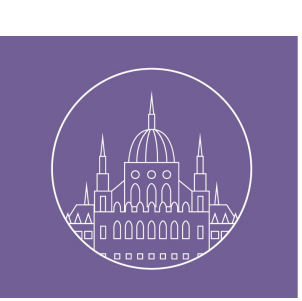
Howto Turn Your Favorite Programming Language into an AOO Macro Language

Rony G. Flatscher

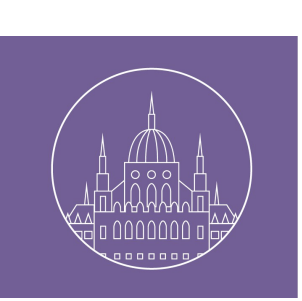
APACHECON EUROPE

CORINTHIA HOTEL
BUDAPEST, HUNGARY
NOVEMBER 17-21, 2014



Overview

- Howto Turn a Scripting Language …
  - Open source software: copy and adapt!
    - “Role-model”: the ooRexx binding
      - Based on the BeanShell binding !
  - Recipe style
    - Please ask questions during the presentation!
- Improving usability for your language
  - Take direct advantage from the ooRexx support!
- Roundup



AOO Programming Languages

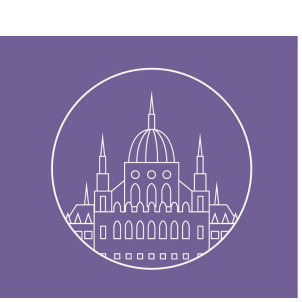
- Programming languages
  - C++ (*queryInterface*)
  - Java (*queryInterface*)
  - Basic (implicit *queryInterface*)
  - Python (implicit *queryInterface*)
- Java-based scripting framework languages
  - BeanShell (*queryInterface*)
  - JavaScript (*queryInterface*)
  - ooRexx (*queryInterface*)
  - …

AOO Scripting Framework, 1

- AOO Java based scripting framework
  - Introduced with OOo 2.0
  - Added the Java implemented scripting engines
    - JavaScript
    - BeanShell (“interpreted Java”)
- To exploit
  - Need for interfacing scripting language with Java
    - Any BSF and JSR-223 supported languages
  - Create a binding in Java, package as oxt

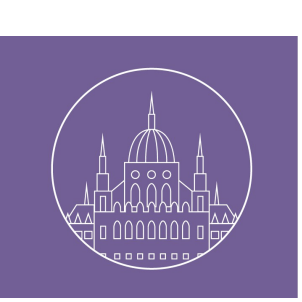
AOO Scripting Framework, 2

- Scripts
  - Stored with document, in user or shared space
  - Invocable using the XScript interface
  - URI-style addressing for cross-invocation
  - Interactive invocation using “Tools . Macros”
- Opensource allows for freely
  - Copying an existing solution and
  - Adapting the copy according to specific needs!



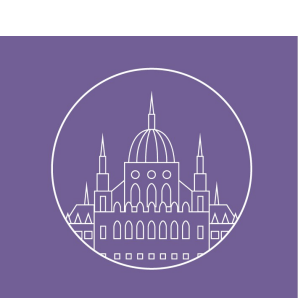
Use Case “ooRexx”, 1

- ooRexx (“Open Object Rexx”, http://www.ooRexx.org)
  - An easy to learn and easy to use dynamically typed scripting language
  - Originally created by IBM, now opensource
    - Based on the then popular IBM “Rexx” scripting language introduced in the 80’ies (e.g. Amiga, OS/2, …), strategic procedural language (“SAA”)
  - Interpreter
  - Object-oriented
    - Influenced by Smalltalk concepts
  - Implemented in C++



Use Case “ooRexx”, 2

- BSF4ooRexx (http://sourceforge.net/projects/bsf4oorexx/)
  - Apache “Bean Scripting Framework (BSF)” engine
    - Originated as opensource software from IBM
    - Bridges scripting languages with Java
      - Originally allow them to be used for JSPs
  - Bridges ooRexx and Java
    - Includes AOO scripting engine support
  - Includes support for AOO
    - Helping programmers by exploiting UNO reflection
    - Installing ooRexx as a macro extension to AOO

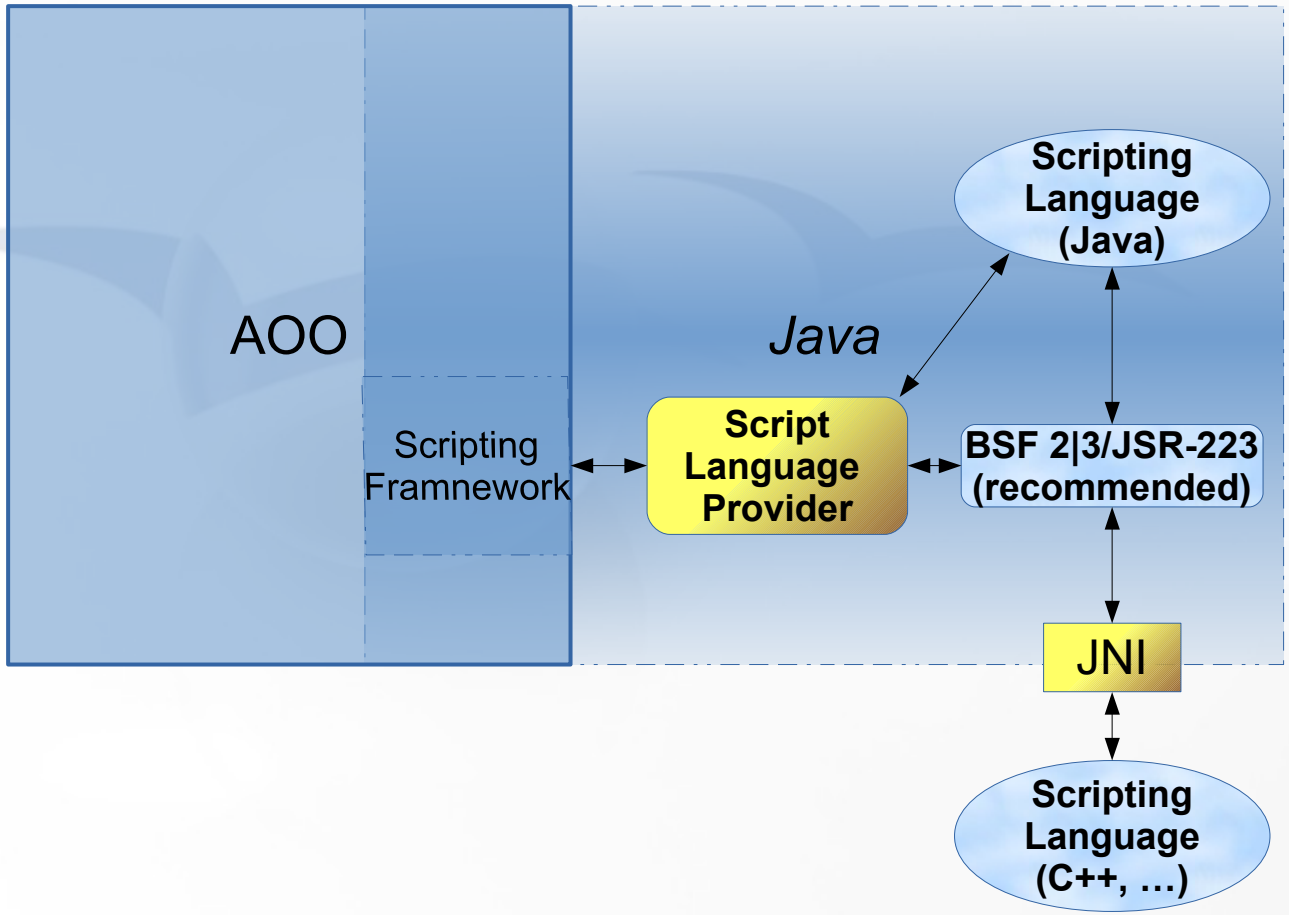


Use Case “ooRexx”, 3

- Presentations based on this infrastructure
  - 2010-09 (10 years OOo, Hungary)
    - “BNF4OOo - An Application Using Writer as a GUI for Creating and Maintaining [e]BNFs”
    - “UNO API Info, Creating Cross-linked Ooo-API-Documentation on the Fly”
  - 2012-11 (ApacheCon Europe, Germany)
    - “Scripting Apache OpenOffice”
    - You may want to use the AOO snippets as examples to transcribe script/macro programs to your scripting language of choice!
      - See links at the end of this presentation!

Apache OpenOffice™

AOO Scripting Framework, 3



Howto/Recipe, 1

- Locate a Java binding for your programming language
  - JSR-223/BSF 3.x, i.e. “javax.script”
    - e.g. http://en.wikipedia.org/wiki/List_of_JVM_languages
  - BSF 2.x
    - e.g. http://commons.apache.org/proper/commons-bsf/
    - BSF4ooRexx uses BSF 2.x
- If no Java binding available for your language
  - Create a Java binding, use JSR-223/BSF 3.x
  - If necessary, use JNI (Java Native Interface)

Howto/Recipe, 2

- Study the script framework provider guide

  https://wiki.openoffice.org/wiki/Documentation/DevGuide/Scripting/How_the_Scripting_Framework_Works

- Locate BeanShell's implementation from svn

  https://svn.apache.org/repos/asf/openoffice/trunk/main/scripting/java/com/sun/star/script/framework/provider/beanshell/

- Locate ooRexx' implementation from svn

  http://sourceforge.net/p/bsf4oorexx/code/HEAD/tree/trunk/com/sun/star/script/framework/provider/oorexx/

- Carry out a diff between both
  - Will show you the areas which you need to adapt with the code to access your programming language

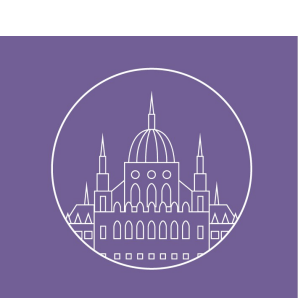
Howto/Recipe, 3

- Carry out a diff between both (cont'd)
  - Work off the BeanShell package provider files
  - Main work in ScriptProviderFor**XXX**.java
  - File template.**XXX** used in GUI (Tools . Macros)
- Setup CLASSPATH for Java
  - Determine the CLASSPATH settings for AOO
    - "*AOO-HOME*/unoinfo java"
  - Add the AOO scripting framework jar in addition!
    - "*AOO-HOME*/classes/ScriptFramework.jar"

Howto/Recipe, 4

- Compile your language provider package
- Create a jar off your language provider
  - This will be used by AOO to dispatch scripts
  - e.g. ScriptProviderForooRexx.jar
- Create an oxt (OpenOffice Extension)
  - Itself a Java archive, hence use the jar tool
  - Include description.xml
    - Defines the package
      - Version, identifier, dependencies, icon, ...

http://sourceforge.net/p/bsf4oorexx/code/HEAD/tree/trunk/bsf4oorexx.dev/ScriptProviderForooRexxPackage/



Howto/Recipe, 5

- Create an oxt (OpenOffice Extension), cont'd
  - Include your language provider jar
    - e.g. ScriptProviderForooRexx.jar
  - Include a META-INF/manifest.**xml**
    - Defines and points to language provider jar

http://sourceforge.net/p/bsf4oorexx/code/HEAD/tree/trunk/bsf4oorexx.dev/ScriptProviderForooRexxPackage/META-INF/

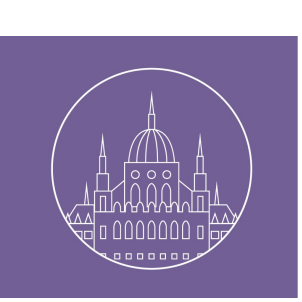
- Deploy/install oxt
  - GUI: "Tools . Extension Manager…"
  - Command line
    - "*AOO-HOME*/unopkg --help"

Howto/Recipe, 6

- Close all instances of AOO
- Restart AOO
  - Will read and cache all installed extensions
- Check GUI
  - Open e.g. word processor (swriter) module
  - "Tools . Macros . Organize Macros . XXX"
    - Select "My Macros", create a macro library
    - Create a new macro (will use "template.XXX")
    - "Edit" and "Run"
- Ready !

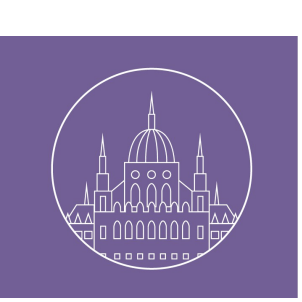
Improving Usability, 1

- UNO reflection
  - There are thousands of UNO classes!
  - All UNO classes are defined with IDL
  - UNO reflection can be used at runtime
    - To document the structure of UNO objects in hand
    - To resolve interfaces of UNO services dynamically
  - However
    - UNO reflection is quite complex!



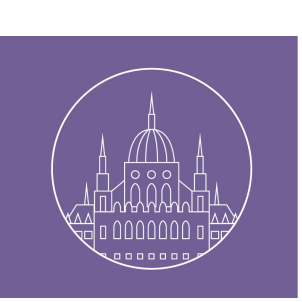
Improving Usability, 2

- UNO reflection (cont'd)
  - Take advantage of the opensource principle
    - No need to reinvent the wheel, save time!
    - Just reuse what exists already, e.g. in BSF4ooRexx!
      - org.oorexx.uno.RgfReflectUNO.java
      - JavaDocs explain its usage



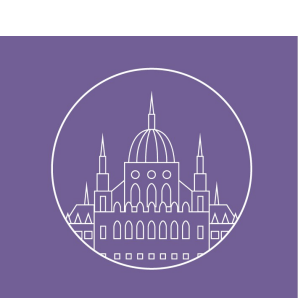
Improving Usability, 2b

- UNO reflection (cont'd)
  - org.oorexx.uno.RgfReflectUNO.java
    - Used for dynamically resolving UNO interfaces for ooRexx scripts in the ooRexx UNO support package UNO.CLS
    - No need for ooRexx programmers to explicitly carry out queryInterface() invocations all the time
    - Used for returning the definitions of all kind of UNO classes
      - Encoded as strings, employing easy to parse delimiters



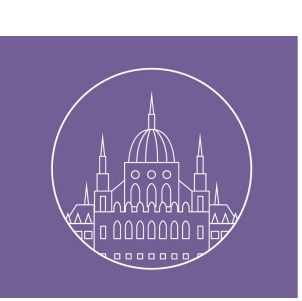
Improving Usability, 3

- UNO reflection (cont'd)
  - org.oorexx.uno.RgfReflectUNO.java
    - Used in the BSF4ooRexx supplied tool "UNO_API_info"
      - Expects a fully qualified name of a UNO class
      - A GUI that renders UNO component definitions
        - As a swriter document with links to the WWW UNO documentation
        - As a PDF file with links to the WWW UNO documentation



Improving Usability, 3b

- UNO reflection (cont'd)
  - A GUI that renders … (cont'd)
    - An ooRexx macro that can be invoked by any other programming language that creates the rendering
      - The macro interface can also process a UNO object supplied as an argument from the other program!
      - Examples are documented and ready to run for Basic, Java, JavaScript ooRexx and Python



Roundup

- AOO Scripting Framework in Java
  - Any programming language with a Java bridge
    - e.g. BSF 2.x, JSR-223/BSF 3.x
  - “Howto” makes it very easy and straightforward
- Opensource
  - Copy working code and adapt!
    - Study the BSF4ooRexx solution, copy and adapt the AOO BeanShell implementation!
  - Do not invent the wheel, reuse existing programs!
    - Use RgfReflectUNO.java and UNO_API_info

Links

- Rexx Language Association (RexxLA), non-profit SIG
  - www.RexxLA.org
- Author's homepage for ooRexx and BSF4ooRexx/AOO related information
  - www.ronyrexx.net
- Author's book on ooRexx
  - www.facultas.at/flatscher

Links to ooRexx

- ooRexx (as of 2014-11-09, version: 4.2.0)
  - An easy to learn and easy to use scripting language
    - Compatible with ("classic") Rexx
    - Developped originally by IBM ("Object REXX")
  - Source code was received by the non-for-profit SIG "Rexx Language Association (http://www.RexxLA.org)"
    - Opensourced as "Open Object Rexx (ooRexx)"
  - Home: http://www.ooRexx.org
  - Downloads: https://sourceforge.net/projects/oorexx/files/oorexx/
  - Brief overview (since opensourcing a lot got added): http://wi.wu.ac.at/rgf/rexx/misc/ecoop06/ECOOP2006_RDL_Workshop_Flatscher_Paper.pdf

Apache OpenOffice™

Links to BSF4ooRexx

- BSF4ooRexx (with built-in AOO support)
  - Allows to use all of Java from ooRexx as if it was an interpreted, typeless and caseless language!
    - "Camouflaging Java as ooRexx" (package "BSF.CLS")
      - All Java classes and Java objects look like ooRexx' ones!
    - Includes specific AOO support (package "UNO.CLS")
  - Developed since 2000 to allow the creation of platform independent Rexx and ooRexx scripts
    - Using Apache's "Bean Scripting Framework (BSF) 2.x", cf. http://commons.apache.org/bsf/
  - Home: https://sourceforge.net/projects/bsf4oorexx/
  - Downloads: https://sourceforge.net/projects/bsf4oorexx/files/GA/

Apache OpenOffice™

Supplement: Scripting AOO Basic UNO Datatypes

Basic UNO Datatype	Java Datatype
UNO_ANY	com.sun.star.uno.Any or java.lang.Object
UNO_VOID	void
UNO_BOOLEAN	boolean
UNO_BYTE (8-bit)	byte
UNO_CHAR (16-bit)	char
UNO_SHORT (16-bit)	short
UNO_UNSIGNED_SHORT (16-bit)	short
UNO_LONG (32-bit)	int
UNO_UNSIGNED_LONG (32-bit)	int
UNO_HYPER (64-bit)	long
UNO_UNSIGNED_HYPER (64-bit)	long
UNO_FLOAT	float
UNO_DOUBLE	double

Supplement: Scripting AOO UNO Types/Classes, 1

- IDL
  - Interface description language
  - Text based definition of UNO types
  - Can be reflected at runtime
- UNO Types/Classes (in alphabetical order)
  - UNO Constants, members:
    - Fields, usually of the same UNO datatype
  - UNO Enum, members:
    - Fields are always of type UNO_LONG (32-Bit integers)

Supplement: Scripting AOO UNO Types/Classes, 1

- IDL
  - Interface description language
  - Text based definition of UNO types
  - Can be reflected at runtime
- UNO Types/Classes (in alphabetical order)
  - UNO Constants, members:
    - Fields, usually of the same UNO datatype
  - UNO Enum, members:
    - Fields are always of type UNO_LONG (32-Bit integers)

Apache OpenOffice™

Supplement: Scripting AOO UNO Types/Classes, 2

- UNO Types/Classes (cont'd)
  - UNO Exception, members:
    - Fields of any datatype
  - UNO Interface, members:
    - UNO Methods
    - UNO Attributes
  - UNO Module, members:
    - Any UNO Type/Class
    - Name of the module(s) are denoted in the fully qualified name of an UNO type, e.g.
      - **com.sun.star.beans**.*PropertyValue*

Supplement: Scripting AOO UNO Types/Classes, 3

- UNO Types/Classes (cont'd)
  - UNO Service, members:
    - UNO Interfaces
    - UNO Services
    - UNO Properties (com.sun.star.beans.*PropertyValue*
      - Regarded as a set (com.sun.star.beans.*XPropertySet*
  - UNO_SINGLETON
  - UNO_STRUCT, members:
    - Fields only
  - UNO_TYPEDEF