip
  - Use PIP bootstrap → bootstrap.pypa.io/get-pip.py
    - Install using PIP → python -m pip install nicepackage
- External Python can be used, but it needs setup

# Running SDK examples (Java)

- Steps to run the C++/Java examples
  - 1) running **`setsdkenv_windows`** / **`setsdkenv_unix`**
  - 2) Building the example using make file
  - 3) Run the example using make file
- Example: java/DocumentHandling on Linux
  - **`cd instdir/sdk; ./setsdkenv_unix`**
  - **`cd examples/java/DocumentHandling`**
  - **`make DocumentHandling`**
  - **`make DocumentHandling.run`**
- **`Alternative: using of LibreOffice jar files`**
  - **`Compile using javac`**
  - **`Run using java`**

# Running SDK examples (C++)

- Requires more complex setup
  - Similar to Java, but compiler settings are important here
  - Set the environment, build and run using Makefile
  - Needs Cygwin shell and some other requirements
  - Has problems on Windows
- Better solution
  - Using cmake
  - Not finalized, but usable
    - Using cmake to build LibreOffice C++ SDK examples
      https://dev.blog.documentfoundation.org/?p=406
  - Only compiler and cmake will be sufficient

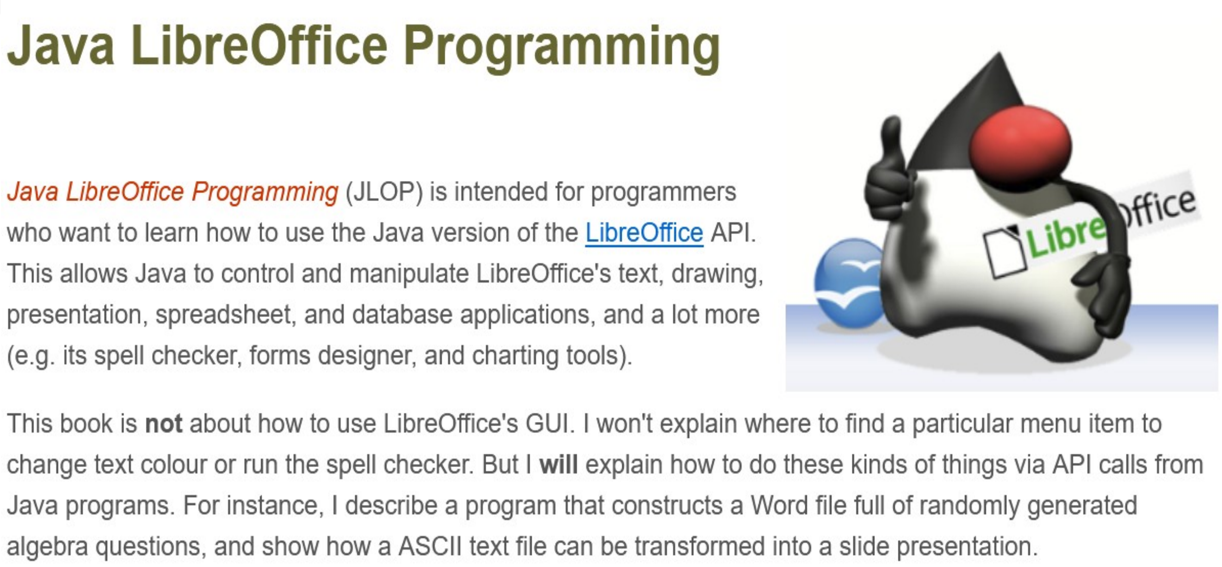
# LibreOfficeKit

- LibreOfficeKit
  - Another set of API
  - Used when visual output is needed
  - Used in LibreOffice Android and LibreOffice Online
  - Uses tiling mechanism to send the visual output
- Gtk TiledViewer Example
  - bin/run gtktiledviewer --lo-path=$PWD/instdir/program test.odt
  - Built on Linux

# LibreOffice UNO API

# Books

- Books
  - Java LibreOffice Programming book
    - Dr. Andrew Davison
  - LibreOffice Developers’ Guide
  - TDF Wiki: Documentation/DevGuide
  - JLOP → Java LibreOffice Programming



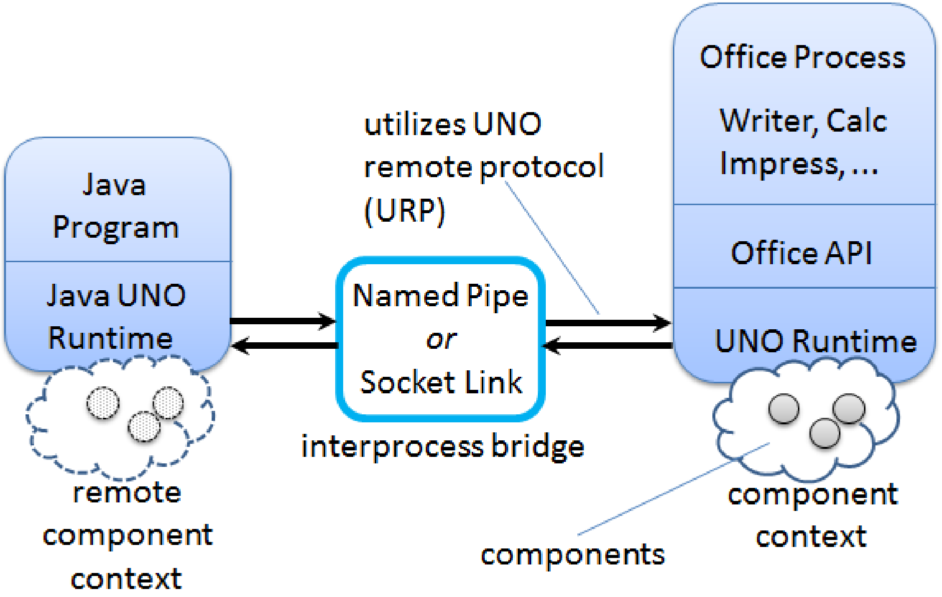
## Java LibreOffice Programming

*Java LibreOffice Programming* (JLOP) is intended for programmers who want to learn how to use the Java version of the LibreOffice API. This allows Java to control and manipulate LibreOffice's text, drawing, presentation, spreadsheet, and database applications, and a lot more (e.g. its spell checker, forms designer, and charting tools).

This book is **not** about how to use LibreOffice's GUI. I won't explain where to find a particular menu item to change text colour or run the spell checker. But I **will** explain how to do these kinds of things via API calls from Java programs. For instance, I describe a program that constructs a Word file full of randomly generated algebra questions, and show how a ASCII text file can be transformed into a slide presentation.

# LibreOffice UNO API

- LibreOffice UNO API
  - Based on UNO (Universal Network Objects)
  - Used to access objects across programming languages over network and sockets
  - Bridge between processes
  - Define and use components that can be used across processes and languages
  - Office will be used as a process listening on the network



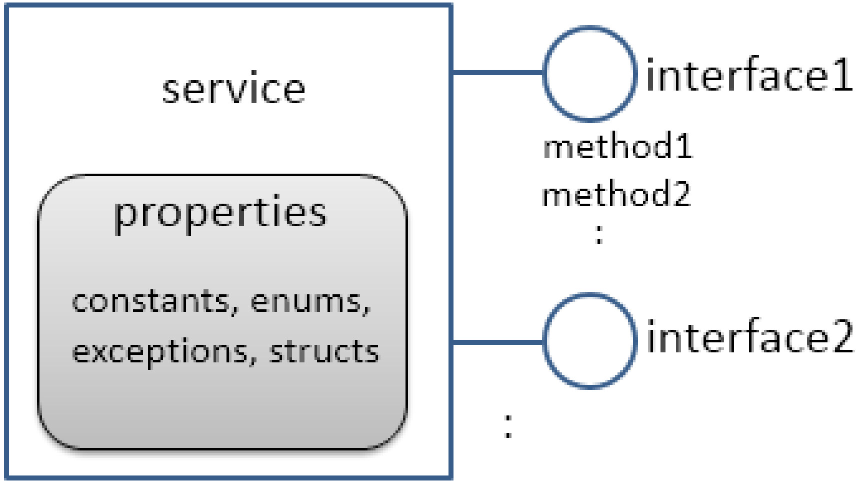
A Java Program Using Office
From JLOP book

# Applications and modules

- 6 different Applications
  - Writer, Calc, Draw, Impress, Base, Math
- Each application → one or more modules
  - Writer → Text
  - Impress → Presentation / Drawing
- Prefix: com.sun.star → css (shortcut)
- Documentation
  - http://api.libreoffice.org/docs/idl/ref/namespacecom_1_1sun_1_1star.html

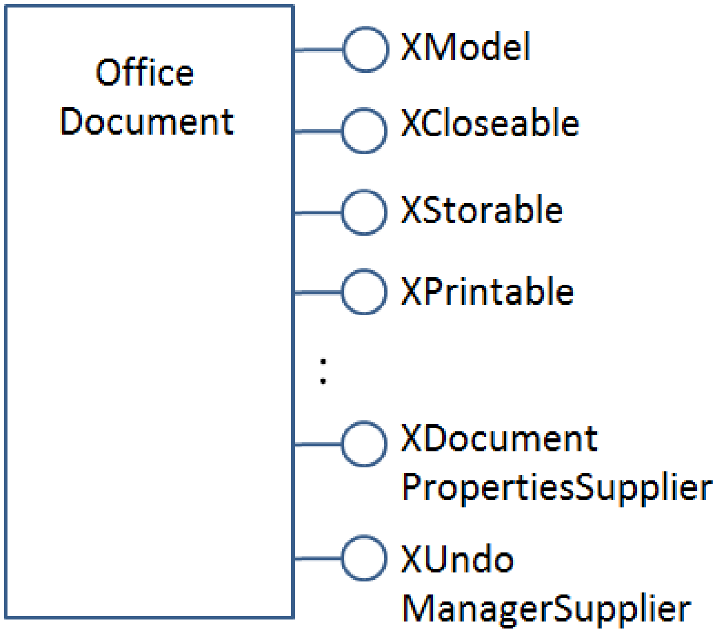
# LibreOffice API concepts

- API Concepts
  - Interface
    - Method prototype: Name, argument list, return type
    - No implementation / data
  - Property
    - Name/value pair to store data
  - Service
    - Interfaces and properties to support specific office feature
  - Component: implements a service

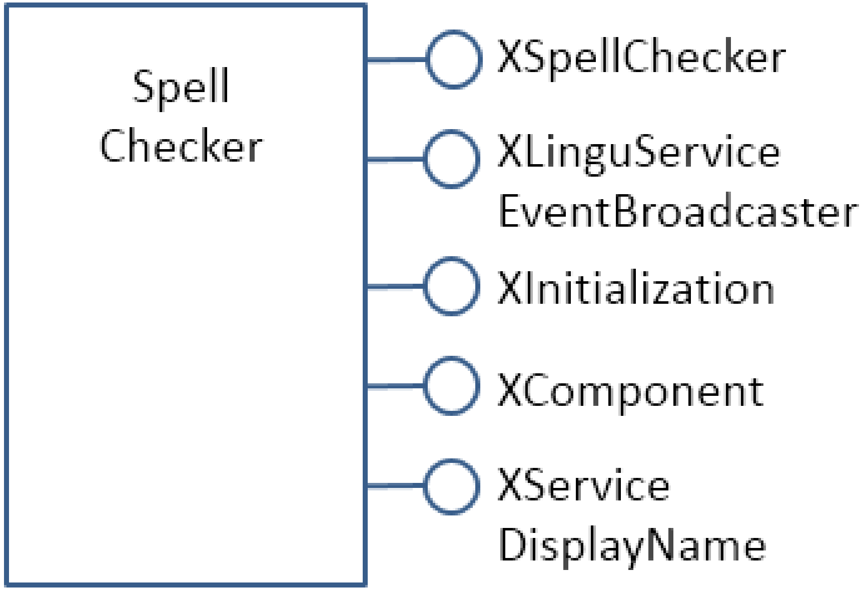


Services, Interfaces, Properties
From JLOP Book

# Example Services

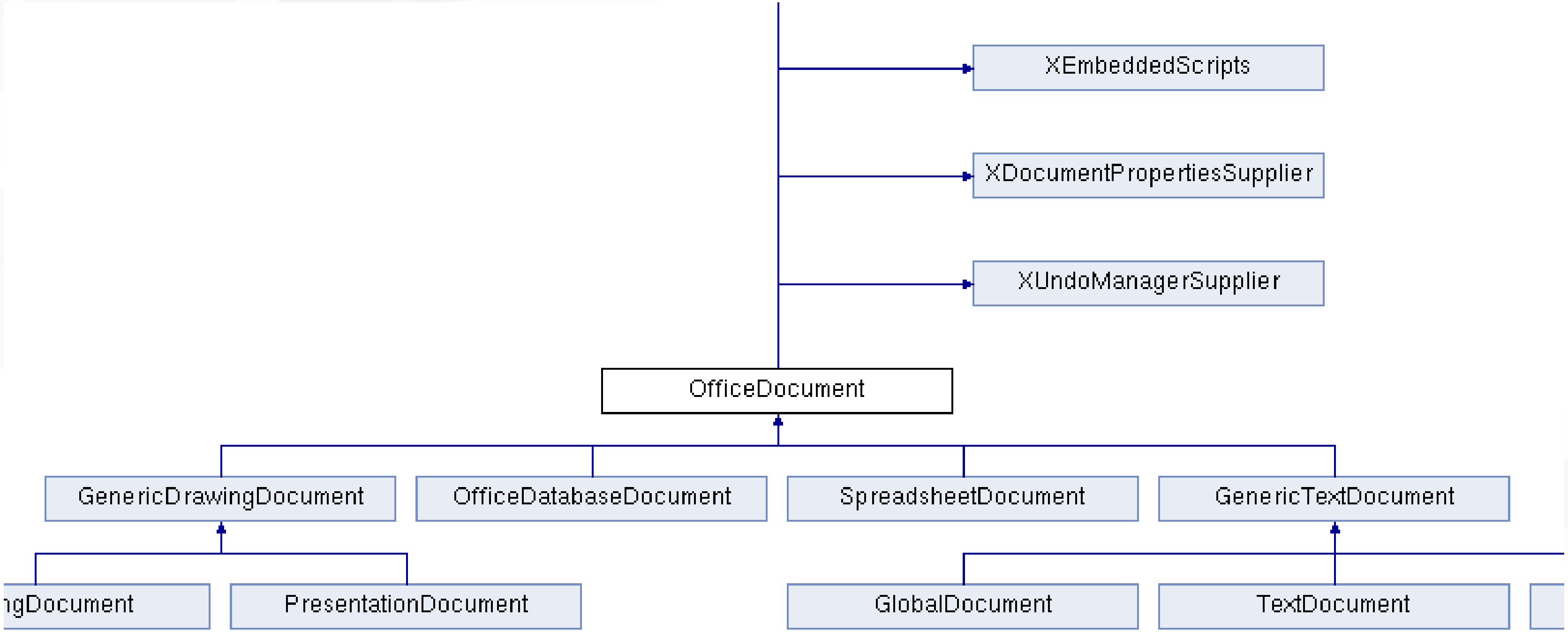


The OfficeDocument service
From JLOP book



The SpellChecker service.
From JLOP book

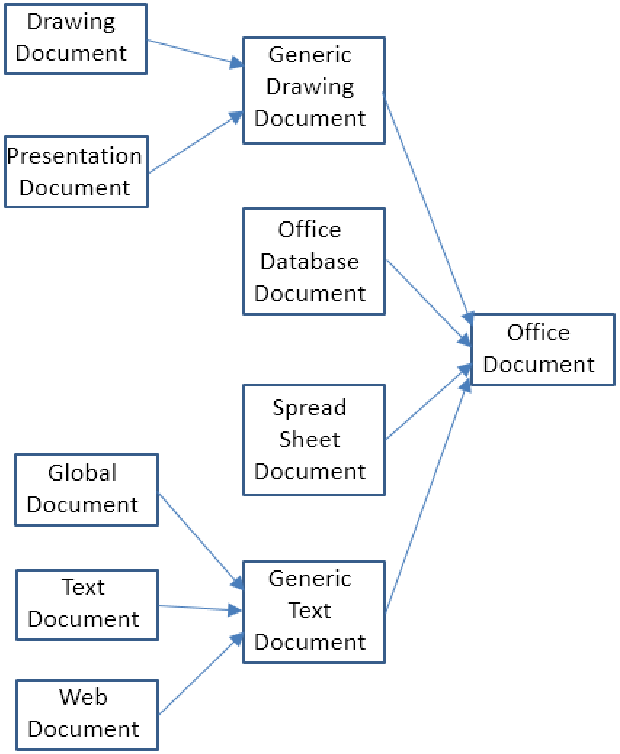
# Inheritance diagram



Inheritance for OfficeDocument Service
From JLOP book

# Hierarchy

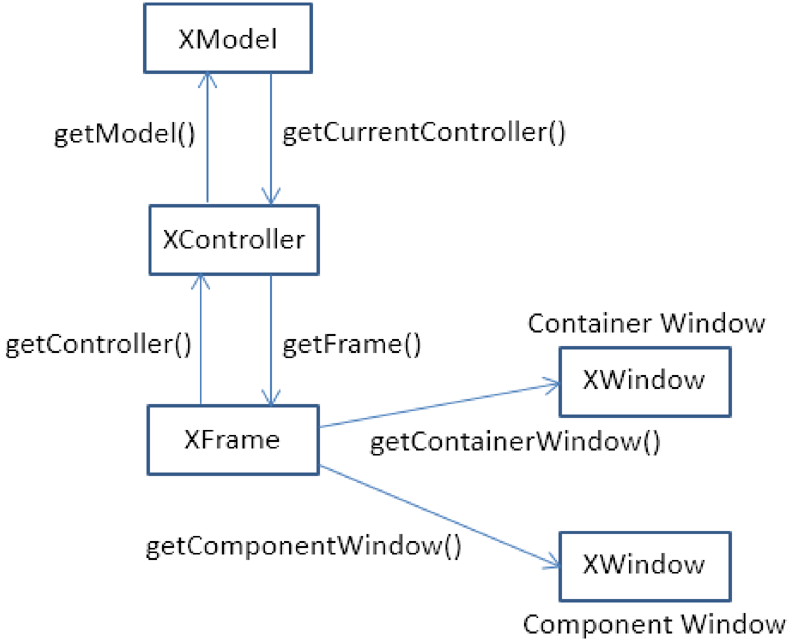
- Hierarchy
  - OfficeDocument as the superclass
    - Drawing and presentation documents have many similar properties because they share a common parent, GenericDrawingDocument
    - Also the same for text and web documents



OfficeDocument as a Superclass Service
From JLOP book

# FCM relationship

- FCM relationship
  - Similar to Model-View-Controller (MVC) pattern
    - Model: XModel
    - View: XFrame
      - getContainerWindow()
      - getComponentWindow() → parent
    - Controller: Xcontroller
  - Not every component supports all of the FCM relationship



# FCM relationship support

- Support level is different among components
  - Support both XModel and XComponent
    - Complex GUI elements → office document
    - We use ComponentLoder to load office documents
  - Support XModel but not XComponent
    - Usually does not load data → spell checker, document forms
  - No XModel but XComponent (XWindow object)
    - Simple GUI elements → Office help window

# API Documentation

- API documentation is available on api.libreoffice.org
  - https://api.libreoffice.org/docs/idl/ref/
    - Generated from offapi/ (high level) and udkapi/ (low level)
  - https://api.libreoffice.org/docs/cpp/ref/
    - Generated from the sources in odk/
  - https://api.libreoffice.org/docs/java/ref/
    - Generated from the sources in odk/
- Can be generated from the code locally using doxygen

# Starting and stopping LibreOffice

# Start LibreOffice

- Multiple methods
  - Running manually and listening
    - Run the command
      - `soffice "--accept=socket,host=localhost,port=2083;urp;StarOffice.ServiceManager"`
  - Bootstraping
    - `>>> import officehelper`
    - `>>> context = officehelper.bootstrap()`
    - The office is invoked automatically
    - If the office is not open, it will be opened
    - If it is open, it will be re-used

# Open document

- Opening a new document

```
service_manager = context.getServiceManager()
doc = desktop.loadComponentFromURL(
            "private:factory/swriter", "_blank", 0, ())
```

- Use the document url

```
doc = desktop.loadComponentFromURL(
        "file:///c:/Users/user/Desktop/test.odt", "_blank", 0, ())
```

# Properties

- Sending options

```
opts = (
   PropertyValue(Name="Overwrite", Value=True),
   PropertyValue(Name="FilterName", Value="writer8"),
)
```

# Save document

```
url = "file:///c:/Users/user/Desktop/test.odt"
opts = (
    PropertyValue(Name="Overwrite", Value=True),
    PropertyValue(Name="FilterName", Value="writer8"),
)
try:
    doc.storeAsURL(url, opts)
finally:
    doc.dispose()
```

# Complete program

```python
import uno
import officehelper
from com.sun.star.beans import PropertyValue
def main():
    context = officehelper.bootstrap()
    service_manager = context.getServiceManager()
    desktop = context.getServiceManager().createInstanceWithContext(
        "com.sun.star.frame.Desktop", context)

    # Load a new text document
    doc = desktop.loadComponentFromURL(
        "private:factory/swriter", "_blank", 0, ())
    if doc:
        # Get the XText interface for the document
        text = doc.getText()
        if text:
            text.setString("Hello World!")
            url = "file:///c:/Users/user/Desktop/test.odt"
            opts = (
                PropertyValue(Name="Overwrite", Value=True),
                PropertyValue(Name="FilterName", Value="writer8"),
            )
            try:
                doc.storeAsURL(url, opts)
            finally:
                doc.dispose()
if __name__ == "__main__":
    main()
```

# Creating extensions: Python

# Structure of an extension

- Extension is essentially a zip file renamed to .oxt
  - Contents:
    - META-INF/manifest.xml: Specification of the script(s), menu/toolbar and language files
    - pkg-description/pkg-description.en: Description of the extension in text, which can be also in languages other than English
    - Icons: Icons of the extension, if any
    - registration/license.txt: License of the extension
    - description.xml: Description of the extension in XML format, as displayed in the extension manager
    - main.py: The main script. Then name can be anything but it should be specified in the META-INF/manifest.xml

# Structure of the Python file

- To be usable in LibreOffice, this 2 lines should be in the Python file

```
g_ImplementationHelper = unohelper.ImplementationHelper()
g_ImplementationHelper.addImplementation(MainJob,
                                            "org.extension.sample.do",
                                            ("com.sun.star.task.Job",), )
```

- This import is also required:

```
from com.sun.star.task import XJobExecutor
```

- A class with this definition is needed:

```
class MainJob(unohelper.Base, XjobExecutor)
```

  - It should have this method:

```
def trigger(self, args):
      ...
```

# How to debug the extension?

- To be able to debug the extension
  - The script should be also run-able
  - The context and service manager should be determined correctly according to the different possible contexts
    - Running inside LibreOffice via APSO (Alternative Script Organizer for Python)
    - Running inside an extension
    - Running inside an IDE like PyCharm or VS Code

# Example main function

```
def main():
    try:
        ctx = XSCRIPTCONTEXT
    except NameError:
        ctx = officehelper.bootstrap()
        if ctx is None:
            print("ERROR: Could not bootstrap default Office.")
            sys.exit(1)
    job = MainJob(ctx)
    job.trigger("hello")

if __name__ == "__main__":
    main()
```

# Sample MainJob class

```
class MainJob(unohelper.Base, XJobExecutor):
    def __init__(self, ctx):
        self.ctx = ctx
        try:
            self.sm = ctx.getServiceManager()
            self.desktop = XSCRIPTCONTEXT.getDesktop()
        except NameError:
            self.sm = ctx.ServiceManager
            self.desktop = 
self.ctx.getServiceManager().createInstanceWithContext(
                "com.sun.star.frame.Desktop", self.ctx)
```

# Sample trigger function

```
def trigger(self, args):
    desktop = self.ctx.ServiceManager.createInstanceWithContext(
        "com.sun.star.frame.Desktop", self.ctx)
    model = desktop.getCurrentComponent()
    if not hasattr(model, "Text"):
        model =
self.desktop.loadComponentFromURL("private:factory/swriter", "_blank",
0, ())
    text = model.Text
    cursor = text.createTextCursor()
    text.insertString(cursor, "Hello argument -> " + args + "\n", 0)
```

# Sample META-INF/manifest.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE manifest:manifest PUBLIC "-//OpenOffice.org//DTD Manifest 1.0//EN" "Manifest.dtd">
<manifest:manifest xmlns:manifest="http://openoffice.org/2001/manifest">
<manifest:file-entry manifest:media-type="application/vnd.sun.star.uno-component;type=Python" manifest:full-path="main.py" />
<manifest:file-entry manifest:full-path="pkg-desc/pkg-description.en" manifest:media-type="application/vnd.sun.star.package-bundle-description;locale=en"/>
<manifest:file-entry manifest:full-path="Addons.xcu" manifest:media-type="application/vnd.sun.star.configuration-data"/>
</manifest:manifest>
```

# Sample Addons.xcu for a submenu

```
<node oor:name="Submenu">
  <node oor:name="M1" oor:op="replace">
    <prop oor:name="Title">
      <value xml:lang="en-US">Example item</value>
    </prop>
    <prop oor:name="URL">
      <value>service:org.extension.sample.do?hello</value>
    </prop>
    <prop oor:name="Target" oor:type="xs:string">
      <value>_self</value>
    </prop>
  </node>
</node>
```

# Sample Description

```
<?xml version='1.0' encoding='UTF-8'?>
<description xmlns="http://openoffice.org/extensions/description/2006"
xmlns:dep="http://openoffice.org/extensions/description/2006"
xmlns:xlink="http://www.w3.org/1999/xlink">
<identifier value="org.extension.example"/><icon><default
xlink:href="icons/image_42.png" /></icon>
<version value="1.0"/>
<registration>
  <simple-license accept-by="admin" default-license-id="ID0" suppress-on-update="true"
  ><license-text xlink:href="registration/license.txt" lang="en" license-id="ID0"
  /></simple-license>
</registration>
<publisher>
<name xlink:href="mailto:developer@mywebsite.org">Developer</name></publisher>
<display-name>
  <name>Example extension</name>
</display-name>
</description>
```

# How to use the sample?

- Use the structure
  - Most of the basic skeleton is the same for
  - It should be customized for the specifics of the new extension
    - Different name and description
    - Different menus, toolbar icons, etc.
    - Different scripts
    - Different programming languages
    - Different UI languages
    - The source code itself

# Thank you …

- Thank you for your patience!

LibreOffice
The Document Foundation

All text and image content in this document is licensed under the Creative Commons Attribution-Share Alike 4.0 License (unless otherwise specified). “LibreOffice” and “The Document Foundation” are registered trademarks. Their respective logos and icons are subject to international copyright laws. The use of these thereof is subject to trademark policy.