"UNO.CLS: An (Open) Object Rexx Module for Universal Network Objects"

2006 International Rexx Symposium
Austin, Texas, U.S.A. (April 2006)

Rony G. Flatscher (Rony.Flatscher@wu-wien.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu-wien.ac.at)
Agenda

• Sources, links
• Outlook given in 2005
• UNO
• UNO.CLS
• Nutshell Examples
• Roundup and Outlook
Sources of figures, examples and hints

- OpenOffice.org papers at http://wi.wu-wien.ac.at/rgf/diplomarbeiten look for the works of Mr. Ahammer (fall 2005) and Mr. Burger (spring 2006) who document OOo with ooRexx, more to come
Outlook given in 2005
"Roundup and Outlook, 2"

• Creating an ooRexx package
  – Simplifying recurring tasks, like establishing a connection with a server
  – Simplifying access to components, e.g. making it easier to manipulate cells of the spreadsheet

• With the advent of OOo 2.0
  – Devise a plug-in for BSF4Rexx, allowing ooRexx to be dispatched from within OOo
  – Will make it possible to use ooRexx wherever StarBasic is used!
Universal Network Objects (UNO), 1

- Component technology used to create the building stones of the OpenOffice.org/StarOffice modules
- CORBA-like
  - IDL: Interface Description Language to define components and types, queryable at runtime
- Defines a communication protocol: urp
- Client/server architecture
- URE: "UNO runtime environment" (fall 2005)
Universal Network Objects (UNO), 2
• urp – UNO remote protocol
• TCP/IP sockets
  – Server can be on another machine (even running another operating system)
Universal Network Objects (UNO), 4
Java Adapter

- Sun bought StarOffice
  - Java interfaces to UNO components
  - Java can be used to implement UNO components
  - ...

Java Adapter
UNO component
OpenOffice.org/StarOffice
Developer's Bird Eye's View, 1

• Set of services to create and maintain documents
• All common functionality of all types of documents is extracted and organized as a set of interfaces
  – E.g. Loading, saving, printing documents
• For each type of document the specific functionality is extracted and organized as a specialized set of interfaces
  • E.g. TextCursors ("write"), Cell-Manipulation ("calc")
OpenOffice.org/StarOffice
Developer's Bird Eye's View, 2

• Client/Server Architecture
  – Employing distributable components ("UNO")
    • Server can run on any computer in the world!
    • Operating system and/or of server as well as that of the client is irrelevant!
  – Communication: urp
    • TCP/IP sockets
    • Named pipes, if available
  – Client can run on the same machine as the server
"Service Managers"
- Supplied by servers
- Can be used to request services from the server
- Returned service allows access to a part of the "office" functionality, E.g.
  - `com.sun.star.frame.Desktop`
  - `com.sun.star.configuration.ConfigurationProvider`
  - `com.sun.star.sdb.DatabaseContext`
Illustration 2.1: Service manager
• "Services"
  – Can be comprehensive
  – Are organized in partitions named
    • "Interfaces" (group of functions/methods) and
    • "structs" (group of related properties only)
  – Depending on the desired task you need to request the appropriate interface, e.g.
    • com.sun.star.view.XPrintable
    • com.sun.star.frame.XStorable
    • com.sun.star.text.XTextDocument
• An example
  – Two services with seven interfaces exposed
    • There are more available
  – "OfficeDocument"
    • Four interfaces
  – "TextDocument"
    • Three interfaces
```java
class Test {
    public static void main (String args[]) {
        com.sun.star.frame.XDesktop xDesktop = null;
        com.sun.star.lang.XMultiComponentFactory xMCF = null;
        try {
            com.sun.star.uno.XComponentContext xContext = null;
            xContext = com.sun.star.comp.helper.Bootstrap.bootstrap();
            xMCF = xContext.getServiceManager();
            if (xMCF != null) {
                System.out.println("Connected to a running office ...");
                Object oDesktop = xMCF.createInstanceWithContext(
                    "com.sun.star.frame.Desktop", xContext);
                xDesktop = (com.sun.star.frame.XDesktop)
                    com.sun.star.uno.UnoRuntime.queryInterface(
                        com.sun.star.frame.XDesktop.class, oDesktop);
                com.sun.star.frame.XComponentLoader xComponentLoader =
                    (com.sun.star.frame.XComponentLoader)
                    com.sun.star.uno.UnoRuntime.queryInterface(
                        com.sun.star.frame.XComponentLoader.class, xDesktop);
                com.sun.star.beans.PropertyValue xEmptyArgs[] = // empty property array
                    new com.sun.star.beans.PropertyValue[0];
                com.sun.star.lang.XComponent xComponent = // text document
                    xComponentLoader.loadComponentFromURL("private:factory/swriter",
                        "_blank", 0, xEmptyArgs);
            } else {
                System.out.println("Not able to create desktop object.");
            }
        } catch (Exception e) {
            e.printStackTrace(System.err);
            System.exit(1);
        }
        System.err.println("Successful run.");
        System.exit(0);
    }
}
```
Creating a Text Document
Java, 2

... cut ...

com.sun.star.frame.XDesktop xDesktop =
    (com.sun.star.frame.XDesktop)
    com.sun.star.uno.UnoRuntime.queryInterface(  
        com.sun.star.frame.XDesktop.class, oDesktop);

com.sun.star.frame.XComponentLoader xComponentLoader =
    (com.sun.star.frame.XComponentLoader)
    com.sun.star.uno.UnoRuntime.queryInterface(  
        com.sun.star.frame.XComponentLoader.class, xDesktop);

com.sun.star.beans.PropertyValue xEmptyArgs[] = // empty property array
    new com.sun.star.beans.PropertyValue[0];

com.sun.star.lang.XComponent xComponent = // text document
    xComponentLoader.loadComponentFromURL("private:factory/swriter",  
        "_blank", 0, xEmptyArgs);

... cut ...
OpenOffice.org 2.x/StarOffice 8
Programming Languages

• OOo version 2.x
  – C++
  – StarBasic
    • Scripting language
  – Java
  – Python
  – New Java-based Scripting Framework
    • BeanShell (interpretable Java)
    • JavaScript (Rhino)
OpenOffice.org/StarOffice and ooRexx?

- No direct support for ooRexx in OOo
- No external Rexx functions available for OOo
- BUT
  - **If** there was a way to bridge ooRexx with Java and then use Java to bridge to UNO, then it would be **possible** to team up OOo with ooRexx!
  - … and there **is** a means available for that: **BSF4Rexx**!
Creating a Text Document

ooRexx with BSF4Rexx

```plaintext
xContext = .bsf~bsf.import("com.sun.star.comp.helper.Bootstrap")~bootstrap
unoRuntime = .bsf~bsf.import("com.sun.star.uno.UnoRuntime")

xMCF = xContext~getServiceManager();
if xMCF<>nil then
do
  say "Connecting to office ..."
oDesktop = xMCF~createInstanceWithContext("com.sun.star.frame.Desktop", xContext)

  xDesktop = unoRuntime~queryInterface(-
      .bsf~bsf.import("com.sun.star.frame.XDesktop"), oDesktop)

  xComponentLoader = unoRuntime~queryInterface(-
      .bsf~bsf.import("com.sun.star.frame.XComponentLoader"), xDesktop)

  PropertyValueClass=.bsf~bsf.import("com.sun.star.beans.PropertyValue")
  xEmptyArgs=bsf.createArray( PropertyValueClass, 0 ) -- empty property array

  xComponent = xComponentLoader~loadComponentFromURL("private:factory/swriter", -
      "_blank", 0, xEmptyArgs);
end
else
  say "Not able to create desktop object."

::requires BSF.CLS -- get Java support (BSF4Rexx)
```
BSF4Rexx, Version 2006-04-10
(17th International Rexx Symposium)

• Install BSF4Rexx
  – Follow the instructions coming with BSF4Rexx
  – Follow the instructions in "readmeBSF4Rexx.txt"

• Configure OOo's Java
  – Make sure OOo is enabled for Java
    • Check "Tools → Options… → Security → OpenOffice.org → Java → Enable"
  – Make sure OOo uses the Java version for which BSF4Rexx was installed!
UNO.CLS, 1

- "OOO.CLS"
  - Created for and introduced at the 16\textsuperscript{th} International Rexx Symposium
  - Eased creation of ooRexx programs that drive OOo

- "UNO.CLS"
  - Generalized support to UNO
  - Drastically enhanced support for UNO and OOo
  - Allows introspection/reflection of UNO objects!
• Environment symbol .UNO
  – Version of UNO.CLS: .uno~version
  – Empty Property array: .uno~noProps
  – Stores UNO classes loaded via the public routine UNO.LoadClass(...)
  
• Preloaded UNO classes:
  – .uno~Any: com.sun.star.uno.Any
  – .uno~PropertyValue: com.sun.star.beans.PropertyValue
  – .uno~RgfReflectUno: org.oorexx.uno.RgfReflectUNO
  – .uno~UnoRuntime: com.sun.star.uno.UnoRuntime
Public Routines

- **UNO.connect([UNO_URL])**
  - Bootstraps UNO and returns the remote object, if `UNO_URL` is given, or local context object else

- **UNO.loadClass(className [, idx])**
  - Creates a proxy class object from the given UNO class name and stores it with `idx` in `.UNO`; if `idx` is omitted the unqualified class name is used

- **UNO.wrap(BSFProxyObject)**
  - Creates and returns a UNO proxy object from a BSF proxy object, adding special UNO behaviour
Public Routines

- UNO.areSame(o1, o2)
  - Returns `.true` if both the UNO object references `o1` and `o2` refer to the same object, `.false` else

- UNO.createDesktop([context])
  - Creates and returns the XDesktop interface object

- UNO.setCell(XSheet, x, y, value)
  - Sets cell with `x/y` co-ordinates (0-based) in Xsheet to `value` using `setFormula()`
Public Routines for Macro Usage

• If ooRexx script is invoked via the OOo scripting framework the following two routines can be used
  
  – **UNO.getScriptContext()**
    • Returns the ScriptContext, a UNO proxy object
  
  – **UNO.getScriptContextVersion()**
    • Returns the ScriptContext version for the Ooo-ooRexx-macro-scripting support
Conversion of Fully Qualified Filenames

• UNO uses URL encoding for filenames, independent of platforms
  – Fully qualified platform dependent file names need therefore be converted
  – `ConvertFromURL(URL)`
    • Returns operating system dependent format of fully qualified file name as encoded in `URL`
  – `ConvertToURL(fileName)`
    • Returns operating system neutral URL encoding of operating system dependent fully qualified `fileName`
URL En-/Decoding

- URLs need to be encoded (and decoded) such that they do not contain illegal characters escaping such characters in the %nn notation
  - `encodeURL(url)`
    - Escapes illegal characters
  - `decodeURL(url)`
    - Translates escaped characters
• Learning the structure of UNO objects can be tedious and time-consuming

• The following routines return all available information in a blank delimited string that can be huge
  – For simple debugging purposes the public routine `ppd(blankDelimitedString)` can be used which will insert CR-LF-TABs for making a SAY output better legible

• `org.oorexx.uno.RgfReflectUNO` methods are used
UNO.CLS, 9

Reflection/Introspection Support, 2

• **UNO.findInterfaceWithMember**(o, name, [bReturnString], [howMany])
  - Looks for a member **name** (case-independently) in the UNO object **o**; if **bReturnString** is .true, then the interface UNOIDL definition is returned, else the queried interface object. If a string is to be returned then up to **howMany** matching interface definitions are encoded, separated by a LF ("0A"x) is returned; a value of -1 will return all interface definitions where a **named** member can be found.

• **UNO.getDefinition**(o)
  - The UNOIDL definition of the UNO object **o** is returned as a blank delimited string.

• **UNO.getInterfaceNamesViaReflection**(o)
  - Returns a blank delimited string of the interface names that the UNO service object **o** implements (reflection, ie. according to UNOIDL)
UNO.CLS, 10
Reflection/Introspection Support, 3

- **UNO.getProperties(o)**
  - Returns a blank delimited string of the property definitions that are defined for the UNO service object o

- **UNO.getServiceNamesViaReflection(o)**
  - Returns a blank delimited string of the interface names that the UNO service object o implements

- **UNO.getTypeName(o)**
  - Returns the type name of the UNO object o

- **UNO.getXTypeProviderTypeNames(o)**
  - Returns a blank delimited string of the interface names that the UNO service object o implements (introspection).
• **UNO queryInterfaceName(o, name)**
  - Returns a fully qualified interface name matching the unqualified, case-insensitive name for the UNO object o

• **UNO queryInterfaceObjectByName(o, name)**
  - Returns the queried interface object for the interface name matching the unqualified, case-insensitive name for the UNO object o

• **UNO queryServiceName(o, name)**
  - Returns a fully qualified service name matching the unqualified, case-insensitive name for the UNO object o
Creating a Text Document ooRexx with UNO.CLS

```
xComponentLoader = UNO.createDesktop() ~XDesktop ~XComponentLoader
xWriterComponent = xComponentLoader~loadComponentFromURL(-
    "private:factory/swriter", "_blank", 0, .UNO~noProps)
say ppd(uno.getDefinition(xWriterComponent))
::requires UNO.CLS -- get UNO support
```
```
ctxt = UNO.connect()  /* connect to server and retrieve the ctxt object */
sm = ctxt~getServiceManager  /* retrieve XMultiComponentFactory */

say "********************"
say "*** Spell Checker ***"
say "********************"
say "Please enter a word: 
parse pull aWord  /* get word to spell check from user */
say
   /* create the LinguServiceManager and the SpellChecker */
lsm = sm~createInstanceWithContext("com.sun.star.linguistic2.LinguServiceManager", ctxt)
sc = lsm~XLinguServiceManager~getSpellChecker

   /* load the required class "com.sun.star.lang.Locale" and set the language to US-english */
call uno.loadClass "com.sun.star.lang.Locale"
 aLocale = .uno~Locale~new("en", "US", ")

isCorrect = sc~isValid(aWord, aLocale, .UNO~noProps)  /* test the word */
say "The word" pp(aWord) "is" iif(isCorrect, ",", "NOT") "correct!"

   /* if the word is not correct submit all alternatives */
sa = sc~spell(aWord, aLocale, .uno~noProps)
if .nil <> sa then
   do
      say; say "Alternatives: 
   do alternative over sa~alternatives
   say " alternative
   end
end

::requires UNO.CLS  -- get UNO support
```

Output:
```
********************
*** Spell Checker ***
********************
Please enter a word:
misterious

The word [misterious] is NOT correct!

Alternatives:
mysterious
```
Roundup and Outlook

- **UNO**
  - CORBA-like component technology
  - URE and OOo build on it
- **Java UNO adapters**
  - Allow using BSF4Rexx
  - ooRexxx can be used to automate/script OOo
- **With OOo v2.0**
  - Java scripting engine framework
  - Using BSF4Rexx and an Ooo-ooRexxx-engine, ooRexxx can be employed as a macro language!