BSF4Rexx: Camouflaging Java

Prof. Rony G. Flatscher

ApacheCon
ASIA 2006

Overview

- Jakarta's Bean Scripting Framework (BSF)
  - Brief History
  - Examples
- Rexx and ooRexx
- BSF4Rexx
  - History and architecture
  - Examples of camouflaging Java as ooRexx
  - Automating/Scripting OpenOffice.org (OOo)
- Roundup

Jakarta's BSF History, 1

- Started 1999 as an IBM research project
  - **Sanjiva Weerawarana**
  - Matthew J. Duftler
  - Sam Ruby
  - Victor Orlikowski
- IBM Developerworks Java project
  - Open source Java framework
  - Package name: "com.ibm.bsf"

Jakarta's BSF History, 2

- Purpose
  - Add scriptability to Java and Java Beans
  - Allow non-Java scripting languages to
    - Be easily deployed by Java programs
    - Access/interact with Java (objects)
- Usage of BSF at IBM
  - Deployed for JSP's in IBM's WebSphere product
  - Allows using any BSF supported scripting language to be used for creating JSP content

Jakarta's BSF History, 3

- 2002 handed over to Apache Jakarta
  - Package name: "**org.apache.bsf**"
- Apache projects employing BSF
  - Some Apache projects that allow using BSF
    - ant
    - Taglib ("BSF" taglib)
    - Xalan (XSLT processor)
  - Search in the documentation for "BSF", also

    → See the last foil of this presentation ("Links")

Jakarta's BSF History, 4

- Some supported BSF scripting languages
  - "javascript": JavaScript (Rhino)
  - "jacl": Jacl (Tcl)
  - "jython": Jython (Python)
  - "netrexx": NetRexx
  - "prolog": Jlog (PROLOG)
  - …
- … but also Java
  - "java"
  - "javaclass"
- … and even XSLT
  - "xslt": XSL transformation

An Example: Java Invoking Scripts

```
import org.apache.bsf.*;// BSF support
import java.io.*;       // exception handling

/** Java program that shows the easyness of deploying scripts with BSF. */
public class TestBSF
{
   /** Running in-line defined scripts. */
   public static void main (String[] args) throws IOException {
      try {
         BSFManager bsfmgr = new BSFManager ();    // get the BSFManager
            // define a Jython (Python) program and invoke it
         String scriptCode = "print 'Jython was here!';";
         bsfmgr.exec("jython", "any debug info", 0, 0, scriptCode);

            // define a Rexx program and invoke it
         scriptCode = "SAY 'Rexx was here!'"; // a Rexx statement
         bsfmgr.exec("rexx", "any debug info", 0, 0, scriptCode);
      }
      catch (BSFException e) {e.printStackTrace();}
   }
}
```

"ScriptedUI.java", 1

- BSF sample program
  - Java program
    - Creates a "java.awt.Frame"
    - Creates "java.awt.Button"s and a "java.awt.Panel"
      - The Java panel object gets stored in the "**BSFRegistry**"
    - Reads the file given at the commandline and uses BSF to dispatch the script
      - The file extension determines the scripting language
  - Script
    - Retrieves the panel object from the "BSFRegistry"
    - Adds a Border layout, creates and places "java.awt" objects

"ScriptedUI.java", 2

```
public class ScriptedUI {

   BSFManager bsfmgr = new BSFManager ();
   public ScriptedUI (String fileName ) {
     Panel p = new Panel ();
     f.add ("Center", p);
     f.add ("North", new Button ("North Button"));
     f.add ("South", new Button ("South Button"));
     bsfmgr.registerBean ("centerPanel", p);

     try {  // exec script engine code to do its thing for this
       String language = BSFManager.getLangFromFilename (fileName);
       FileReader in = new FileReader (fileName);
       String script = IOUtils.getStringFromReader (in);
       bsfmgr.exec (language, fileName, -1, -1, script);

... Cut ...
```

"ScriptedUI.java", 3

```
/* filename "ui.js" (JavaScript)    */

   /* pick up the center panel bean    */
p = bsf.lookupBean ("centerPanel");

   /* set the layout manager to border */
p.setLayout (new java.awt.BorderLayout ());

   /* add a few things */
p.add ("Center", new java.awt.Label ("Middle from JavaScript"));
p.add ("North",  new java.awt.TextField ("north text from JavaScript"));
p.add ("South",  new java.awt.TextField ("south text from JavaScript"));
p.add ("East",   new java.awt.Button ("inner east from JavaScript"));
p.add ("West",   new java.awt.Button ("inner west from JavaScript"));

   /* configure p a bit */
p.setBackground (java.awt.Color.red);

   /* configure the frame that p is in */
f = p.getParent ();
f.setTitle ("Hello from JavaScript (title reset from JavaScript)");
```

"ScriptedUI.java", 4

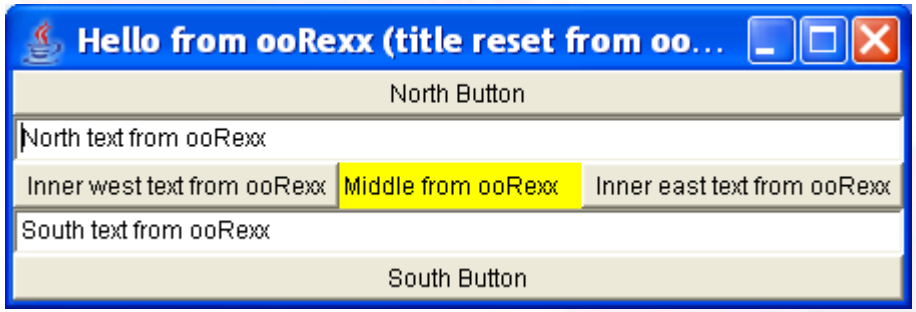
```
java ScriptedUI ui.js
```

"ScriptedUI.java", 5

```
/* filename "ui.rex" (ooRexx)    */

   /* pick up the center panel bean */
p = bsf.lookupBean("centerPanel")

   /* set the layout manager to border */
p~setLayout(.bsf~new("java.awt.BorderLayout"))

   /* add a few things */
p~add("Center", .bsf~new("java.awt.Label",    "Middle from ooRexx"))
p~add("North",  .bsf~new("java.awt.TextField", "North text from ooRexx"))
p~add("South",  .bsf~new("java.awt.TextField", "South text from ooRexx"))
p~add("East",   .bsf~new("java.awt.Button",  "Inner east text from ooRexx"))
p~add("West",   .bsf~new("java.awt.Button",  "Inner west text from ooRexx"))

   /* configure p a bit */
p~setBackground(.bsf~bsf.getStaticValue("java.awt.Color", "yellow"))
p~getParent~setTitle("Hello from ooRexx (title reset from ooRexx)")

::requires BSF.CLS    -- get ooRexx wrapper support for BSF
```

"ScriptedUI.java", 6

```
java ScriptedUI ui.rex
```



Rexx

- Rexx
  - 1979 by Mike F. Cowlishaw (IBM)
    - Goal for devising a "**human centric**" scripting language
      - "Everything is a string"
      - Small language (stable set of builtin functions)
      - Easy syntax (almost like pseudo-code!)
        - *"End-user programming*" becomes possible !
    - To replace the awkward "EXEC II" on mainframes
  - ANSI Rexx standard in 1996
    - Decimal arithmetics served for standardizing "BigNumber"
  - Available for practically all operating systems

IBM's Object REXX

- Object Rexx
  - Developed by IBM as an OO successor to Rexx
    - Work started in 1988 (Simon Nash, IBM England)
    - Final concept created and implemented in the United States (Rick McGuire, IBM US)
  - IBM large customers (SHARE) were in favor
  - Became part of IBM's "OS/2 Warp" in 1997
  - Commercial versions sold for AIX and Windows, mostly to IBM's large customers
  - Still a small, easy to use and powerful language
    - *"End-user programming"* becomes possible

ooRexx, 1

- ooRexx (Open Object Rexx)
  - IBM handed source-code to non-profit SIG "Rexx Language Association" (RexxLA) to opensource it
  - Spring 2005: first release (ooRexx 3.0)
  - Summer 2006: new release (ooRexx 3.1)
  - Future planned version "4.0"
    - Work has already started…

ooRexx, 2

- ooRexx (Open Object Rexx)
  - "Everything is an object"
  - Interpreter, backwardly compatible with Rexx
  - Influenced by Smalltalk
  - Among other things
    - Metaclasses, Reflection, Multiple-inheritance, Unknown-mechanism, One-off objects, ...
  - Explicit message operator: tilde (**~**)
    - Receiving object left of the operator, message name right to it

ooRexx, 3

- Entire class hierarchy!
  - Very few classes
- Language remains small
- Syntax looks almost like pseudo-code

Class Name			Comment
`Object`			Root class, fundamental
	`Alarm`		Send message asynchroneously
	`Array`		Collection class, no predefinition of size or dimension necessary, ordered
	`Class`		Metaclass, fundamental
	`Directory`		Collection class, index is a string, one object per index, no order implied
	`List`		Collection class, ordered
	`Message`		Fundamental class
	`Method`		Fundamental class
	`Monitor`		Monitors messages sent to objects
	`MutableBuffer`		Comparable to Java's `StringBuffer`
	`Queue`		Collection class, ordered
	`Relation`		Collection class, index is any object, multiple objects per index possible, no order implied
		`Bag`	Index and associated object are the same object
	`Stem`		Represents "classic Rexx" stems
	`Stream`		Stream (e.g. file) input/output
	`String`		Object's string values are not mutable
	`Supplier`		Iterator for collection classes
	`Table`		Collection class, index is any object, one object per index, no order implied
		`Set`	Index and associated object are the same object

ooRexx, 4

```
say reverse("aloha")     /* the reverse function returns: ahola    */
say "aloha"~reverse      /* the reverse message returns:  ahola    */
```

- All "classic" Rexx programs run unchanged!
  - Classic Rexx syntax is still allowed
  - OO and classic Rexx syntax can be freely intermixed
- "Classic Rexx" statements get transformed to the object-oriented version "behind the curtain" by the interpreter

BSF4Rexx History, 1

- Started in the Winter semester of 2000/01
  - Since 2001: University of Essen
    - Proof-of-concept successful
    - Full development of a "BSFRexxEngine" started
      - Rexx and Object Rexx could be used with BSF
    - Versions for OS/2 and Windows using IBM's BSF
    - Strong typing necessary
      - Each Java argument was prepended by its type indicator
  - Since 2003: University of Augsburg
    - Jakarta BSF, Linux and Windows versions
    - Rexx became able to start up Java, if necessary

BSF4Rexx History, 2

- Since 2006: Wirtschaftsuniversität Wien ("WU")
  - No type indicators necessary anymore
    - Now matches (weakly typed) ooRexx philosophy
    - Allows all of Java to be camouflaged as ooRexx!
  - All Java objects look like ooRexx objects
    - ooRexx messages can be sent to Java (proxy) objects
    - Datatype conversions for arguments and return values, picking of matching methods etc. done automatically
    - Transparently handling getter and setter semantics
    - …

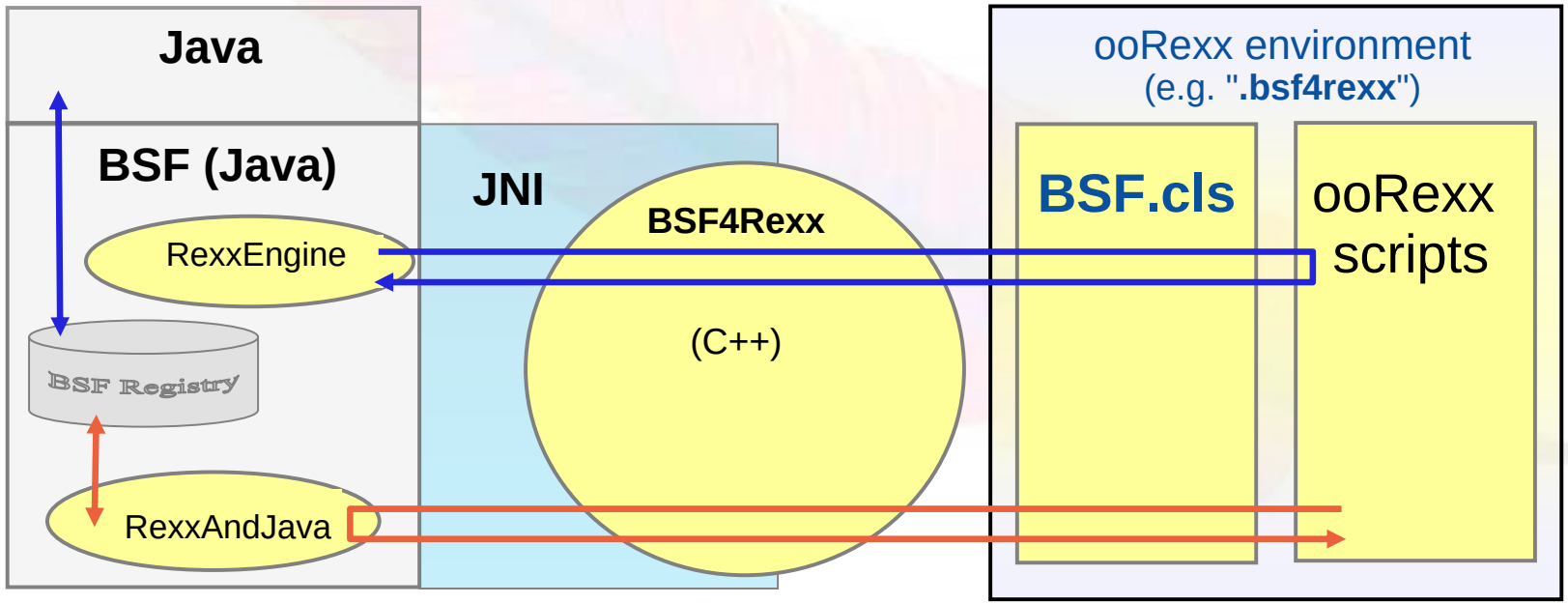
BSF4Rexx, An Example

```
call bsf.cls                                /* get ooRexx suppoert for BSF */

s=bsf.loadClass("java.lang.System")         /* get the Java class object   */
say s~getProperty("java.version")           /* get and output Java version */
```

Yields (maybe):

```
1.5.0_07
```

BSF4Rexx, Architecture, 1



BSF4Rexx, Architecture, 2

- RexxEngine.java
  - Extends "org.apache. bsf.util.BSFEngineImpl"
  - All there is to allow ooRexx to be invoked
- RexxAndJava.java
  - Realizes string based interfaces from ooRexx into Java, could be exploited by other languages
- BSF.CLS
  - ooRexx utility program, camouflaging Java
  - Java objects are turned into ooRexx proxies

BSF4Rexx, Another Example

```
/* creating a two dimensional Java array */

clo=bsf.loadClass("java.lang.String")  /* get the Java class object  */
arr=bsf.createArray(clo, 5, 10)        /* create the Java array      */

   /* from now on, use the Java array as if it was an ooRexx array!  */
arr[1,1]="First element in Java array."       /* place an element    */
arr~put("Last element in Java array.", 5, 10) /* place another one   */

do i over arr         /* loop over elements in array                */
   say i
end

::requires bsf.cls   /* directive: get ooRexx support for BSF       */
```

Yields:

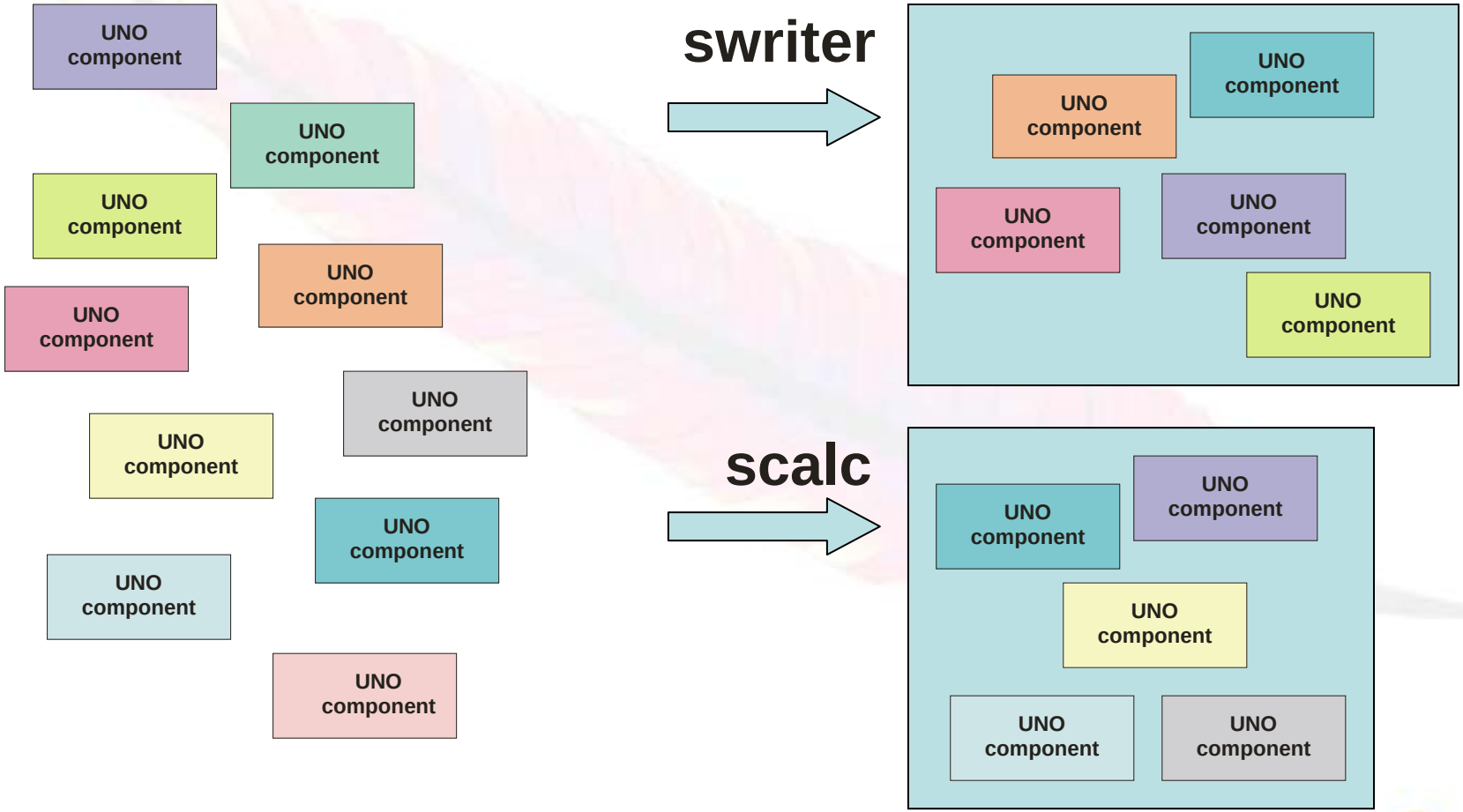
```
First element in Java array.
Last element in Java array.
```

StarOffice/OpenOffice.org

- Office-suite with OO-design to the core
  - Written in C++
  - Multiplatform
  - *Many* filters (eg. Microsoft Office file formats)
- Sun bought StarDivision in 1999
  - Opensource version in 2001 released
  - http://www.**O**pen**O**ffice.**o**rg ("OOo")

→ **Java (proxy) interfaces to all C++ classes!**

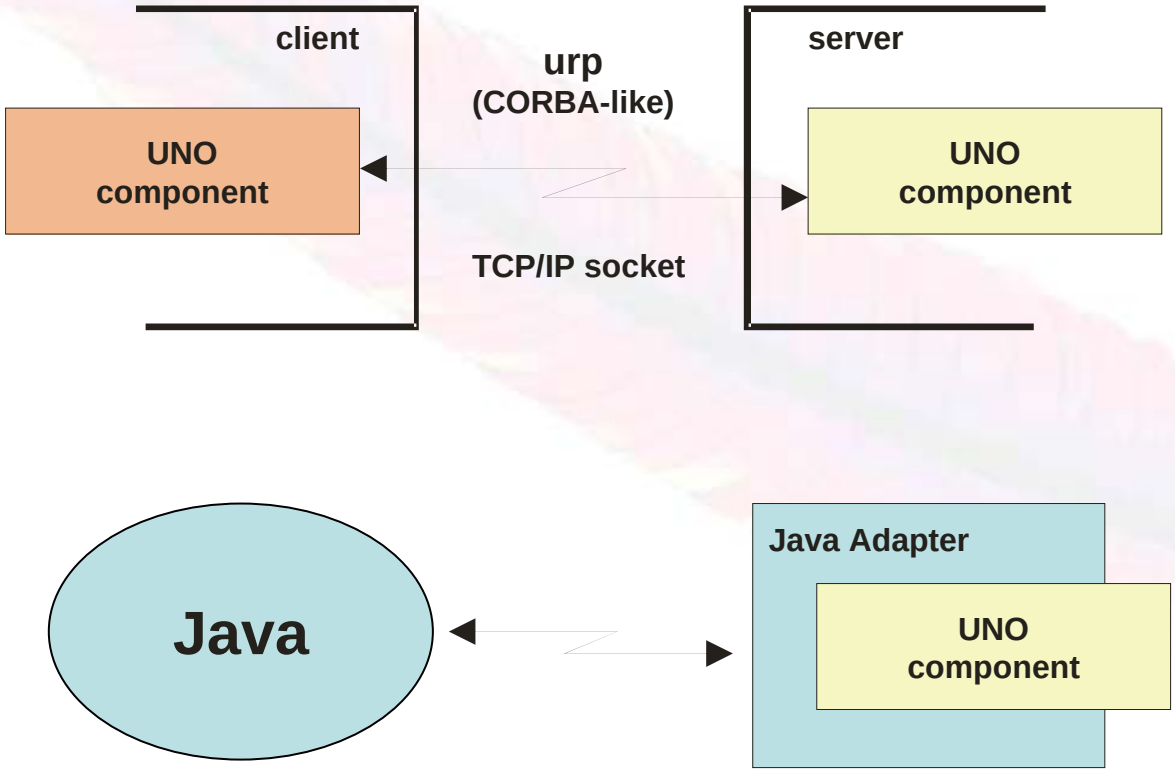
OOo "UNO" Architecture



Universal Network Objects

- 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
  - TCP/IP
- Client/server architecture
  - Server can be on another computer running another operating system!

UNO Interaction

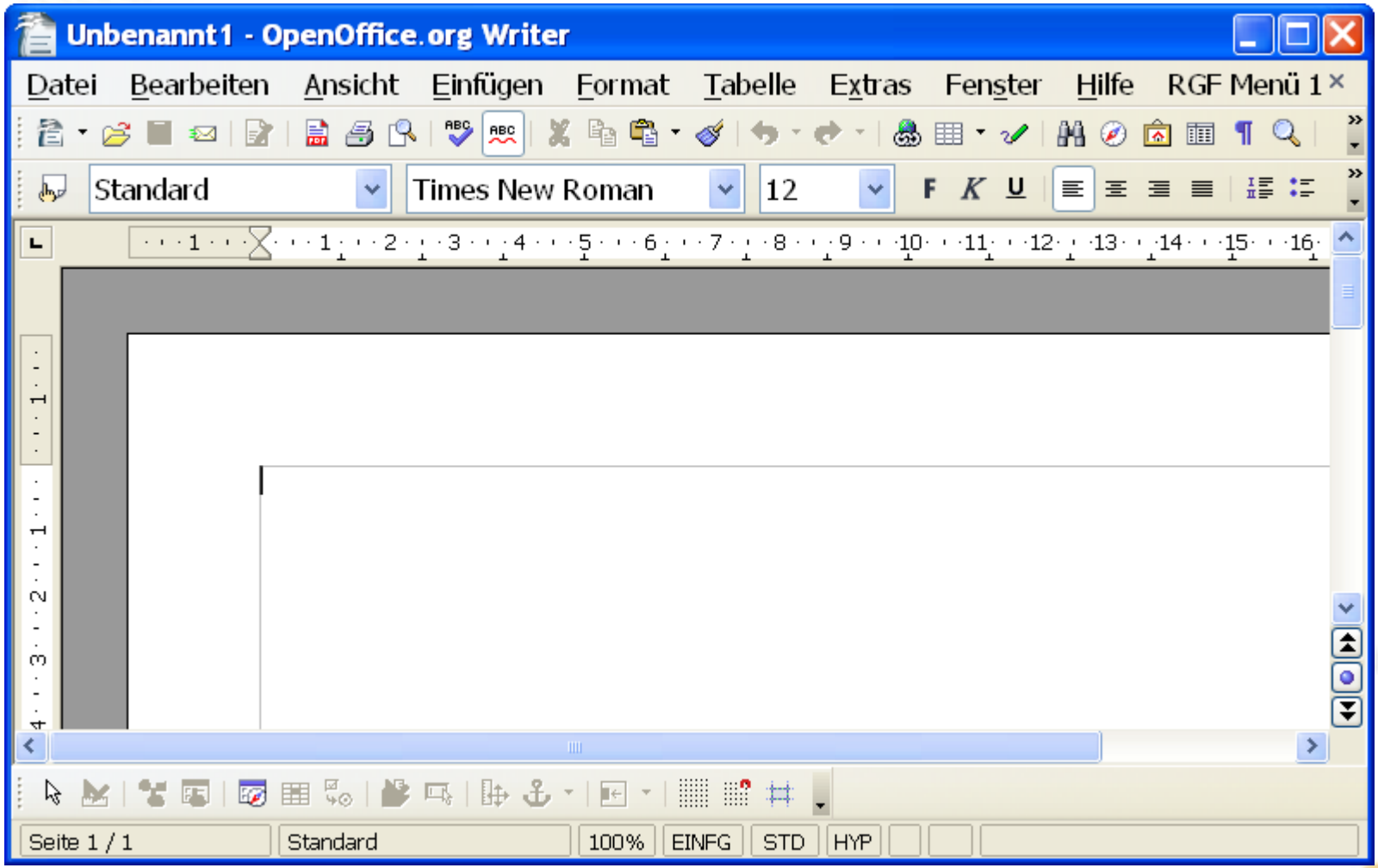


OOo/UNO – Some Thoughts

- **All** UNO components are addressable via Java!
- Jakarta BSF could be used to
  - Address any UNO component from scripts
    - Use "org.apache.bsf.Main" to start BSF scripts that are stored in files from the command line, hence
      - Automating OOo with script files becomes possible!
    - *Any* BSF engine could be used to create OOo scripts!
    - Scripts are multiplatform by virtue of Java!
- Possible to further ease UNO scripting for ooRexx
  - UNO.CLS:
    - Requires (uses) itself BSF.CLS (BSF4Rexx)
    - Makes e.g. mandatory interface querying a breeze!
    - Adds **powerful** UNO reflection capabilities!

```
class OOoWithJava {   // A Java program to create an empty word processor document with OOo 2.x
   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];

         String url="private:factory/swriter";
         com.sun.star.lang.XComponent xComponent =       // text document
                xComponentLoader.loadComponentFromURL(url, "_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);
   }
}
```

An Empty Text Document



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

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

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

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

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

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

::requires BSF.CLS   /* get the ooRexx BSF4Rexx support  */
```

Some Observations

- The ooRexx transcription is more ledgible
  - Mostly, because of no type casting
- There are recurrent statements, especially:
  - **queryInterface(***interfaceClass, UNO_Object***)**
    - Of **utmost** importance for strongly typed languages like Java
- Using ooRexx features, UNO specifics can be swapped out and coding needs can be cut down even more and somtimes dramatically
  - "UNO.CLS"

```
xCompLoader = UNO.createDesktop() ~XDesktop ~XComponentLoader

url="private:factory/swriter"
xWriterComp = xCompLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
say "Successful run."

::requires UNO.CLS   /* get the ooRexx UNO support */
```

Some Observations on UNO.CLS

- Recurrent sequence of statements are organized in public routines, e.g. "UNO.createDesktop()"
  - Connects to OOo and retrieves the Desktop object
- UNO objects become ooRexx proxies, e.g.
  - Querying interfaces reduced to merely sending the *unqualfied (!) name of the interface class* as a message, e.g. "~XDesktop", "~XComponentLoader"
- Recurrently needed objects are pre-created and pre-registered, e.g. ".uno~noProps"

OOo: Creating a Text Document

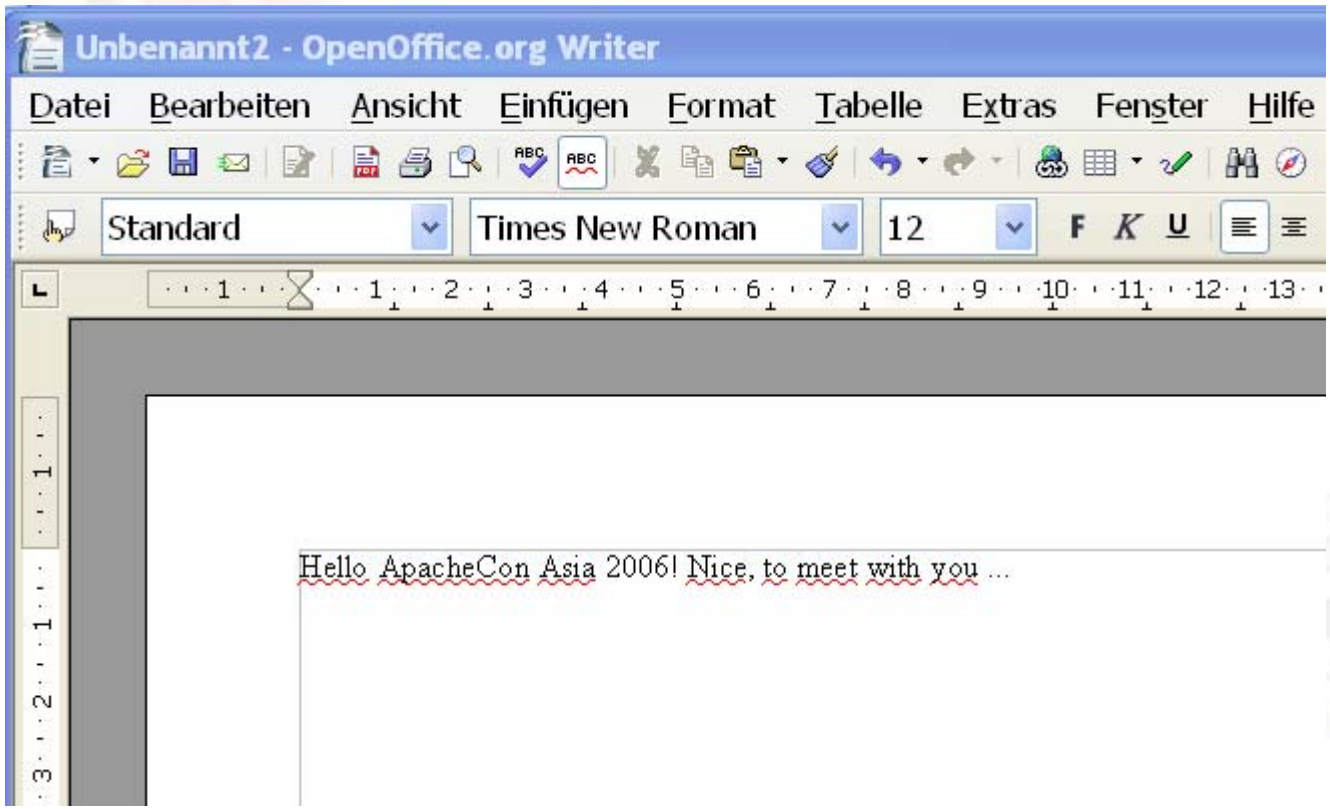
```
 /* get the OOo Desktop service object */
oDesktop = UNO.createDesktop()

   /* query interfaces to get to componentLoader interface */
xCompLoader = oDesktop~XDesktop~XComponentLoader

   /* open a blank (new) wordprocessor component */
url = "private:factory/swriter"  /* determine the component type  */
xWriterComp = xCompLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
```

```
   /* get the text object via the XTextDocument interface   */
text = xWriterComp~XTextDocument~getText

   /* set the string */
text~setString("Hello ApacheCon Asia 2006! Nice, to meet with you ...")
```

```
::requires UNO.CLS    /* get UNO support (which loads BSF.CLS) */
```

file:///c:/docs/aFile.odt
http://www.RexxLA.org/aFile.odt
ftp://www.RexxLA.org/aFile.odt

OOo: Creating a Text Document



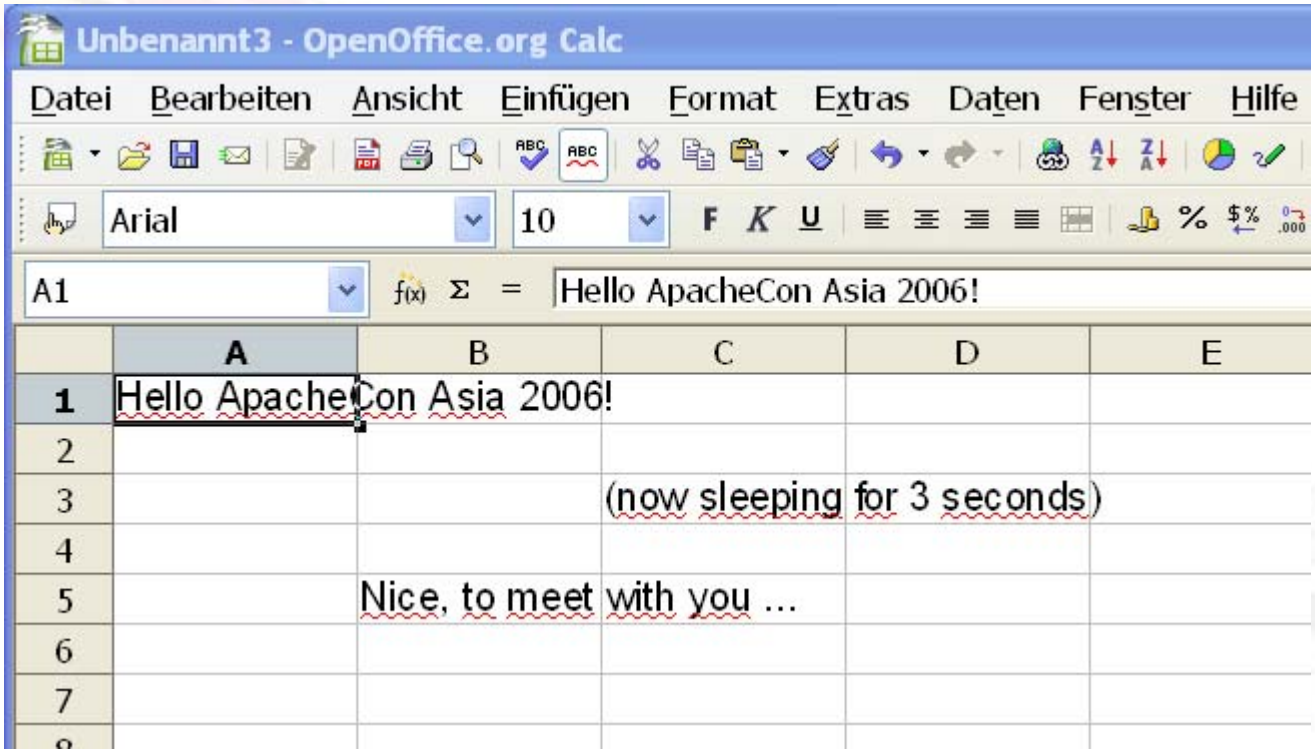
OOo: Creating a Calc Document

```
 /* get the OOo Desktop service object */
oDesktop = UNO.createDesktop()

   /* query interfaces to get to componentLoader interface  */
xCompLoader = oDesktop~XDesktop~XComponentLoader

   /* open a blank (new) spreadsheet component        */
url = "private:factory/scalc"
xCalcComp = xCompLoader~loadComponentFromURL(url, "_blank", 0, .UNO~noProps)
```

```
   /* get first sheet in the spreadsheet document     */
sheet = xCalcComp~XSpreadSheetDocument~getSheets~XIndexAccess~getByIndex(0)
xSheet=sheet~XSpreadSheet  /* query the XSpreadsheet interface of it */

   /* insert some text in the cells "A1", "C3, "B5"   */
call UNO.setCell xSheet, 0, 0, "Hello ApacheCon Asia 2006!"

sleep=3     /* determine how many seconds to sleep    */
call UNO.setCell xSheet, 2, 2, "(now sleeping for" sleep "seconds)"
call sysSleep sleep         /* sleep two seconds      */

call UNO.setCell xSheet, "B5", "Nice, to meet with you ..."
```

```
::requires UNO.CLS    /* get UNO support (which loads BSF.CLS) */
```

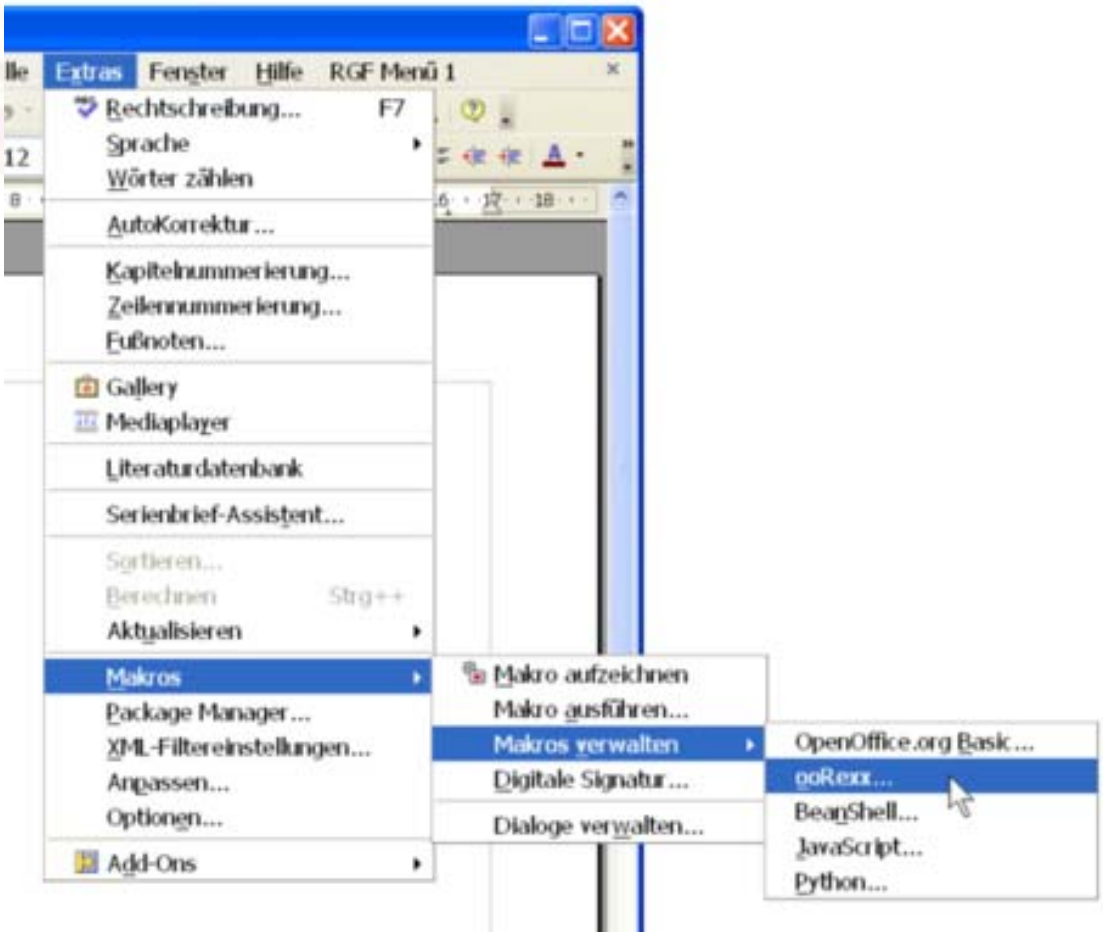
OOo: Creating a Calc Document

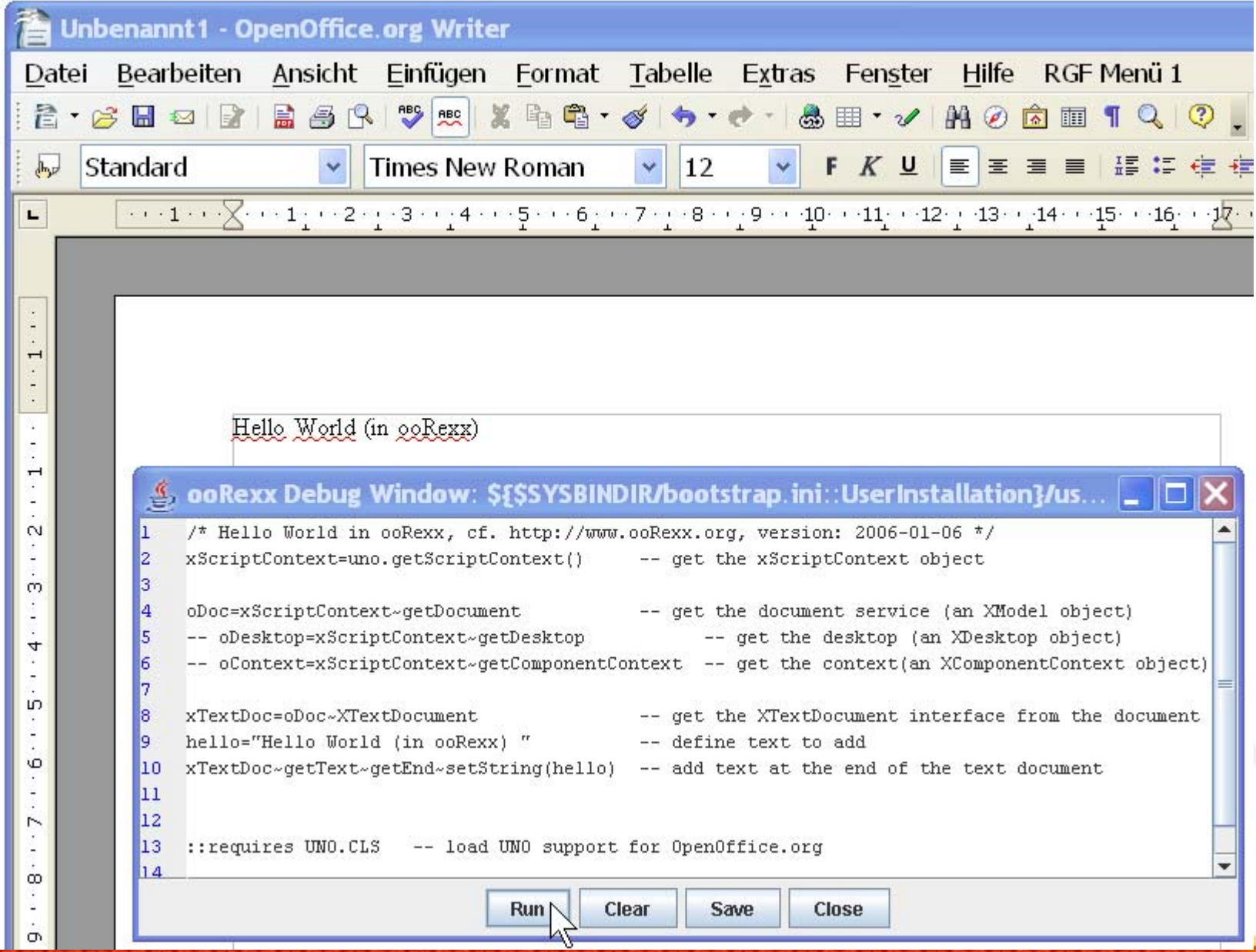


OpenOffice.org – BSF Macros!

- OOo version 2.0 (Fall 2005)
  - Incorporates a *Java-based OOo Scripting Framework!*
  - Allows OOo to deploy scripting languages, which are created/supplied by third parties
    - Enables the creation of macros that can be run against any OOo UNO component
    - The OOo scripting framework supplies the components as arguments for which the macro should run/got invoked
  - *Using Jakarta's BSF one can incorporate any of the BSF engines to be used as a macro language for OOo !*

ooRexx as an OOo Macro Language





Roundup, 1

- Jakarta BSF
  - Great enabling technology
  - Easy to use from Java
    - Couldn't possibly get easier
    - *No excuse that Java applications do not supply scripting interfaces for their users/customers!*
  - Quite a number of available scripting languages
    - Easy to add new scripting languages thanks to the BSF framework
  - Apache projects allow to take advantage of Jakarta BSF (e.g. ant, Taglibs, Xalan) for *your* benefit!

Roundup, 2

- BSF4Rexx
  - Employs Jakarta BSF
  - Makes Rexx and ooRexx available as a scripting language for Java
  - Allows ooRexx to use Java objects, as if they were ooRexx objects
    - in effect camouflaging Java as ooRexx!
  - Allows remote-controlling/interacting with any application that supplies a Java interface !
  - Allows *end-user-programmers* to interact with Java and automate/script OOo in a remarkable easy and therefore efficient manner!

Links

- http://jakarta.apache.org/bsf
- http://www.ooRexx.org
- http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/
  will be moved eventually to:
  http://sourceforge.net/projects/bsf4rexx
- http://www.RexxLA.org
- http://www.OpenOffice.org
- http://api.openoffice.org/SDK/
- http://codesnippets.services.openoffice.org/
- http://wi.wu-wien.ac.at/rgf/diplomarbeiten
- http://ant.apache.org/manual/OptionalTasks/script.html
- http://jakarta.apache.org/taglibs/index.html
- http://jakarta.apache.org/taglibs/doc/bsf-doc/index.html
- http://xml.apache.org/xalan-j/extensions.html