FOSDEM 2021

OpenOffice UNO Programming with Groovy

Carl Marcum
Apache OpenOffice PMC

Biography

- 39 years in Manufacturing Engineering
- 30 years Programming and Application Development
- Sun Certified Java Programmer 2008
- Founded Code Builders, LLC in 2016
- OpenOffice Committer since 2011
- OpenOffice PMC since 2016
- Current VP OpenOffice

Agenda

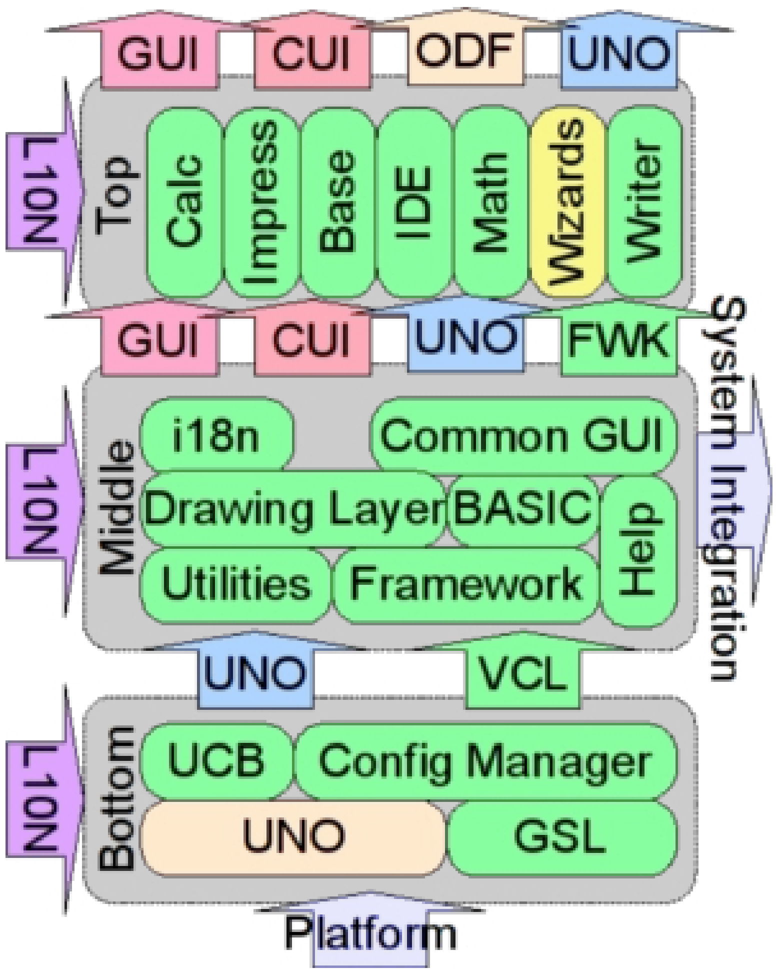
- Apache OpenOffice and UNO
- Past Project Work and Looking Forward
- UNO First Contact
- UNO Libraries for Java
- Apache Groovy
- Groovy Script UNO Client
- Groovy UNO Extension
- CLI Project Templates
- Calc Add-In Example
- Groovy Scripting Extension
- Sample Macros Extension
- Summary
- Questions?

Code Builders, LLC
Business Software Solutions

What is OpenOffice?

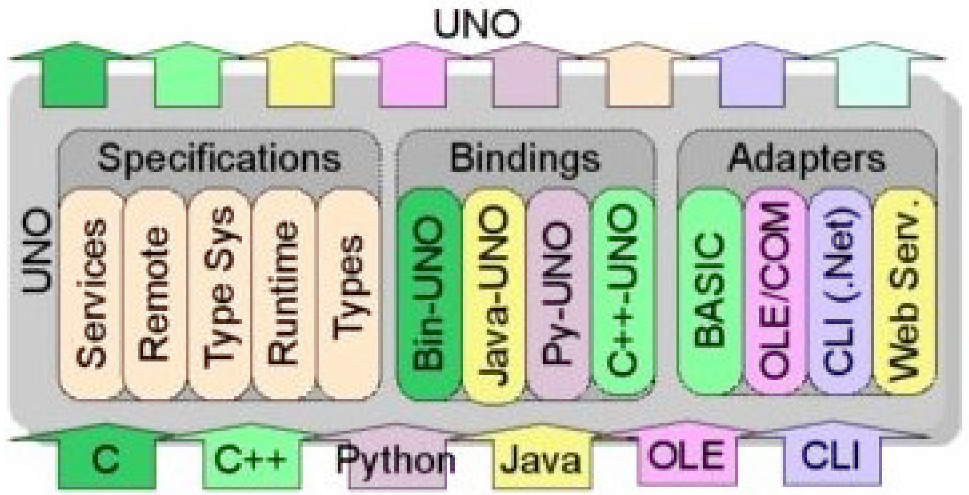
- Text Document
- Spreadsheet
- Presentation
- Drawing
- Database
- Formula

Why OpenOffice?



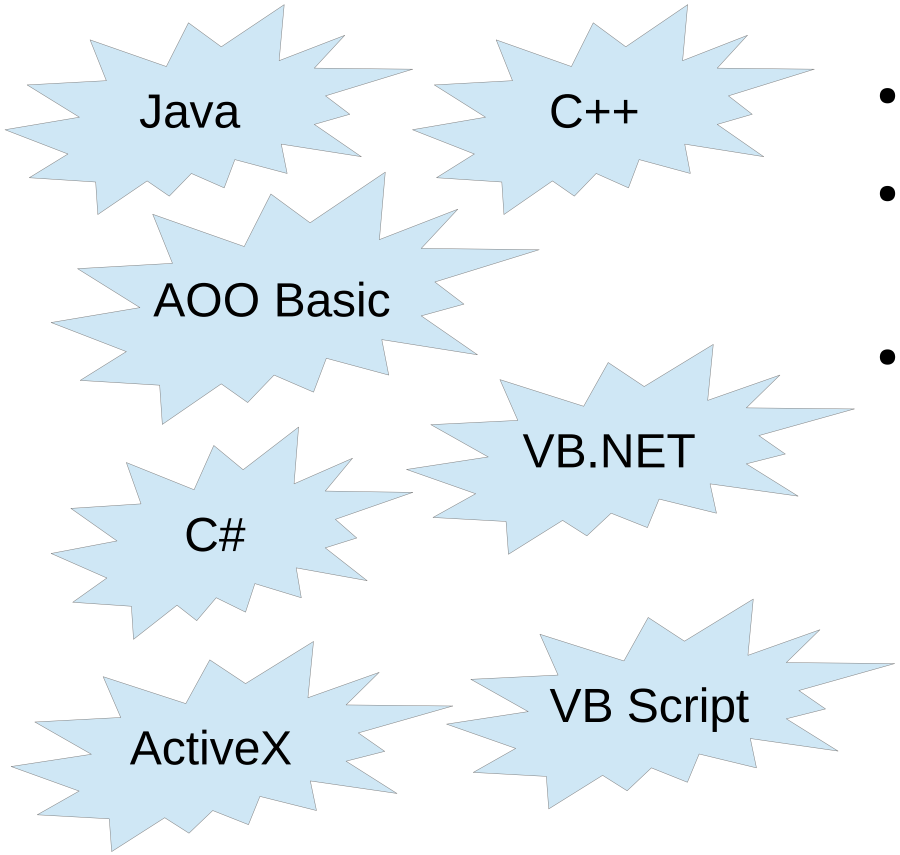
- Apache 2.0 Licensed
- Open Architecture
- Extension Mechanism
- Scripting Framework

What is UNO?



- UNO stands for Universal Network Objects
- Interface based component model
- Allows interoperability between languages and hardware architectures.
- Implemented in and used by any language with a binding.

Software Development Kit



- Available as a separate download.
- Contains libraries, binaries, and API documentation.
- Examples for Java, C++, OpenOffice Basic, CLI (C#, and VB.NET) and OLE (ActiveX and VB Script)

Past Work

- Focused on developer tools for client applications and extensions using OpenOffice UNO API's
- Updated NetBeans OpenOffice API Plugin for

  AOO 3.x / NetBeans 7.x

  through

  AOO 4.1 / NetBeans 8.1

Looking to the Future

- Great new languages for the JVM
- New build tools like Gradle gaining in popularity
- Ecosystem around Apache Groovy
- Apache Licensed

What is Groovy?

Groovy is...

- An optionally typed dynamic language for the JVM.
- Static-typing and compilation capable.
- Aimed at developer productivity with it's familiar and easy to learn syntax.
- Integrates smoothly with any Java program.

Why Groovy?

- Concise Java-like syntax
- Vibrant ecosystem
- Features like Closures and builders
- API Extensions with meta-programming.
- Domain-Specific Languages
- Great for writing Tests and build automation.

Groovyisms

- No need to use semicolons.
- Classes and methods public by default.
- Properties are private fields and get public *getter* and *setter* methods without needing to specify them.
- Property-like field access:

  myObj.myProp = "some value"

- GString interpolation :

  "Is this ${groovy} or what?"

- Closures:

  someList.each { item ->

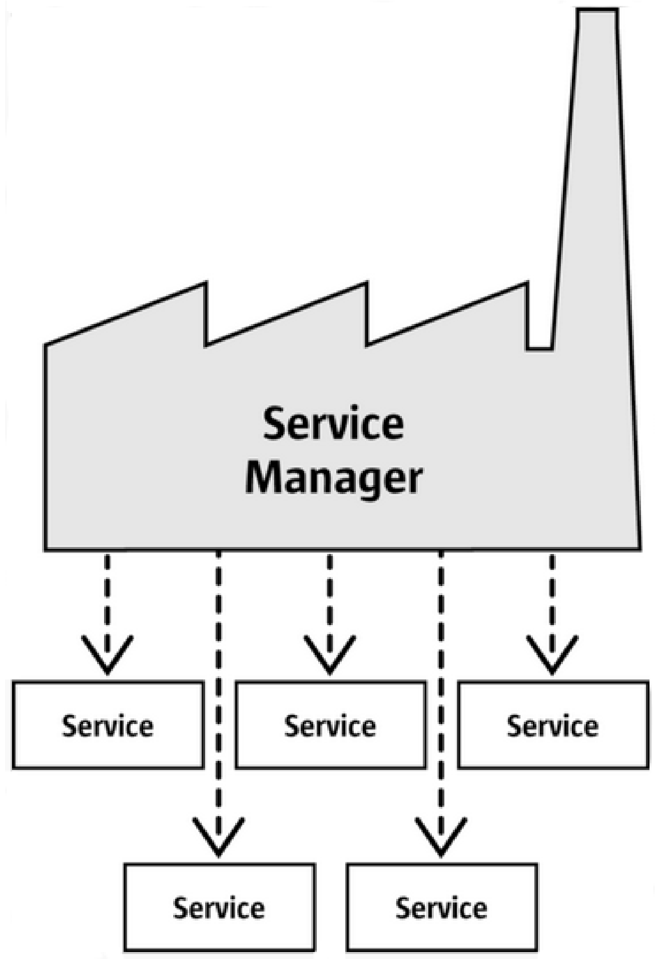
  println(item) }

- No primitives:

  int is actually an *Integer*

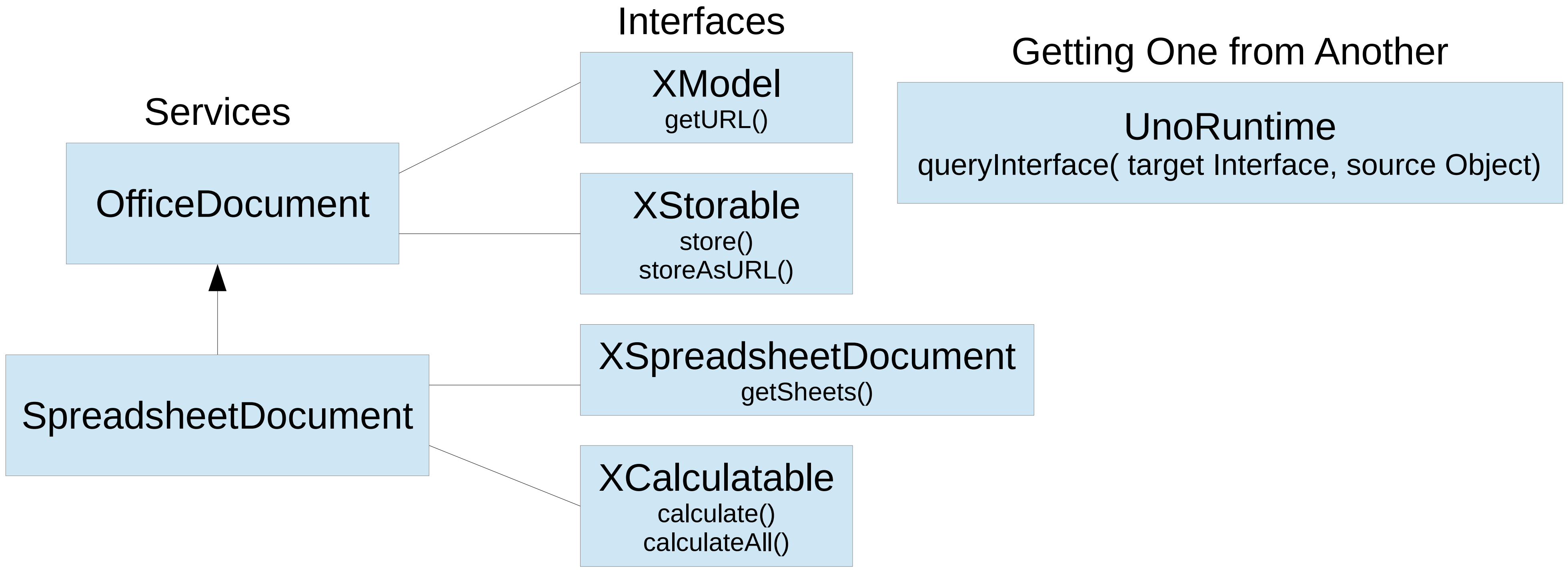
- *println()* replaces *System.out.println()*

Service Managers



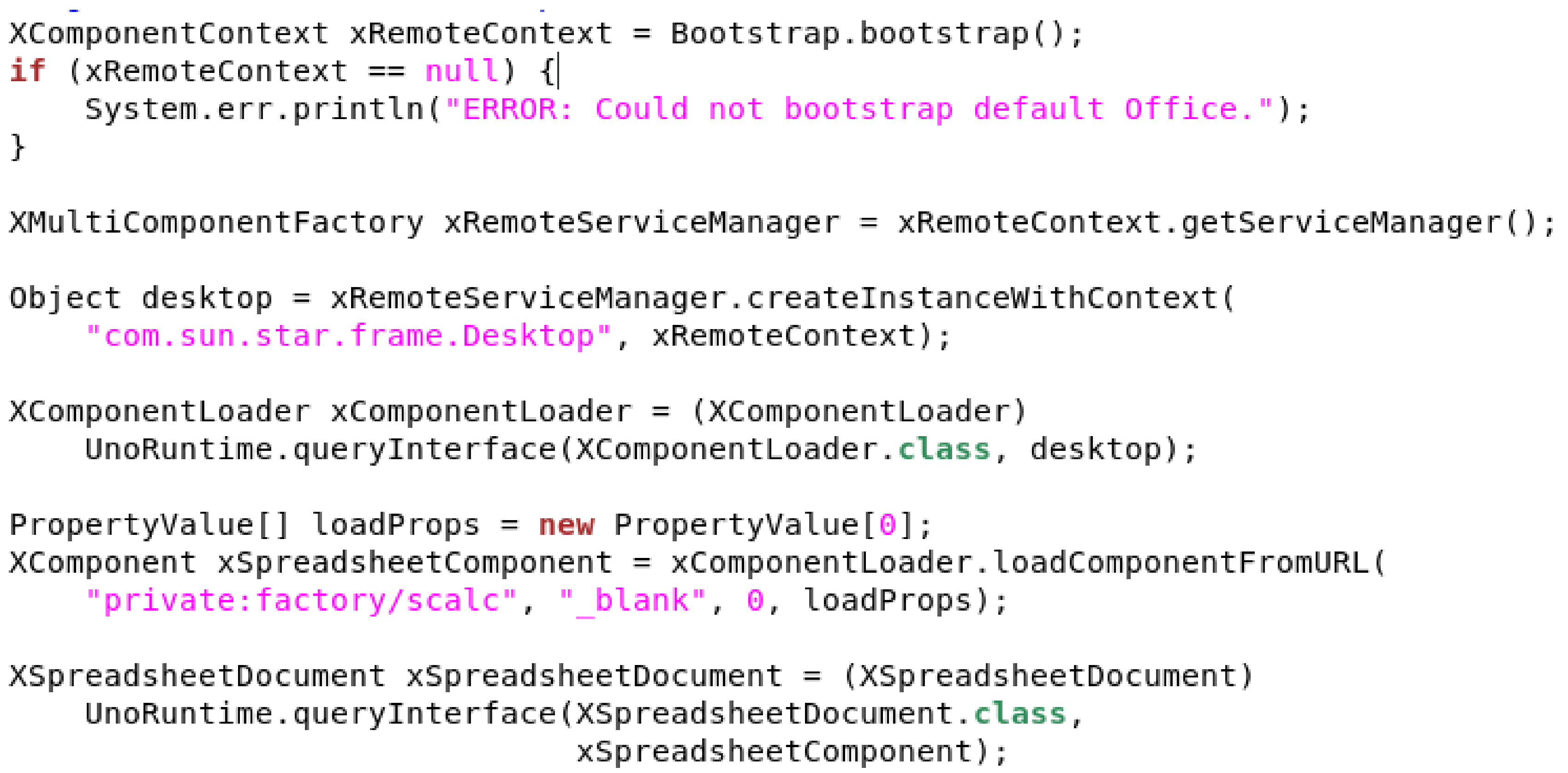
- Desktop
- Configuration Provider
- Database Context
- System Shell Execute
- Global Settings

Services and Interfaces



First Contact

Java Example from SDK



```java
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Getting a Connection

```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

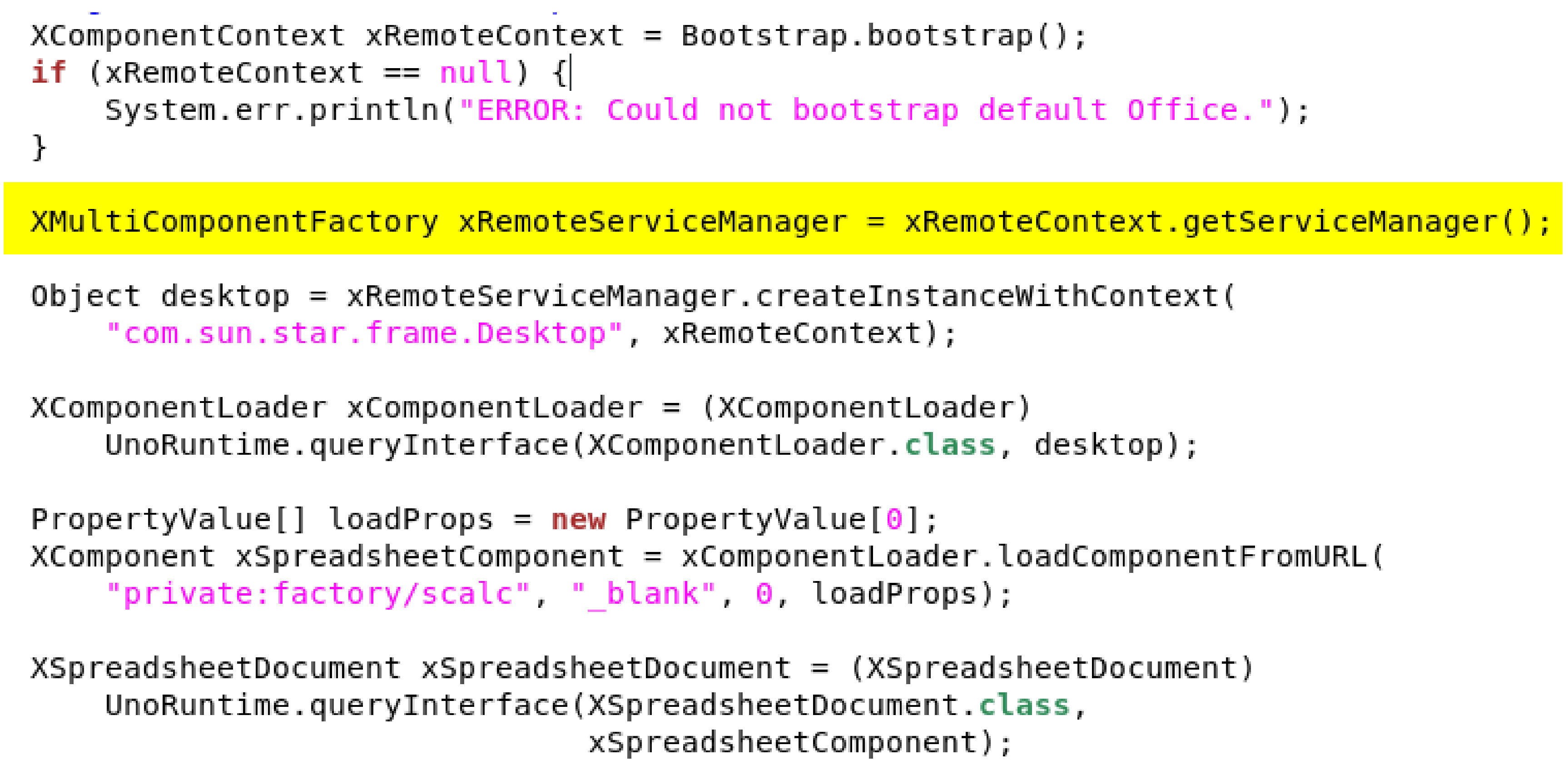
XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Getting a Service Manager



```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Getting the Desktop Object

```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Getting a Component Loader

```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Loading a Spreadsheet Component

```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

First Contact

Getting a Spreadsheet Document

```
XComponentContext xRemoteContext = Bootstrap.bootstrap();
if (xRemoteContext == null) {
    System.err.println("ERROR: Could not bootstrap default Office.");
}

XMultiComponentFactory xRemoteServiceManager = xRemoteContext.getServiceManager();

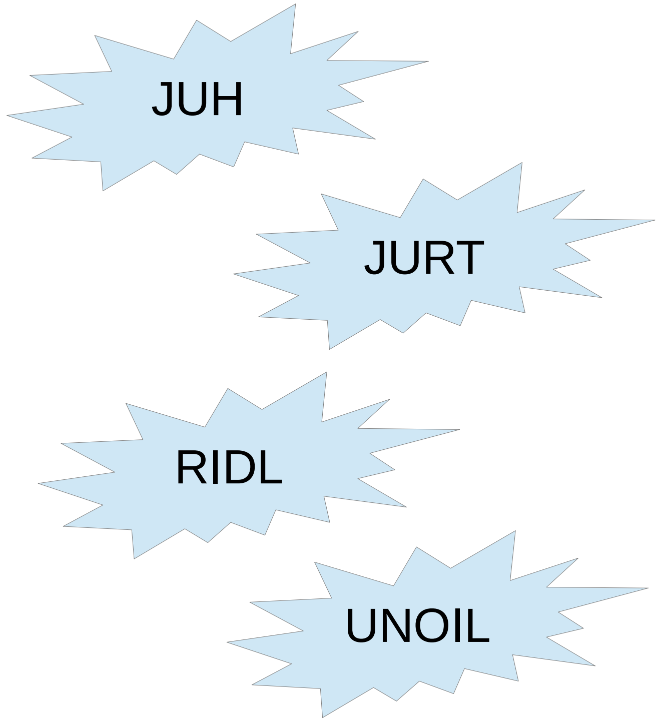
Object desktop = xRemoteServiceManager.createInstanceWithContext(
    "com.sun.star.frame.Desktop", xRemoteContext);

XComponentLoader xComponentLoader = (XComponentLoader)
    UnoRuntime.queryInterface(XComponentLoader.class, desktop);

PropertyValue[] loadProps = new PropertyValue[0];
XComponent xSpreadsheetComponent = xComponentLoader.loadComponentFromURL(
    "private:factory/scalc", "_blank", 0, loadProps);

XSpreadsheetDocument xSpreadsheetDocument = (XSpreadsheetDocument)
    UnoRuntime.queryInterface(XSpreadsheetDocument.class,
                              xSpreadsheetComponent);
```

Java UNO Libraries



- Java UNO Helper: Tools and Adapters
- Java UNO Runtime: Implements Java UNO
- RIDL: Implements Base Types and Access
- UNOIL: Java UNO Implementation. Generated from UNO IDL files.
- Interface Definition Language is used to generate the class files of various implementation languages.

FOSDEM 2021

Groovy Script UNO Client

Groovy Script UNO Client

Grape Dependency Manager

```
@Grab('net.codebuilders:bootstrap-connector:4.1.6.0')

@Grab("net.codebuilders:juh:4.1.6")
@Grab("net.codebuilders:ridl:4.1.6")
@Grab("net.codebuilders:unoil:4.1.6")
@Grab("net.codebuilders:jurt:4.1.6")

@Grab('net.codebuilders:guno-extension:4.1.6.13')
```

- Single Script File
- Managed Dependencies
- Compiled at Runtime

Location of Office Executable

```
// location of openoffice executable soffice
static String oooExeFolder = "/opt/openoffice4/program"
// static String oooExeFolder = "C:/Program Files (x86)/OpenOffice 4/program"
```

Groovy Script UNO Client

Getting a Spreadsheet with Groovy

```
mxRemoteServiceManager = mxRemoteContext.getServiceManager()

Object desktop = mxRemoteServiceManager.createInstanceWithContext(
        "com.sun.star.frame.Desktop", mxRemoteContext)

XComponentLoader aLoader = UnoRuntime.queryInterface(XComponentLoader.class, desktop)

xComponent = aLoader.loadComponentFromURL(
        "private:factory/scalc", "_default", 0, new com.sun.star.beans.PropertyValue[0])

XSpreadsheetDocument xSpreadsheetDocument = UnoRuntime.queryInterface(
        XSpreadsheetDocument.class, xComponent)
```

Using the Groovy UNO Extension

```
XComponentLoader aLoader = mxRemoteContext.componentLoader

xComponent = aLoader.loadComponentFromURL(
        "private:factory/scalc", "_default", 0, new com.sun.star.beans.PropertyValue[0])

XSpreadsheetDocument xSpreadsheetDocument = xComponent.getSpreadsheetDocument(mxRemoteContext)
```

FOSDEM 2021

Groovy UNO Extension

Groovy Extensions

Adding a new Runtime method to XComponent

```
class SpreadsheetExtension {

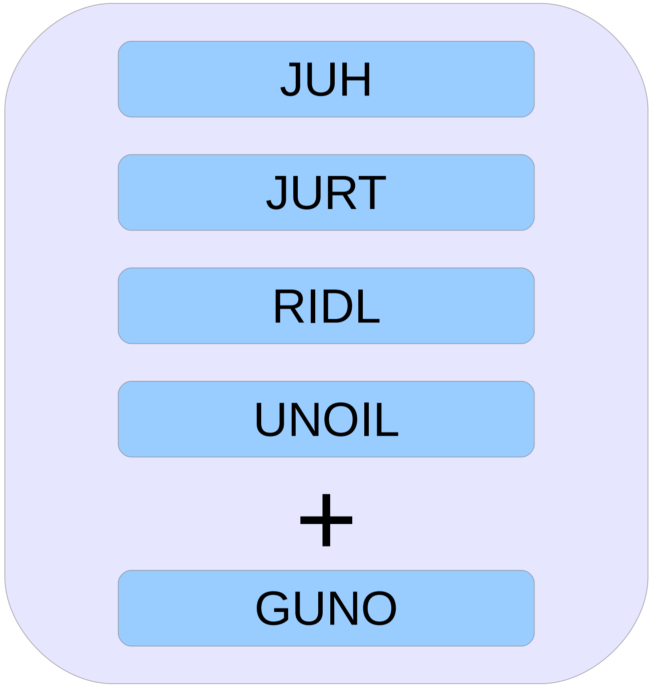

    /** Returns the spreadsheet document with the specified component context
     * @param mxRemoteContext the remote context.
     * @return XSpreadsheetDocument interface of the spreadsheet document.
     */
    static XSpreadsheetDocument getSpreadsheetDocument(final XComponent self, XComponentContext mxRemoteContext) {

        XSpreadsheetDocument xSpreadsheetDocument = null

        xSpreadsheetDocument = UnoRuntime.queryInterface(
                XSpreadsheetDocument.class, self)

        return xSpreadsheetDocument
    }
```

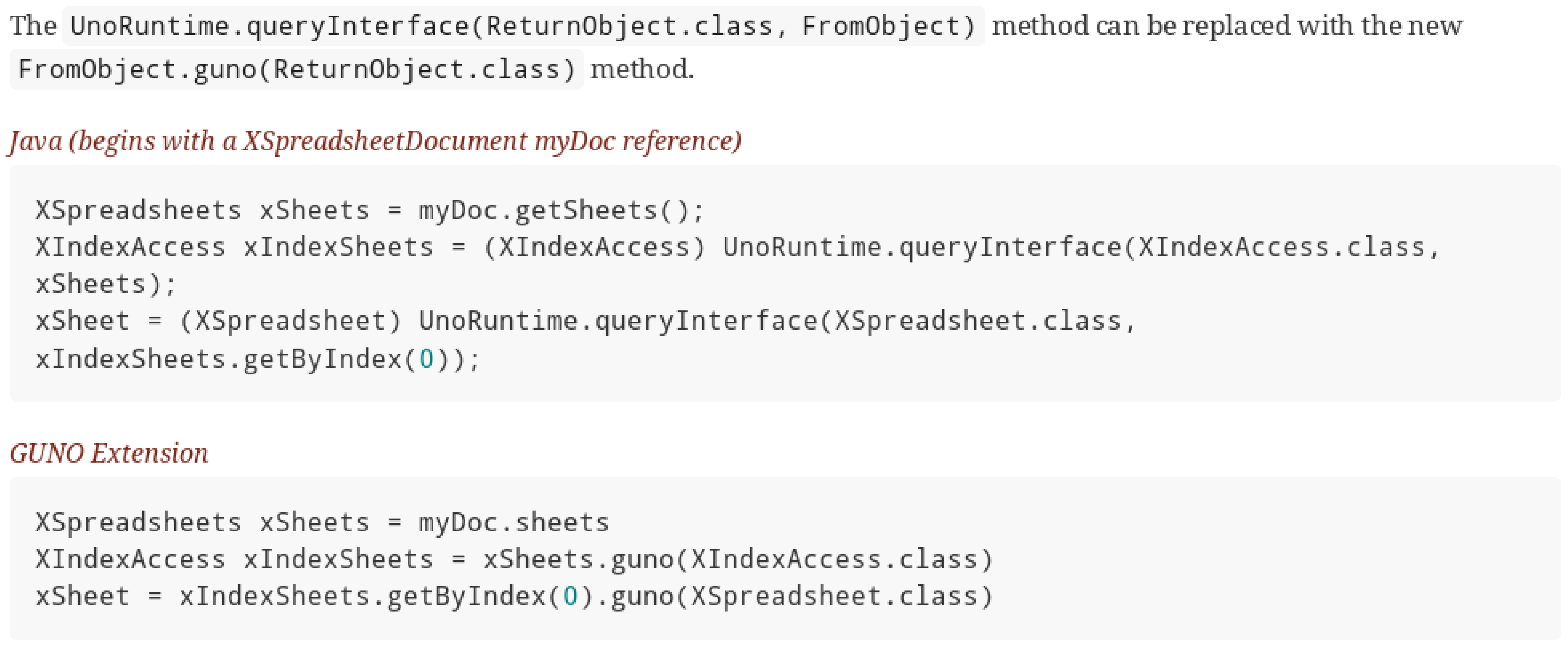
Groovy UNO Extension



- The goal of the Groovy UNO Extension is to allow UNO programming that is less verbose than using the Java UNO API's alone.
- Initial focus on Spreadsheet API's.

Groovy UNO Extension

Replacing the static queryInterface method..



The `UnoRuntime.queryInterface(ReturnObject.class, FromObject)` method can be replaced with the new `FromObject.guno(ReturnObject.class)` method.

Java (begins with a XSpreadsheetDocument myDoc reference)

```
XSpreadsheets xSheets = myDoc.getSheets();
XIndexAccess xIndexSheets = (XIndexAccess) UnoRuntime.queryInterface(XIndexAccess.class,
xSheets);
xSheet = (XSpreadsheet) UnoRuntime.queryInterface(XSpreadsheet.class,
xIndexSheets.getByIndex(0));
```

GUNO Extension

```
XSpreadsheets xSheets = myDoc.sheets
XIndexAccess xIndexSheets = xSheets.guno(XIndexAccess.class)
xSheet = xIndexSheets.getByIndex(0).guno(XSpreadsheet.class)
```

Groovy UNO Extension

Property Access

Example: Set the CellStyle of a spreadsheet Cell `xCell`.

Java

```
XPropertySet xCellProps = (XPropertySet)UnoRuntime.queryInterface(XPropertySet.class, xCell);
xCellProps.setPropertyValue("CellStyle", "Result");
```

GUNO Extension

```
XPropertySet xCellProps = xCell.guno(XPropertySet.class)
xCellProps.putAt("CellStyle", "Result")
```

GUNO Extension using the Subscript operator for assignment.

```
xCellProps["CellStyle"] = "Result"
```

Groovy UNO Extension

Cell Contents

```
String getFormulaOfCell(int column, int row)
void setFormulaOfCell(int column, int row, String value)
Double getValueOfCell(int column, int row)
void setValueOfCell(int column, int row, float value)
```

Without Extension (begins with an XSpreadsheet xSpreadsheet reference)

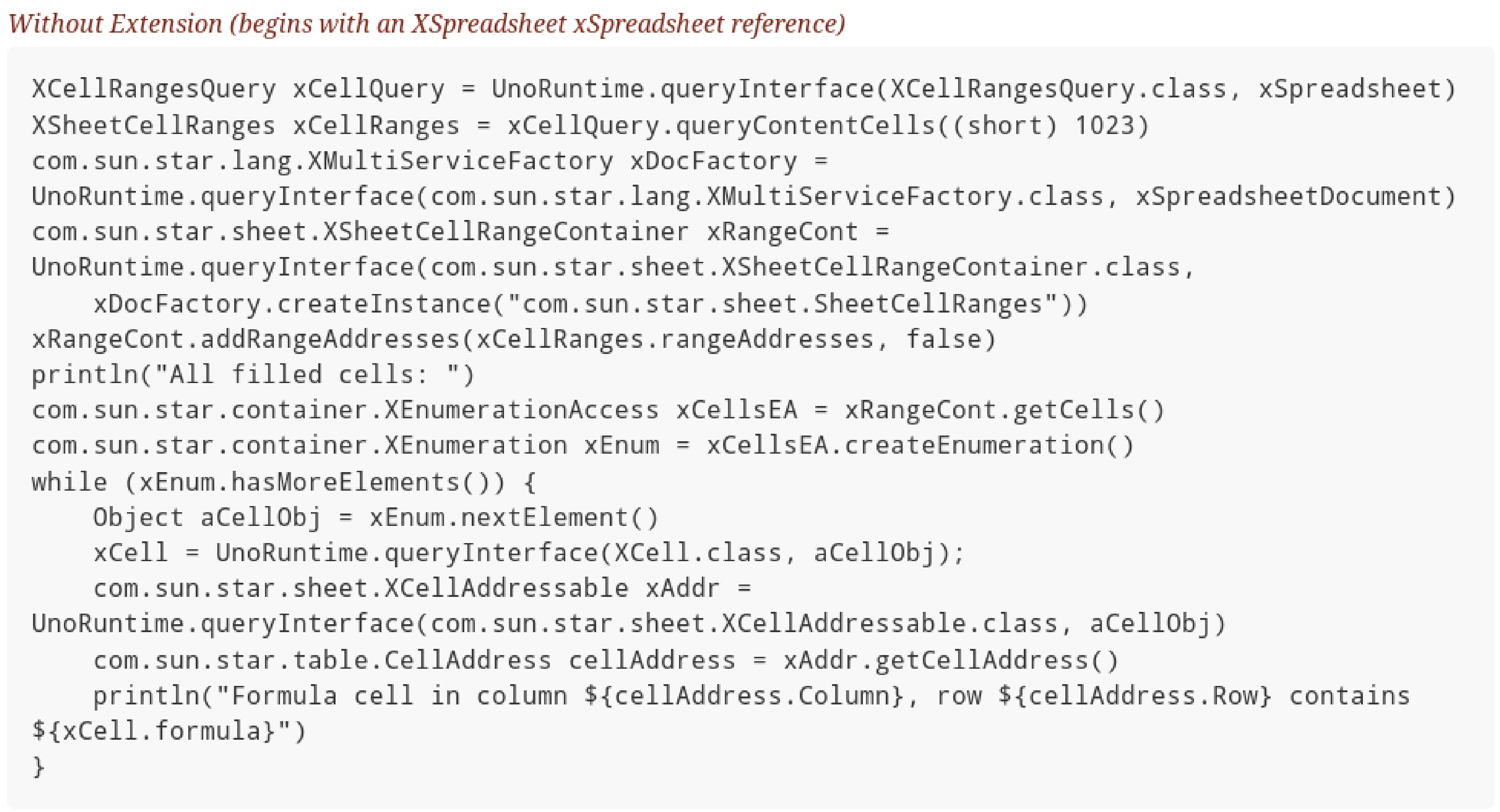
```
XCellRange xCellRange = UnoRuntime.queryInterface(XCellRange.class, xSpreadsheet)
xCell = xCellRange.getCellByPosition(2,2)
XText xCellText = UnoRuntime.queryInterface(XText.class, xCell)
xCellText.setString("Quotation")
```

With Extension

```
xSpreadsheet.setFormulaOfCell(2,2, "Quotation")
```

Groovy UNO Extension

Example: Create a new cell range container, add all cells that are filled, and iterate through them.



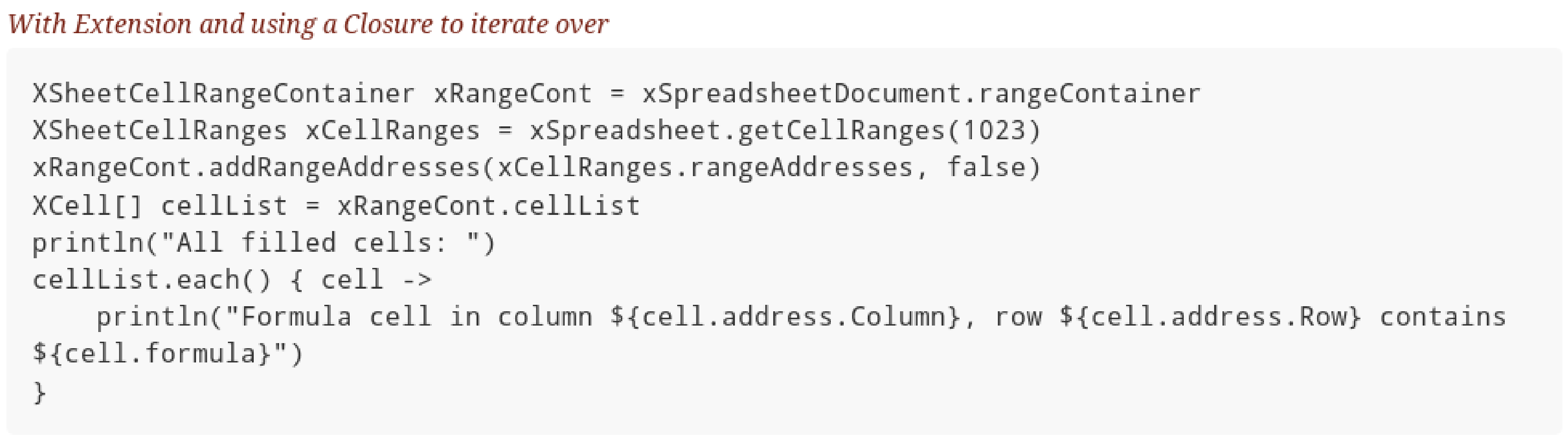
Without Extension (begins with an XSpreadsheet xSpreadsheet reference)

```
XCellRangesQuery xCellQuery = UnoRuntime.queryInterface(XCellRangesQuery.class, xSpreadsheet)
XSheetCellRanges xCellRanges = xCellQuery.queryContentCells((short) 1023)
com.sun.star.lang.XMultiServiceFactory xDocFactory =
UnoRuntime.queryInterface(com.sun.star.lang.XMultiServiceFactory.class, xSpreadsheetDocument)
com.sun.star.sheet.XSheetCellRangeContainer xRangeCont =
UnoRuntime.queryInterface(com.sun.star.sheet.XSheetCellRangeContainer.class,
    xDocFactory.createInstance("com.sun.star.sheet.SheetCellRanges"))
xRangeCont.addRangeAddresses(xCellRanges.rangeAddresses, false)
println("All filled cells: ")
com.sun.star.container.XEnumerationAccess xCellsEA = xRangeCont.getCells()
com.sun.star.container.XEnumeration xEnum = xCellsEA.createEnumeration()
while (xEnum.hasMoreElements()) {
    Object aCellObj = xEnum.nextElement()
    xCell = UnoRuntime.queryInterface(XCell.class, aCellObj);
    com.sun.star.sheet.XCellAddressable xAddr =
UnoRuntime.queryInterface(com.sun.star.sheet.XCellAddressable.class, aCellObj)
    com.sun.star.table.CellAddress cellAddress = xAddr.getCellAddress()
    println("Formula cell in column ${cellAddress.Column}, row ${cellAddress.Row} contains
${xCell.formula}")
}
```

Code Builders, LLC
Business Software Solutions

Groovy UNO Extension

Example: Create a new cell range container, add all cells that are filled, and iterate through them.



With Extension and using a Closure to iterate over

```
XSheetCellRangeContainer xRangeCont = xSpreadsheetDocument.rangeContainer
XSheetCellRanges xCellRanges = xSpreadsheet.getCellRanges(1023)
xRangeCont.addRangeAddresses(xCellRanges.rangeAddresses, false)
XCell[] cellList = xRangeCont.cellList
println("All filled cells: ")
cellList.each() { cell ->
    println("Formula cell in column ${cell.address.Column}, row ${cell.address.Row} contains
${cell.formula}")
}
```

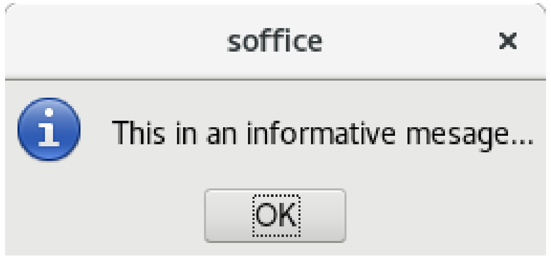
getRangeContainer() method added to XSpreadsheetDocument

getCellRanges() method added to XSpreadsheet

Groovy UNO Extension

MessageBox with Default Title

(without GUNO Extension)


soffice

This in an informative mesage...

Figure 1. Info Box

Without Extension (begins with an XcomponentContext xContext reference)

```
XMultiComponentFactory xMCF = xContext.getServiceManager()
XDesktop xDesktop = xMCF.createInstanceWithContext("com.sun.star.frame.Desktop", xContext)
XFrame xFrame = xDesktop.getCurrentFrame()
Object oToolkit = xMCF.createInstanceWithContext("com.sun.star.awt.Toolkit", xContext)
XMessageBoxFactory xMessageBoxFactory = UnoRuntime.queryInterface(XMessageBoxFactory.class,
oToolkit)
XWindow xWindow = xFrame.getContainerWindow()
XWindowPeer xWindowPeer = UnoRuntime.queryInterface(XWindowPeer.class, xWindow)

XMessageBox xMessageBox = xMessageBoxFactory.createMessageBox(xWindowPeer,
    MessageBoxType.INFOBOX, MessageBoxButtons.BUTTONS_OK,
    "Window Title", "This in an informative mesage...")

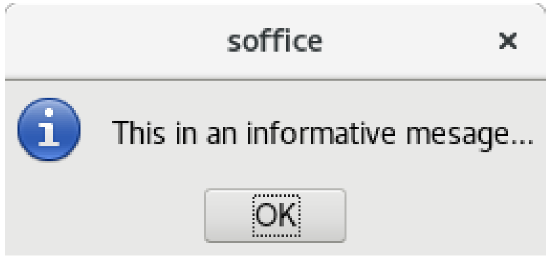
short infoBoxResult = xMessageBox.execute()
```

Groovy UNO Extension

MessageBox with Default Title

(with GUNO Extension)

getMessageBox methods added to XComponentContext



This in an informative mesage...

Figure 1. Info Box

With Extension (Info Box example using default title)

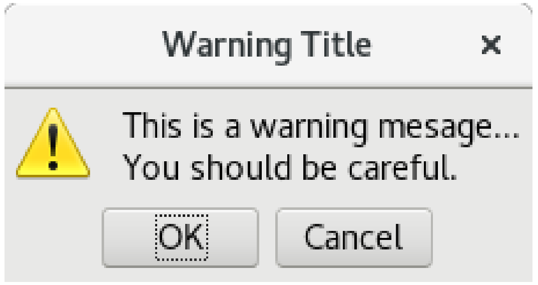
```
XMessageBox infoBox = xContext.getMessageBox(MessageBoxType.INFOBOX,
    MessageBoxButtons.BUTTONS_OK, "This in an informative mesage...")

short infoBoxResult = infoBox.execute()
```

Groovy UNO Extension

MessageBox – Warning Box with a Title

(with GUNO Extension)



This is a warning mesage...
You should be careful.

With Extension (Warning Box example with title and default okay button and a cancel button)

```
String warnMsg = "This is a warning mesage...\nYou should be careful."
Integer warnButtons = MessageBoxButtons.BUTTONS_OK_CANCEL +
MessageBoxButtons.DEFAULT_BUTTON_OK
XMessageBox warningBox = xContext.getMessageBox(MessageBoxType.WARNINGBOX,
    warnButtons, warnMsg, "Warning Title")

short warnBoxResult = warningBox.execute()
```

Buttons are defined as an integer by adding the Enums together.

User selection is returned as a short.

Groovy Swing Builder

Groovy DSL for Swing UI's

Simple SwingBuilder Example

```
import groovy.swing.SwingBuilder
import java.awt.BorderLayout as BL

count = 0
new SwingBuilder().edt {
  frame(title: 'Frame', size: [150, 80], show: true) {
    borderLayout()
    textlabel = label(text: 'Click the button!', constraints: BL.NORTH)
    button(text:'Click Me',
         actionPerformed: {count++; textlabel.text = "Clicked ${count} time(s)."; println
"clicked"}, constraints:BL.SOUTH)
  }
}

// Groovy OpenOffice scripts should always return 0
return 0
```

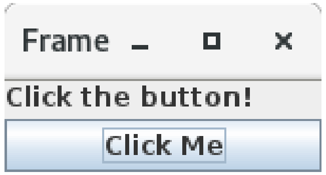


Figure 3. After launch

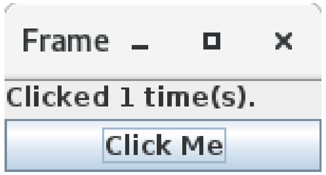
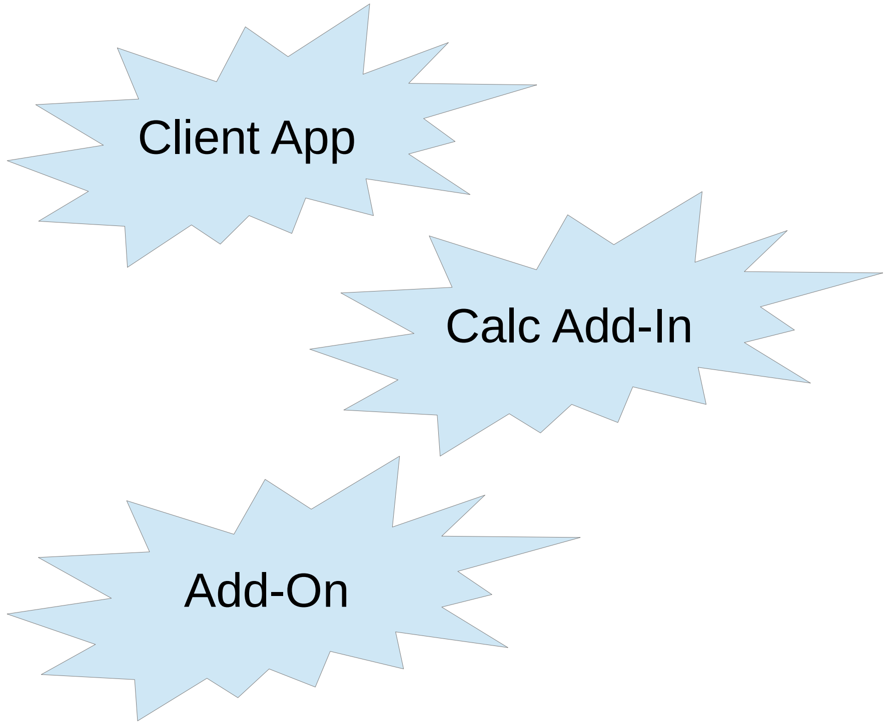


Figure 4. After first button click

FOSDEM 2021

UNO Project Templates

Project Templates



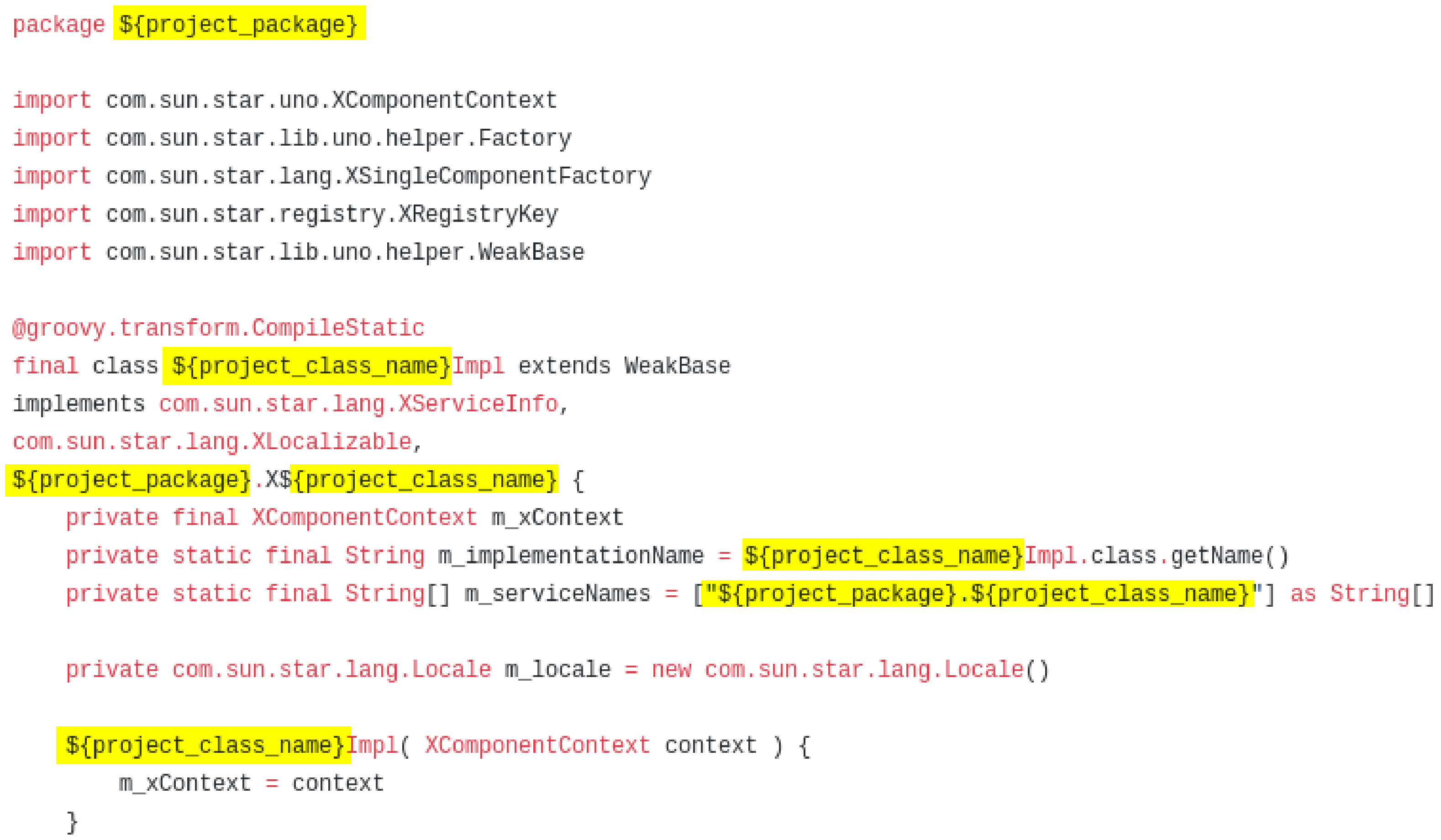
- Client Application: A portable jar file application that can bootstrap the office on any supported OS.
- Calc Add-In: A portable OXT extension for Calc built-in functions.
- Add-On: A portable OXT extension for adding functionality to OpenOffice. (Not yet released)

Project Templates

- Simple to create UNO projects that require minimal setup.
- Projects are IDE independent. IDE only needs to support Gradle and Groovy.
- Derived from projects created by the well established OpenOffice API plugin for the NetBeans IDE.
- Uses the Lazybones project creation tool.

Lazybones

Templates use variables that are replaced during project creation.



```
package ${project_package}

import com.sun.star.uno.XComponentContext
import com.sun.star.lib.uno.helper.Factory
import com.sun.star.lang.XSingleComponentFactory
import com.sun.star.registry.XRegistryKey
import com.sun.star.lib.uno.helper.WeakBase

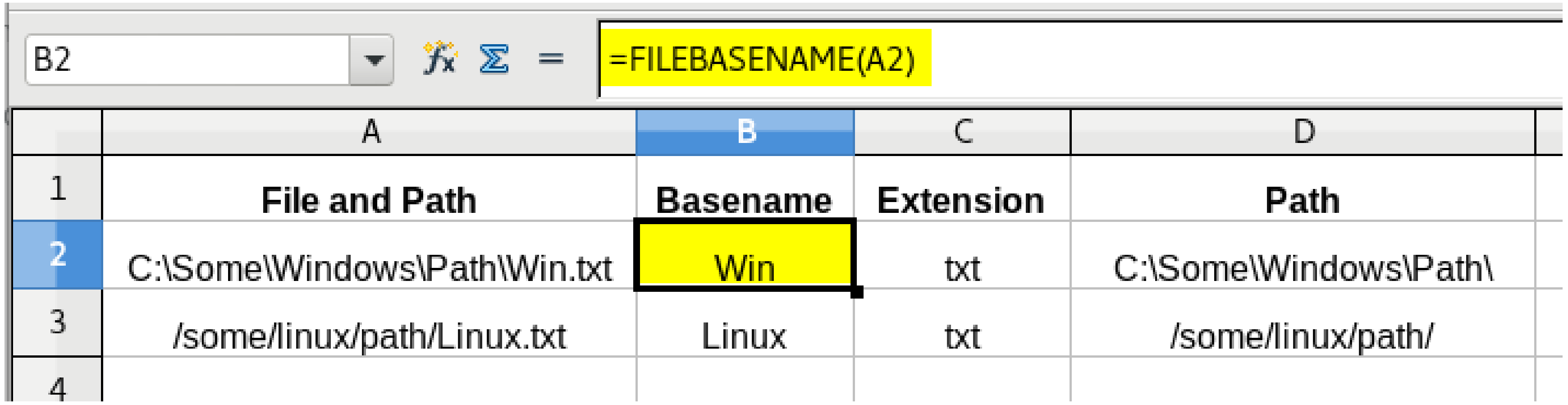
@groovy.transform.CompileStatic
final class ${project_class_name}Impl extends WeakBase
implements com.sun.star.lang.XServiceInfo,
com.sun.star.lang.XLocalizable,
${project_package}.X${project_class_name} {
    private final XComponentContext m_xContext
    private static final String m_implementationName = ${project_class_name}Impl.class.getName()
    private static final String[] m_serviceNames = ["${project_package}.${project_class_name}"] as String[]

    private com.sun.star.lang.Locale m_locale = new com.sun.star.lang.Locale()

    ${project_class_name}Impl( XComponentContext context ) {
        m_xContext = context
    }
```

Calc Add-In Example

File Utilities Add-In to add functions to breakup a path and filename string into basename, extension, and path using Apache Commons IO FilenameUtils class



B2 =FILEBASENAME(A2)

	A	B	C	D
1	**File and Path**	**Basename**	**Extension**	**Path**
2	C:\Some\Windows\Path\Win.txt	Win	txt	C:\Some\Windows\Path\
3	/some/linux/path/Linux.txt	Linux	txt	/some/linux/path/
4				

Calc Add-In Example

~~~~
$ lazybones create aoo-addin 0.3.0 file-utils-addin
Creating project from template aoo-addin 0.3.0 in 'file-utils-addin'
Define value for 'group' [org.example]: net.codebuilders
Define value for 'artifactId' [file-utils-addin]:
Define value for 'version' [0.1.0]:
Define value for 'package' [net.codebuilders]:
Define value for 'className' [FileUtilsAddin]:

Apache OpenOffice Calc Add-In for Groovy project template
-----------------------------------------------------

You have just created a basic Apache OpenOffice Calc Add-In project.
There is a standard project structure for source code and tests.
Simply add your source files to `src/main/<groovy or java>`, your test cases
to `src/test/<groovy or java>` and then you will be able to build your project
with `./gradlew distZip` and clean with `./gradlew clean`.


## Using the project:
1. Edit the build.gradle file and add any additional dependencies if needed.
2. Edit XFileUtilsAddin.idl for your new functions and parameters.
3. Edit FileUtilsAddinImpl.groovy for your new functions and parameters.
4. Edit CalcAddins.xcu for your new functions and parameters.

## Final Touches:
1. Change the description in description/description_en.txt
2. Add your own 42x42 pixel jpg or png logo in images directory.
3. Add your license or keep the Apache License in registration directory.
4. Edit description.xml for these changes and the Add-In display name.

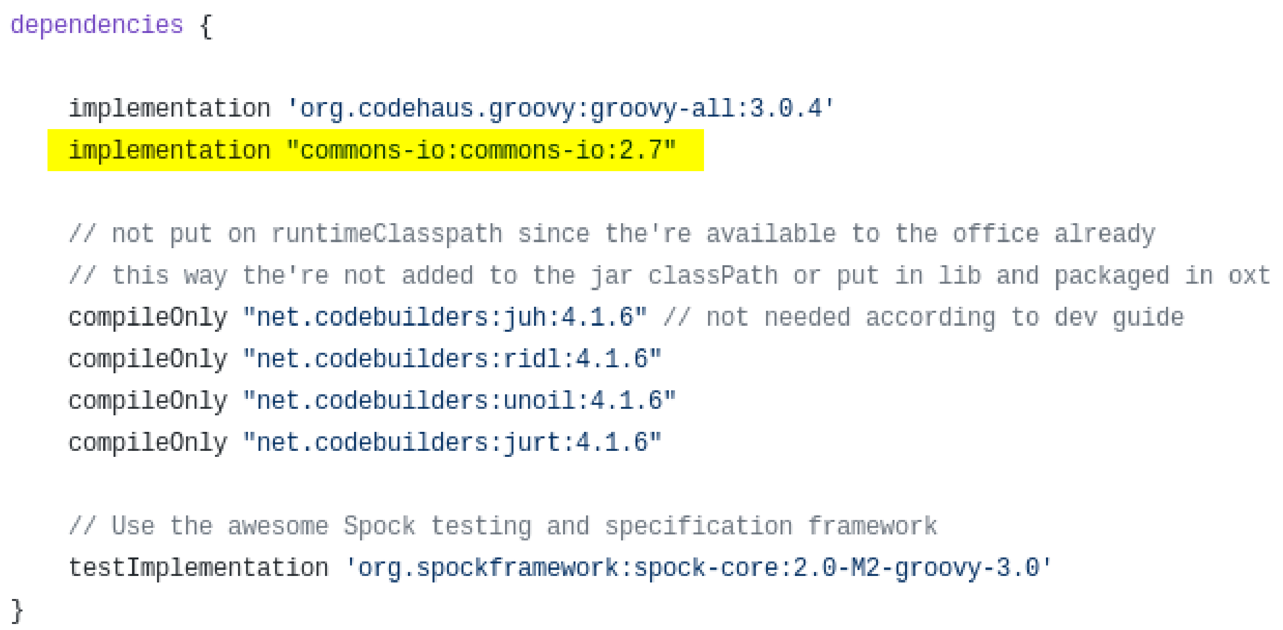
## Building the Extension
- Clean and build the extension with:
```
./gradlew distZip
```
~~~~

~~~~
In this project you get:

* A Gradle build file
* A standard project structure:
```
   <proj>
      |
      +- ant
      |     |
      |     +- ant.properties
      |     |
      |     +- build.xml
      |
      +- src
           |
           +- main
           |     |
           |     +- dist
           |     |     |
           |     |     +- META-INF
           |     |     |     |
           |     |     |     +- manifest.xml
           |     |     |
           |     |     +- description
           |     |     |     |
           |     |     |     +- description_en.txt
           |     |     |
           |     |     +- images
           |     |     |     |
           |     |     |     +- cblogo-42x42.png
           |     |     |
           |     |     +- registration
           |     |     |     |
           |     |     |     +- LICENSE
           |     |     |
           |     |     +- registry
           |     |     |    |
           |     |     |    +- data
           |     |     |          |
           |     |     |          +- org
           |     |     |               |
           |     |     |               +- openoffice
           |     |     |                     |
           |     |     |                     +- Office
           |     |     |                          |
           |     |     |                          +- CalcAddins.xcu
           |     |     |
           |     |     +- description.xml
           |     |
           |     +- groovy
           |     |     |
           |     |     +- <your-package>
           |     |     |    |
           |     |     |    +- CentralRegistrationClass.groovy
           |     |     |    |
           |     |     |    +- FileUtilsAddin.idl
           |     |     |    |
           |     |     |    +- FileUtilsAddinImpl.groovy
           |     |     |    |
           |     |     |    +- XFileUtilsAddin.idl
           |     |
           |     +- resources
           |
           +- test
                |
                +- groovy
~~~~

Calc Add-In Example

build.gradle



```
dependencies {

    implementation 'org.codehaus.groovy:groovy-all:3.0.4'
    implementation "commons-io:commons-io:2.7"

    // not put on runtimeClasspath since the're available to the office already
    // this way the're not added to the jar classPath or put in lib and packaged in oxt
    compileOnly "net.codebuilders:juh:4.1.6" // not needed according to dev guide
    compileOnly "net.codebuilders:ridl:4.1.6"
    compileOnly "net.codebuilders:unoil:4.1.6"
    compileOnly "net.codebuilders:jurt:4.1.6"

    // Use the awesome Spock testing and specification framework
    testImplementation 'org.spockframework:spock-core:2.0-M2-groovy-3.0'
}
```

Calc Add-In Example

XFileUtilsAddin.idl

```
/*
 * XFileUtilsAddin.idl
 *
 * Created on 2020.07.31 - 18:17:08
 *
 */

#ifndef _net_codebuilders_XFileUtilsAddin_
#define _net_codebuilders_XFileUtilsAddin_

#include <com/sun/star/lang/XLocalizable.idl>

#include <com/sun/star/uno/XInterface.idl>

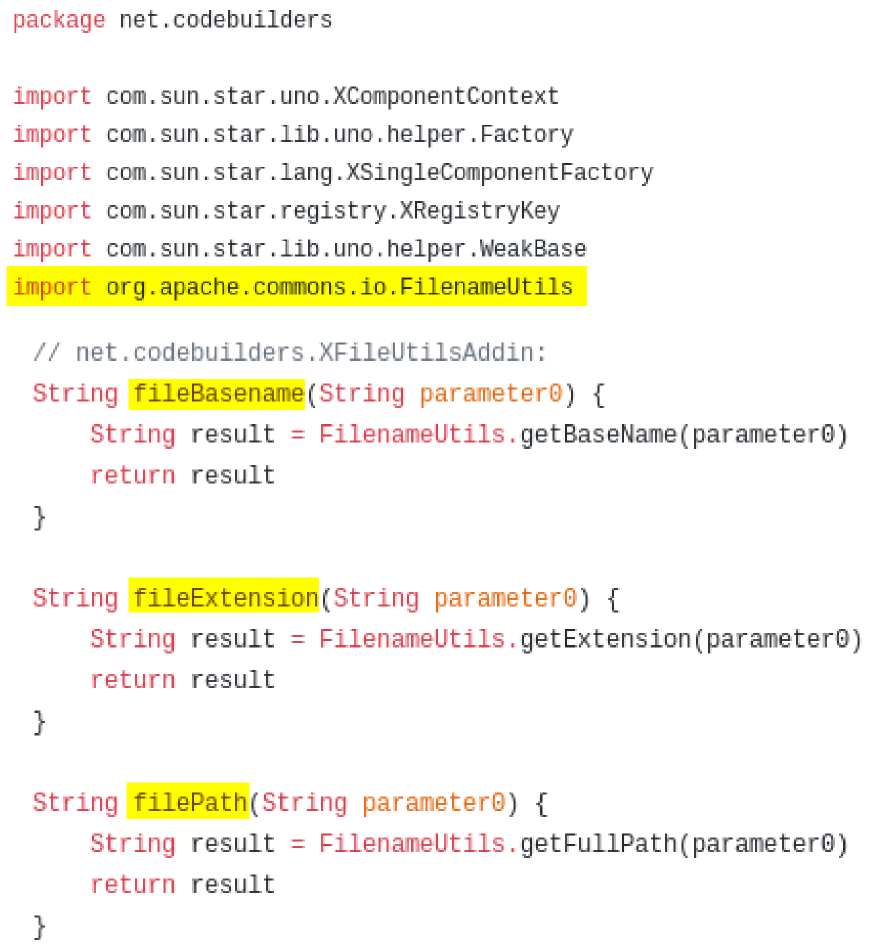
// TODO: fix these for function names and parameters
module net { module codebuilders {
    interface XFileUtilsAddin {
        /// used to set an add-in locale for formatting reasons for example
        [optional] interface ::com::sun::star::lang::XLocalizable;

        string fileBasename([in] string parameter0);
        string fileExtension([in] string parameter0);
        string filePath([in] string parameter0);
    };
}; };

#endif
```

Calc Add-In Example

XFileUtilsAddinImpl.groovy



```
package net.codebuilders

import com.sun.star.uno.XComponentContext
import com.sun.star.lib.uno.helper.Factory
import com.sun.star.lang.XSingleComponentFactory
import com.sun.star.registry.XRegistryKey
import com.sun.star.lib.uno.helper.WeakBase
import org.apache.commons.io.FilenameUtils

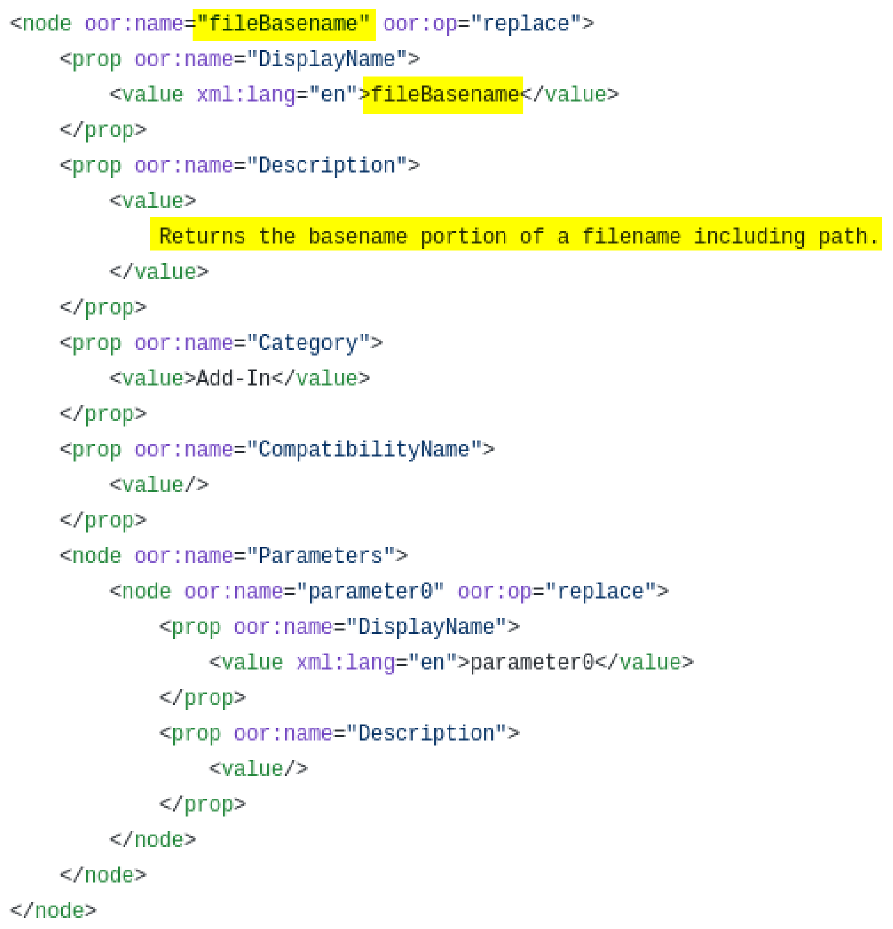
 // net.codebuilders.XFileUtilsAddin:
 String fileBasename(String parameter0) {
     String result = FilenameUtils.getBaseName(parameter0)
     return result
 }

 String fileExtension(String parameter0) {
     String result = FilenameUtils.getExtension(parameter0)
     return result
 }

 String filePath(String parameter0) {
     String result = FilenameUtils.getFullPath(parameter0)
     return result
 }
```

Calc Add-In Example

CalcAddins.xcu



```xml
<node oor:name="fileBasename" oor:op="replace">
    <prop oor:name="DisplayName">
        <value xml:lang="en">fileBasename</value>
    </prop>
    <prop oor:name="Description">
        <value>
            Returns the basename portion of a filename including path.
        </value>
    </prop>
    <prop oor:name="Category">
        <value>Add-In</value>
    </prop>
    <prop oor:name="CompatibilityName">
        <value/>
    </prop>
    <node oor:name="Parameters">
        <node oor:name="parameter0" oor:op="replace">
            <prop oor:name="DisplayName">
                <value xml:lang="en">parameter0</value>
            </prop>
            <prop oor:name="Description">
                <value/>
            </prop>
        </node>
    </node>
</node>
```

Calc Add-In Example

description.xml

```
<publisher>
    <name xlink:href="http://codebuilders.net" lang="en">Code Builders, LLC</name>
</publisher>

<!-- release-notes -->

<display-name>
    <name lang="en">File Utilities Calc Addin</name>
</display-name>

<icon>
    <default xlink:href="images/cblogo-42x42.png"/>
    <high-contrast xlink:href="images/cblogo-42x42.png"/>
</icon>
```

description_en.txt

```
File Utilities Calc Addin is an extension to add file utility functions to Apache OpenOffice Calc.
```

Calc Add-In Example

Building the Extension

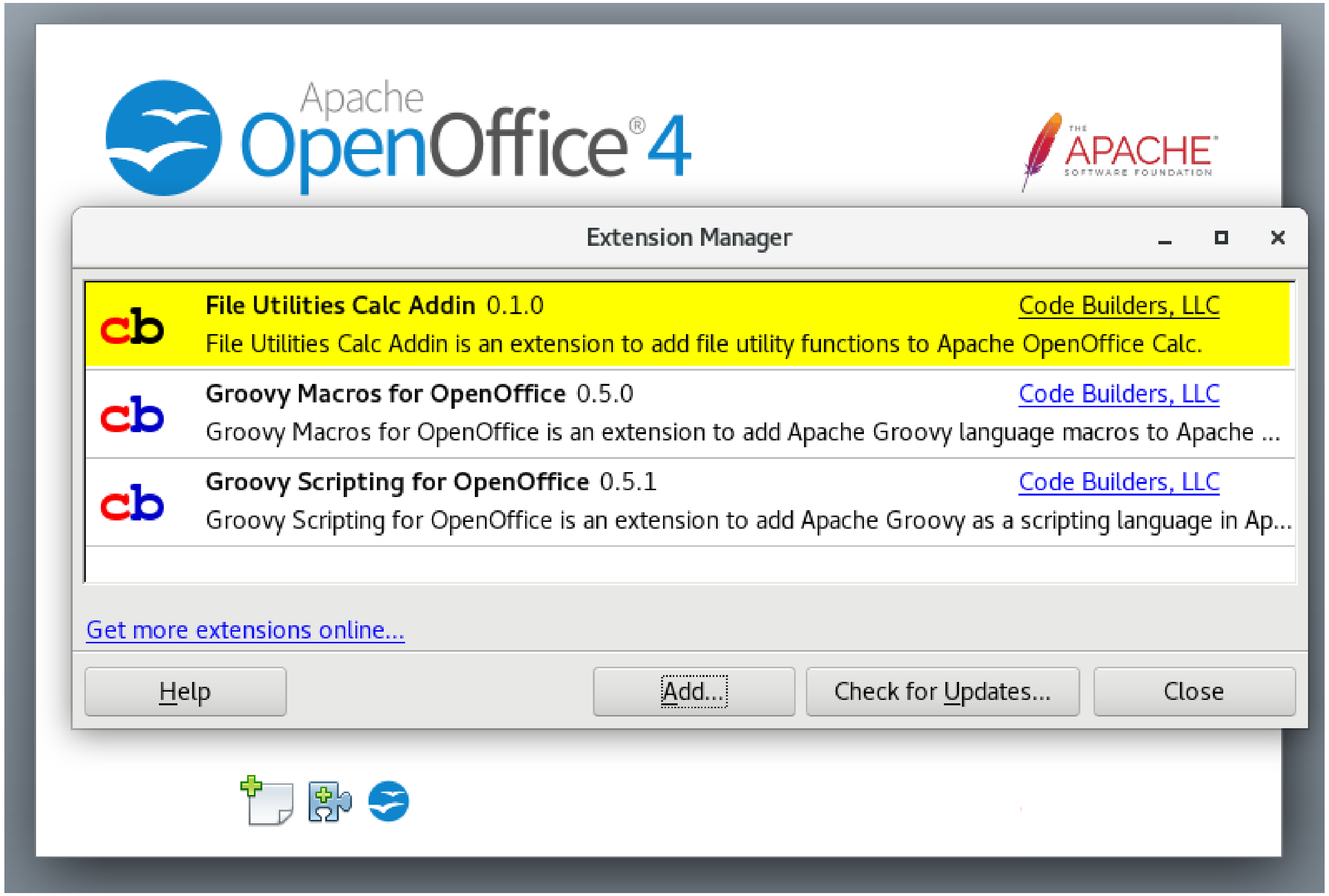
```
$ ./gradlew distZip

> Task :-uno-project-init
[ant:echo] setting up UNO environment ...

> Task :-uno-idl-javamaker
[ant:echo] generating java class files from rdb...

> Task :-uno-idl-result
[ant:echo] build UNO idl files finished

BUILD SUCCESSFUL in 4s
18 actionable tasks: 17 executed, 1 up-to-date
```

Build Directory

```
├── build
│   ├── classes
│   │   ├── groovy
│   │   │   └── main
│   │   │       └── net
│   │   │           └── codebuilders
│   │   │               ├── CentralRegistrationClass.class
│   │   │               └── FileUtilsAddinImpl.class
│   │   └── net
│   │       └── codebuilders
│   │           ├── FileUtilsAddin.class
│   │           └── XFileUtilsAddin.class
│   ├── cpreg
│   ├── distributions
│   │   └── FileUtilsAddin-0.1.0.oxt
│   ├── generated
│   │   └── sources
│   │       └── annotationProcessor
│   │           └── groovy
│   │               └── main
│   ├── idl
│   │   ├── idl_list.properties
│   │   ├── rdb
│   │   │   └── types.rdb
│   │   └── urd
│   │       ├── FileUtilsAddin.urd
│   │       └── XFileUtilsAddin.urd
│   ├── idlc.compile
│   ├── img
│   ├── libs
│   │   ├── FileUtilsAddin_IDL_types.jar
│   │   └── FileUtilsAddin.jar
│   ├── MANIFEST.MF
│   └── tmp
│       ├── compileGroovy
│       │   └── groovy-java-stubs
│       └── jar
│           └── MANIFEST.MF
```

Calc Add-In Example

Extension Manager



Extension Manager

File Utilities Calc Addin 0.1.0 Code Builders, LLC
File Utilities Calc Addin is an extension to add file utility functions to Apache OpenOffice Calc.

Groovy Macros for OpenOffice 0.5.0 Code Builders, LLC
Groovy Macros for OpenOffice is an extension to add Apache Groovy language macros to Apache ...

Groovy Scripting for OpenOffice 0.5.1 Code Builders, LLC
Groovy Scripting for OpenOffice is an extension to add Apache Groovy as a scripting language in Ap...

Groovy Macros in OpenOffice

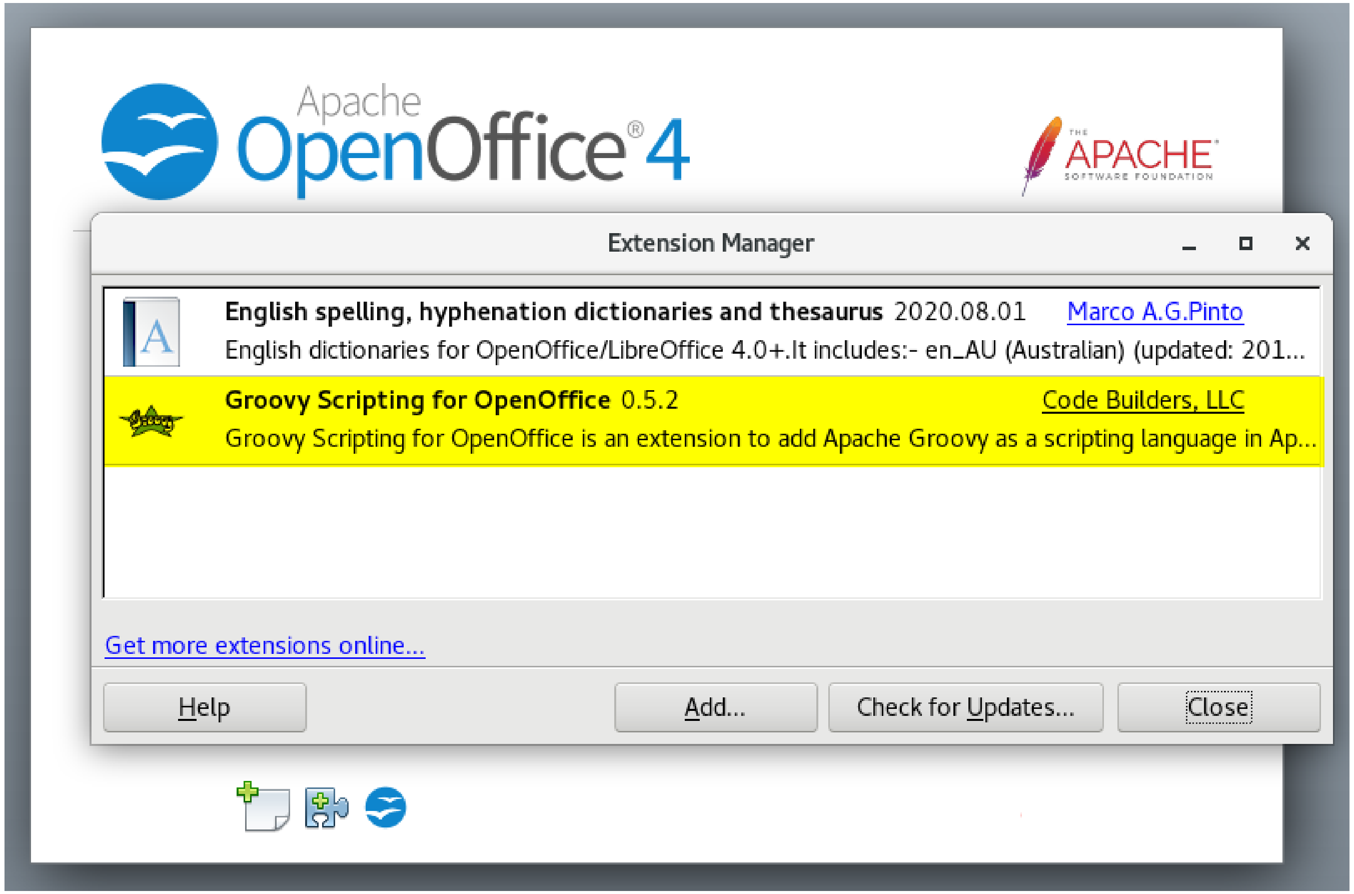
Groovy Scripting Extension

+

- Uses the Scripting Framework to add Groovy as a Macro Language.
- New macros start with runnable code that has URL's to help content.

Groovy Scripting Extension

Install OXT using Extension Manager

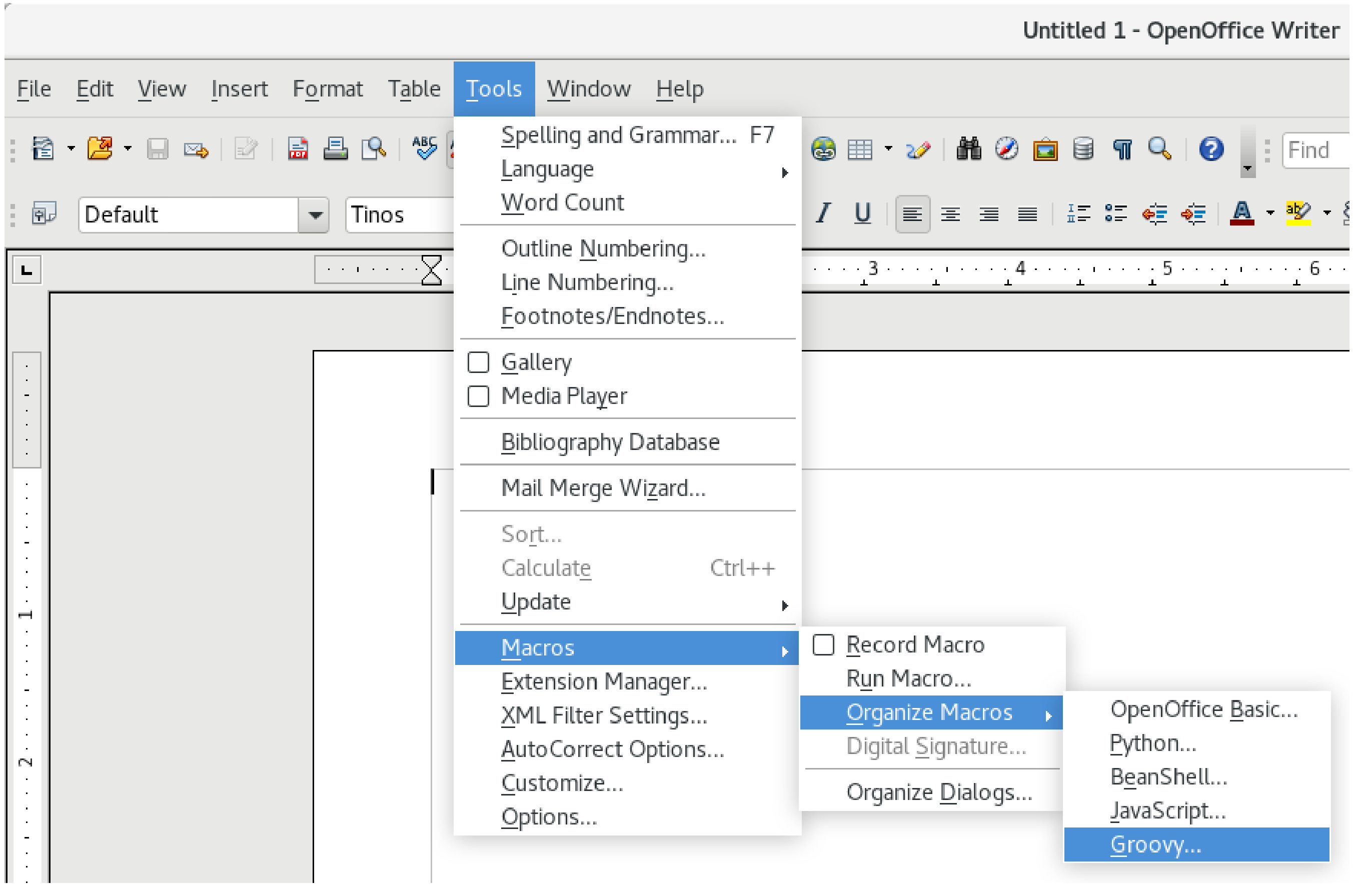


English spelling, hyphenation dictionaries and thesaurus 2020.08.01 Marco A.G.Pinto
English dictionaries for OpenOffice/LibreOffice 4.0+.It includes:- en_AU (Australian) (updated: 201...

Groovy Scripting for OpenOffice 0.5.2 Code Builders, LLC
Groovy Scripting for OpenOffice is an extension to add Apache Groovy as a scripting language in Ap...

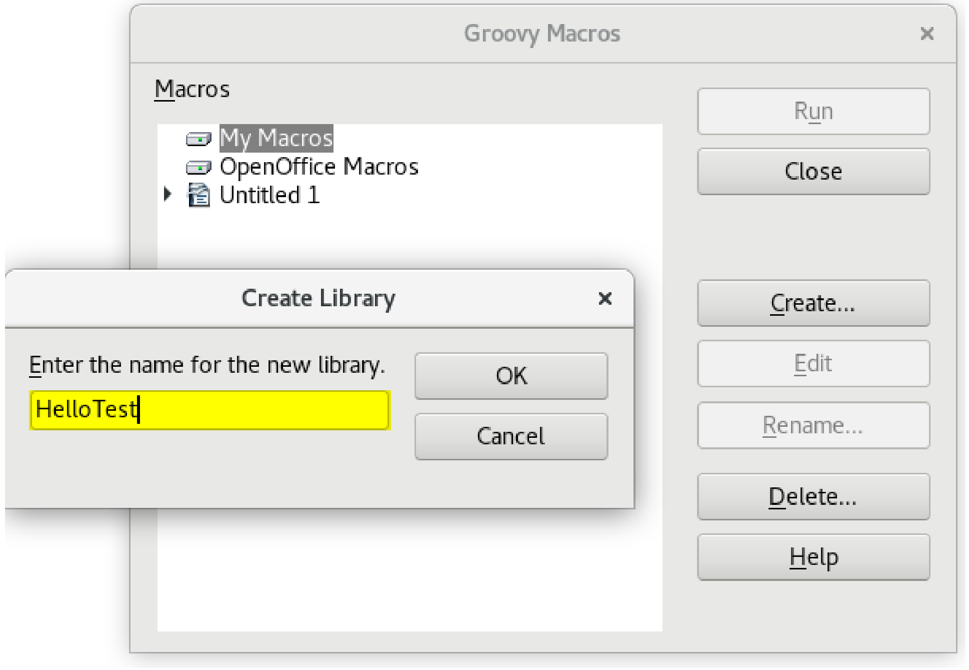
Groovy Scripting Extension

Restart OpenOffice



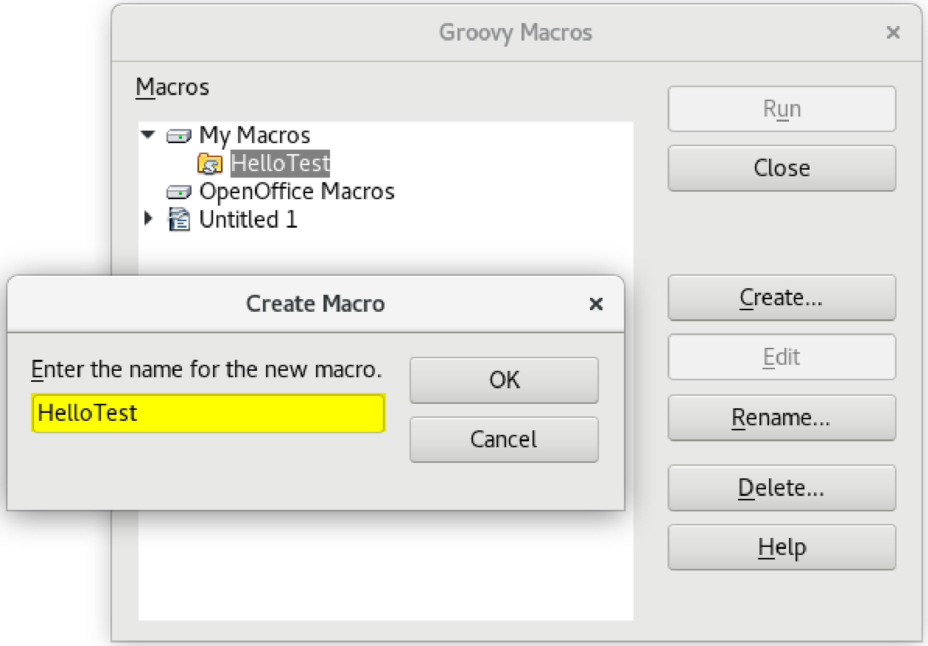
Groovy Scripting Extension

Create a new Groovy Library and Macro



Enter the name for the new library.

HelloTest

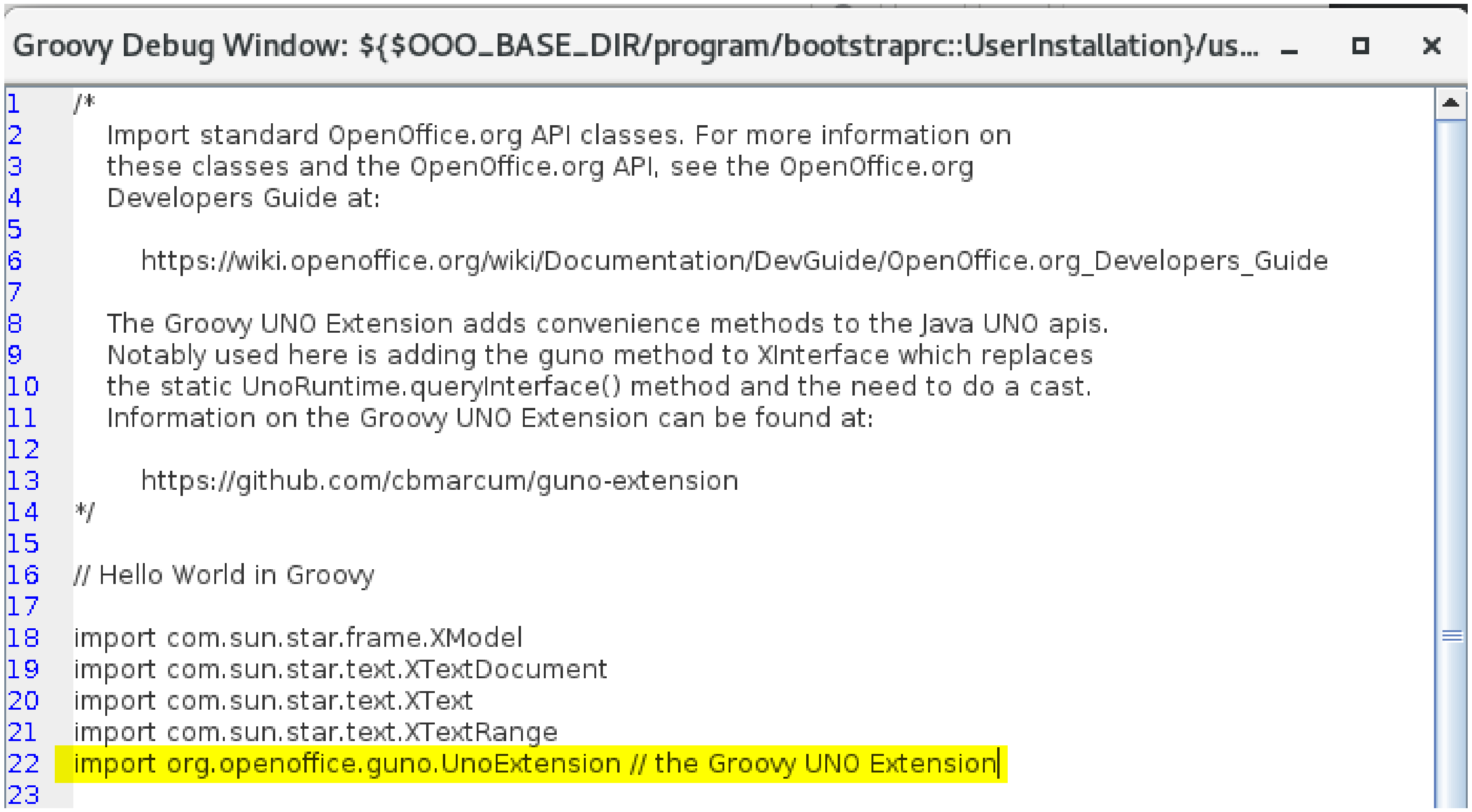


Enter the name for the new macro.

HelloTest

Groovy Scripting Extension

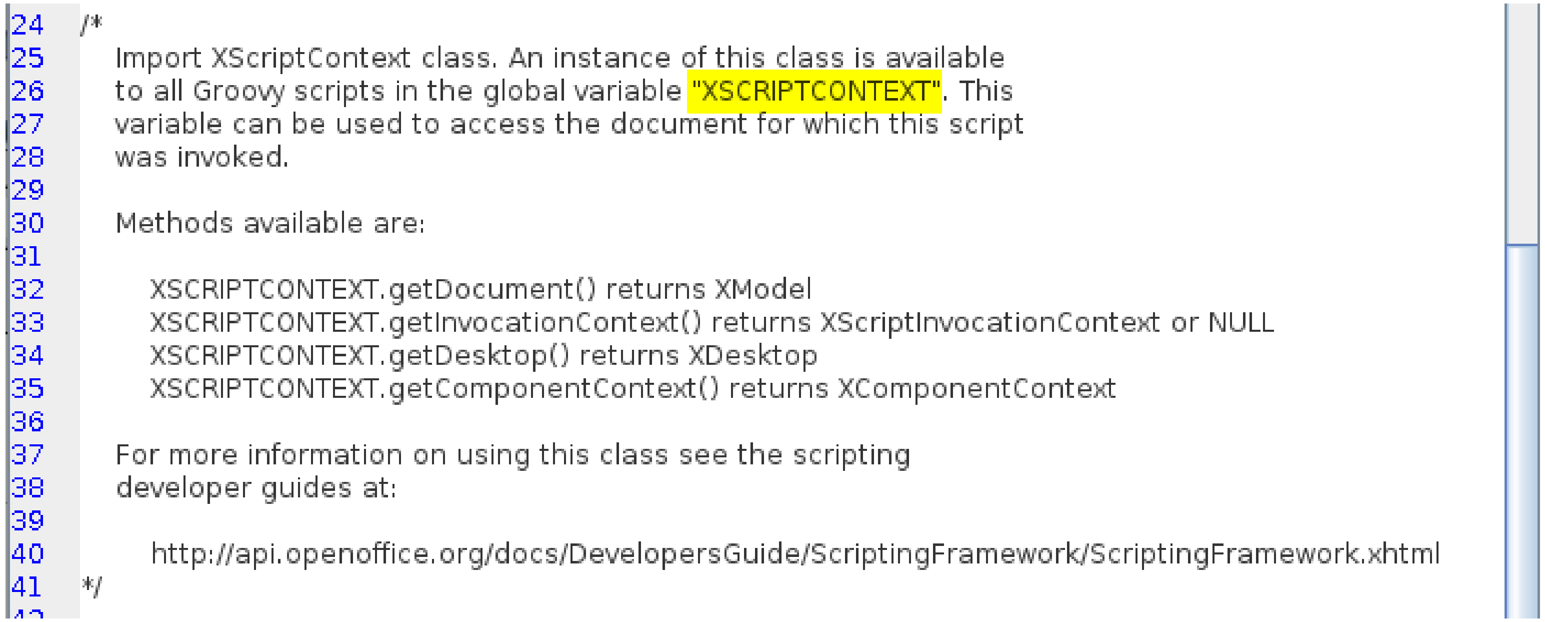
- Includes Groovy UNO Extension


Groovy Debug Window: ${$OOO_BASE_DIR/program/bootstraprc::UserInstallation}/us...

```
/*
   Import standard OpenOffice.org API classes. For more information on
   these classes and the OpenOffice.org API, see the OpenOffice.org
   Developers Guide at:

      https://wiki.openoffice.org/wiki/Documentation/DevGuide/OpenOffice.org_Developers_Guide

   The Groovy UNO Extension adds convenience methods to the Java UNO apis.
   Notably used here is adding the guno method to XInterface which replaces
   the static UnoRuntime.queryInterface() method and the need to do a cast.
   Information on the Groovy UNO Extension can be found at:

      https://github.com/cbmarcum/guno-extension
*/

// Hello World in Groovy

import com.sun.star.frame.XModel
import com.sun.star.text.XTextDocument
import com.sun.star.text.XText
import com.sun.star.text.XTextRange
import org.openoffice.guno.UnoExtension // the Groovy UNO Extension
```

Groovy Scripting Extension

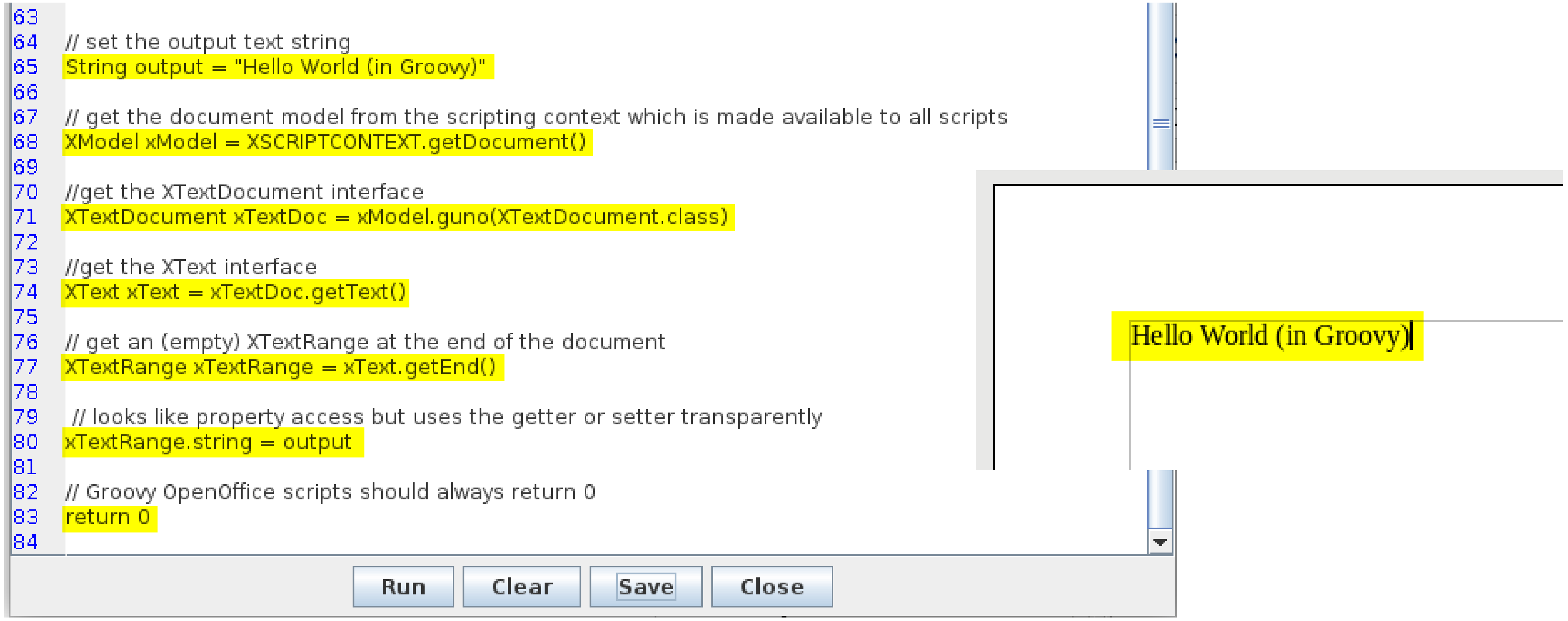
- No need to Bootstrap the Office
- XScriptContext is automatically available



```
/*
    Import XScriptContext class. An instance of this class is available
    to all Groovy scripts in the global variable "XSCRIPTCONTEXT". This
    variable can be used to access the document for which this script
    was invoked.

    Methods available are:

        XSCRIPTCONTEXT.getDocument() returns XModel
        XSCRIPTCONTEXT.getInvocationContext() returns XScriptInvocationContext or NULL
        XSCRIPTCONTEXT.getDesktop() returns XDesktop
        XSCRIPTCONTEXT.getComponentContext() returns XComponentContext

    For more information on using this class see the scripting
    developer guides at:

        http://api.openoffice.org/docs/DevelopersGuide/ScriptingFramework/ScriptingFramework.xhtml
*/
```

Groovy Scripting Extension

- Get the Model
- Get the TextDocument
- Get the Document's Text
- Get the TextRange
- Set the Text



```
// set the output text string
String output = "Hello World (in Groovy)"

// get the document model from the scripting context which is made available to all scripts
XModel xModel = XSCRIPTCONTEXT.getDocument()

//get the XTextDocument interface
XTextDocument xTextDoc = xModel.guno(XTextDocument.class)

//get the XText interface
XText xText = xTextDoc.getText()

// get an (empty) XTextRange at the end of the document
XTextRange xTextRange = xText.getEnd()

 // looks like property access but uses the getter or setter transparently
xTextRange.string = output

// Groovy OpenOffice scripts should always return 0
return 0
```

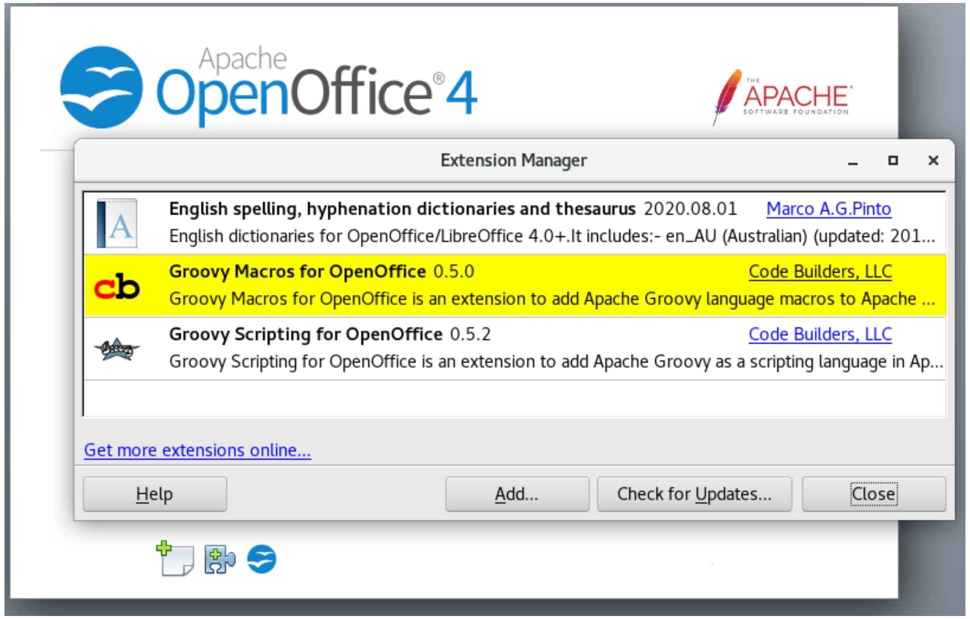
Hello World (in Groovy)

FOSDEM 2021

Sample Macros

Sample Macros Extension

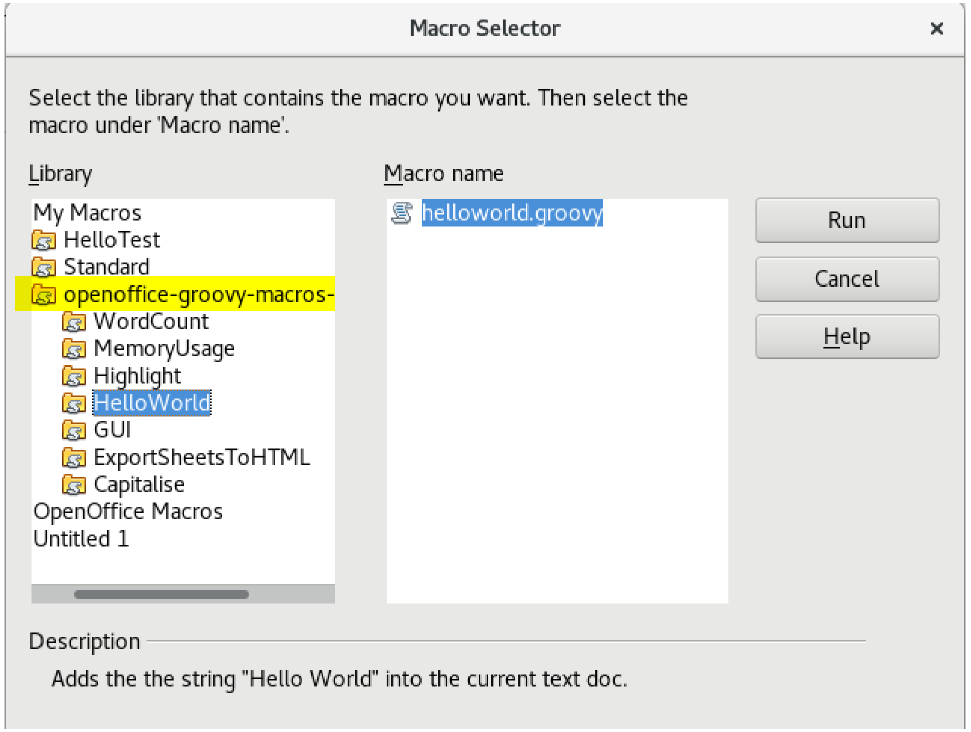
Install OXT using Extension Manager



English spelling, hyphenation dictionaries and thesaurus 2020.08.01 Marco A.G.Pinto
English dictionaries for OpenOffice/LibreOffice 4.0+.It includes:- en_AU (Australian) (updated: 201...

Groovy Macros for OpenOffice 0.5.0 Code Builders, LLC
Groovy Macros for OpenOffice is an extension to add Apache Groovy language macros to Apache ...

Groovy Scripting for OpenOffice 0.5.2 Code Builders, LLC
Groovy Scripting for OpenOffice is an extension to add Apache Groovy as a scripting language in Ap...



Select the library that contains the macro you want. Then select the macro under 'Macro name'.

Library	Macro name
My Macros	helloworld.groovy
HelloTest	
Standard	
openoffice-groovy-macros-	
WordCount	
MemoryUsage	
Highlight	
HelloWorld	
GUI	
ExportSheetsToHTML	
Capitalise	
OpenOffice Macros	
Untitled 1	

Description

Adds the the string "Hello World" into the current text doc.

Summary

+

- OpenOffice is highly customizable through Extensions and Macros.
- Groovy offers unique capabilities to improve UNO programming.
- Efforts underway to improve it even more.
- Help Wanted !!

Links

Apache OpenOffice
https://openoffice.apache.org/
https://www.openoffice.org/
https://wiki.openoffice.org/wiki/Main_Page

Apache Groovy
http://groovy.apache.org/
http://groovy-lang.org/

Groovy Script UNO Client Example
https://github.com/cbmarcum/groovy-script-uno-client

Groovy UNO Extension
https://github.com/cbmarcum/guno-extension

UNO Project Templates
https://github.com/cbmarcum/openoffice-lazybones

Calc FileUtils Add-In Example
https://github.com/cbmarcum/file-utils-addin

Groovy Scripting Extension
https://github.com/cbmarcum/openoffice-groovy

Groovy Example Macros Extension
https://github.com/cbmarcum/openoffice-groovy-macros

Lazybones Project Creation Tool
https://github.com/pledbrook/lazybones

Gradle Build Tool
https://gradle.org/

Contact Info
carl.marcum@codebuilders.net

FOSDEM 2021

Thank You !

THE APACHE SOFTWARE FOUNDATION

Code Builders, LLC

Business Software Solutions

Apache OpenOffice