ApacheCon

Jakarta BSF

Add Scriptability to Your Java Application

Leading the Wave of Open Source

Overview

- Jakarta's Bean Scripting Framework (BSF) 2.4
  - Brief History
  - Java
  - Java (BSF) ↔ C++-Examples
  - C++ ↔ Java (BSF) ↔ C++
- Upcoming BSF 3.0 (JSR-223/Java 1.6 Scripting)
  - Brief History
  - Java-Examples
- Roundup

Jakarta's Bean Scripting Framework (BSF) 2.4

Jakarta's BSF History, 1

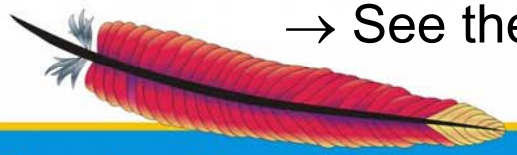
- Started 1999 as an IBM research project
  - **Sanjiva Weerawarana**
  - Matthew J. Duftler
  - Sam Ruby
  - Victor Orlikowski
- IBM Developerworks Java project
  - Open source Java framework
  - Package name: "com.ibm.bsf"

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**"
  - October 2006: version 2.4 officially released
- Apache projects employing BSF
  - Some Apache projects that use BSF
    - ant
    - Taglib ("BSF" taglib)
    - Xalan (XSLT processor)
  - Search in the documentation for "BSF", also
    - → See the last foil of this presentation ("Links")



BSF 2.4 Scripting Languages

- Some of the supported BSF scripting languages:
  - "**jacl**": Jacl (Tcl)
  - "**javascript**": JavaScript (Rhino)
  - "**jython**": Jython (Python)
  - "**netrexx**": NetRexx
  - "**prolog**": Jlog (PROLOG)
  - "**ruby**": JRuby
  - ...
- ... but also Java …
  - "**java**" (Java code "script style", gets compiled into boilerplate)
  - "**javaclass**" (invoke methods on Java class and Java objects)
- ... and even XSLT (Xalan) !
  - "**xslt**": XSL transformation

BSF in a Nutshell: Java Invoking Two 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 ("host")
    - 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 ("client")
    - Retrieves the panel object from the "BSFRegistry"
    - Adds a Border layout, creates and places "java.awt" objects

"ScriptedUI.java", 2 (Java Host)

```
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 (Javascript Client)

```
/* 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
```

„ant“ Examples, 1

- Examples by Kevin Jackson
  - In this case use Ruby
    - Read in a file and print out the file with associated line # using the <script> element
    - Using the <scriptdef> element define a script in a <target> element that prints out the classpath values

"ant" Examples, 2
Employing the <script> Element

```
<script language="ruby"><![CDATA[
    filename = "#{$project.getBaseDir()}/jruby-scriptdef-test.xml"
    f = File.new(filename, "r")
    f.readlines.each_with_index {|l,i|
      puts "#{i}: #{l}"
    }
  ]]>
</script>
```

"ant" Examples, 3
Employing the <scriptdef> Element

```
<target name="testScriptDef1">
    <!-- define it -->
    <scriptdef language="ruby" name="classpaths">
    <element name="classpath" type="path"/>
        paths = $elements.get("classpath")
        $self.fail("no classpaths") unless paths
        paths.each {|p| $self.log(p.toString()) }
    </scriptdef>
    <!-- use it -->
    <classpaths>
        <classpath path=".:${user.home}"/>
        <classpath path="${ant.file}" />
    </classpaths>
</target>
```

Using XSL as a Scripting Engine, 1

- BSF sample program
  - Java program (host)
    - Creates a java.awt.Frame
      - Creates a java.awt.Panel with a java.awt.GridLayout (no row limit, two columns) and adds it to the frame object
        - » Creates two java.awt.Labels and adds them (as headings) to the panel object
    - Uses the BSF registry to store
      - the "panel" Java object (used to output the result)
      - a java.io.FileReader Java object (created from the supplied XML file name) under the name "xslt:src"

Using XSL as a Scripting Engine, 2

- Invokes the BSF "xslt" engine supplying the XSL file's content
  - The XSL transformation will sort the person elements by the attribute named "first" (name)
  - and for each person selects the attributes "first" and "last", creating for each a "java.awt.Label" and adding that to the Java panel
- Packs and displays the frame object which contains the Java panel object that just got extended by the XSL transformed data from the given XML file

– Command line invocation:

java TableFiller table-data.xml style1.xsl

Using XSL as a Scripting Engine, 3 "TableFiller.java" (Java Host)

```
... cut ...
        Frame frame = new  Frame ("Table Filler");
... cut ...
        Panel panel = new  Panel (new  GridLayout (0, 2));
        Font f = new  Font ("SansSerif", Font.BOLD, 14);
        Label l = new  Label ("First"); l.setFont (f); panel.add (l);
                l = new  Label ("Last"); l.setFont (f); panel.add (l);
        frame.add ("Center", panel);
```

```
        BSFManager mgr = new  BSFManager ();
        // make the panel available for playing in XSL
        mgr.declareBean ("panel", panel, panel.getClass ());

            // tell xslt (Xalan) what the input xml file is
        mgr.registerBean ("xslt:src", new  FileReader (xmlfilename));

            // load and run the xsl file to fill in the Java table
        mgr.exec ("xslt", xslfilename, 0, 0,
                        IOUtils.getStringFromReader (new  FileReader (xslfilename)));
```

```
        // now pack and display the frame
        frame.pack ();
        frame.setVisible(true);
... cut ...
```

Using XSL as a Scripting Engine, 4 "style1.xsl" (XSL Client)

```
<?xml version='1.0'?>
    <!-- This stylesheet fills in the data by sorting on the first name. -->
    <xsl:stylesheet  xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                     xmlns:java="http://xml.apache.org/xslt/java"
                        version="1.0">

      <xsl:param name="panel"/> <!-- get access to the "panel" Java object -->

      <xsl:template match="data">
        <xsl:apply-templates select="person">
          <xsl:sort select="@first"/>        <!-- sort by attribute "first" -->
        </xsl:apply-templates>
      </xsl:template>

      <xsl:template match="person">
        <xsl:variable name="junk1"
                      select="java:add ($panel, java:java.awt.Label.new (string(@first)))"/>

        <xsl:variable name="junk2"
                      select="java:add ($panel, java:java.awt.Label.new (string(@last)))"/>
      </xsl:template>

    </xsl:stylesheet>
```

Using XSL as a Scripting Engine, 5 "table-data.xml"

```
<?xml version="1.0"?>

<data>
  <person first="Sanjiva"    last="Weerawarana"/>
  <person first="Matt"       last="Duftler"/>
  <person first="Paco"       last="Curbera"/>
  <person first="Sam"        last="Ruby"/>
  <person first="Stephen"    last="Boies"/>
  <person first="Sanka"      last="Samaranayake"/>
  <person first="Anthony"    last="Elder"/>
  <person first="Kevin"      last="Jackson"/>
  <person first="Nandika"    last="Jayawardana"/>
  <person first="Stefan"     last="Bodewig"/>
  <person first="Henning"    last="Schmiedehausen"/>
  <person first="Martin"     last="van den Bemt"/>
  <person first="Rony G."    last="Flatscher"/>
</data>
```

```
java TableFiller style1.xsl table-data.xml
```

Java (BSF) ↔ C++

- BSF is a Java framework
- There are plenty of scripting languages that are not implemented in Java, but in C or C++
- It would be just great to be able to employ all such scripting languages by Java via BSF!
- In addition, it would become possible for such scripting languages to *use all of Java as a huge external function* library which has been ported to all major operating systems already!

Rexx
A Non-Java Scripting Language

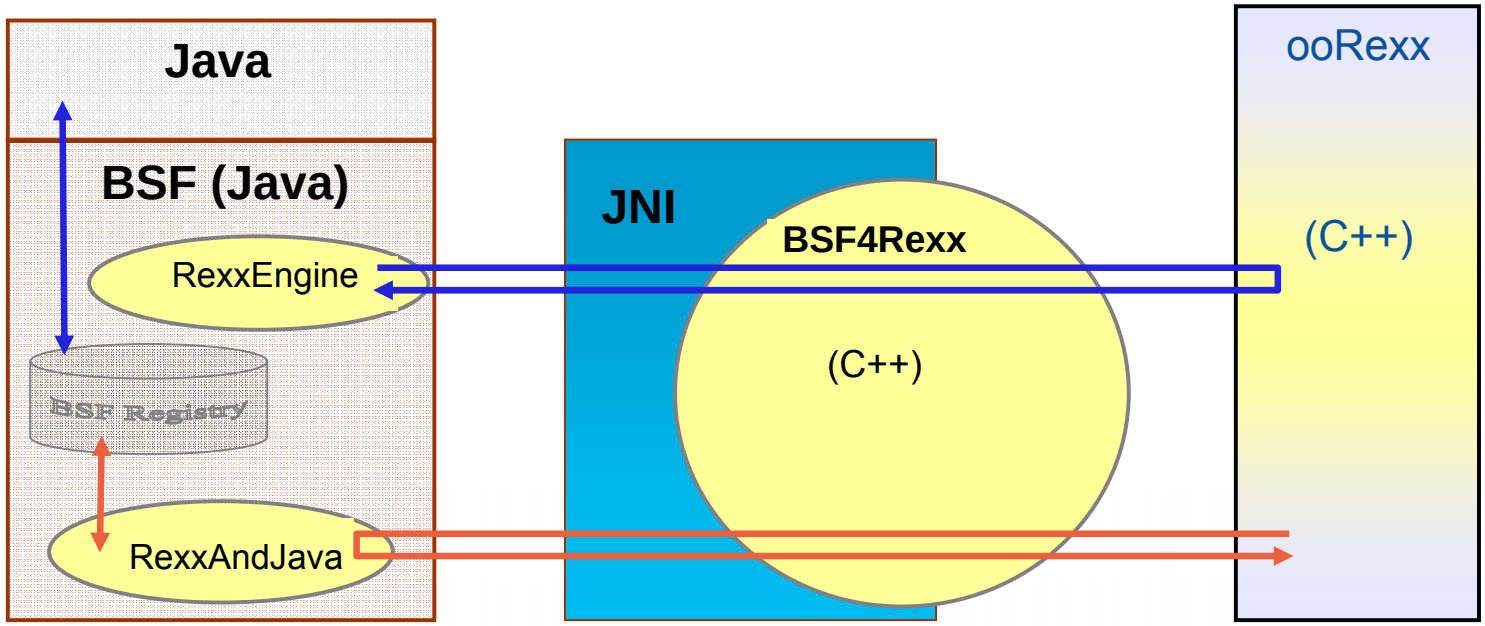
- Rexx
  - 1979 by Mike F. Cowlishaw (IBM)
    - Goal for devising a "**human centric**" scripting language
      - "Everything is a string"
      - Small language (stable set of builtin functions)
      - Easy syntax (almost like pseudo-code!)
        → *End-user programming* possible !
    - Originally to replace the awkward "EXEC II" on mainframes
  - ANSI Rexx standard in 1996
  - Available for practically all operating systems, e.g.
    - AmigaOS, AIX, DOS, IBM mainframe OS, Linux, MacOSX, OS/2, Solaris, Windows, …

ooRexx
A Non-Java Scripting Language

- Non-profit "Rexx Language Association (RexxLA)"
  - Fall 2004: Received the source code for IBM's commercial Object REXX
  - Spring 2005: Released "ooRexx 3.0" as FOSS (free & open source)
  - November 2007: Released "ooRexx 3.2" (a very significant update)
- Fully compatible with Rexx
- Adds a powerful object-oriented model
  - "Everything is an object"
  - Metaclasses, Reflection, Multiple-inheritance, Unknown-mechanism, One-off objects, ...
  - Easy syntax (almost like pseudo-code!)
    - → *End-user programming* possible as well !
  - Still a small and easy to learn language
  - Available pre-compiled for, e.g.
    - AIX, Linux, MacOSX, Solaris, Windows
- Implemented in C++

"BSF4Rexx" - A BSF Engine (Bridging Java and C++)

- Java (BSF) ↔ C++



BSF4Rexx, An Example ooRexx Using Java

```
call bsf.cls                              /* get ooRexx support for BSF        */
s=bsf.loadClass("java.lang.System")       /* load the Java System class object */
say s~getProperty("java.version")         /* get and output Java version       */
```

Yields (maybe):

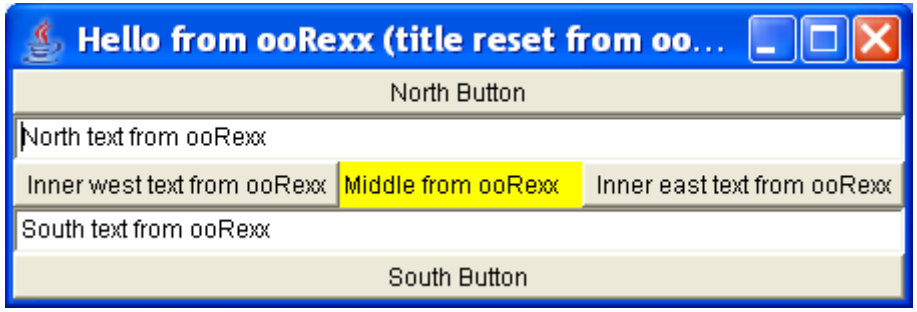
```
1.6.0_02
```

"ScriptedUI.java" (cont'd), 5 (ooRexx Client)

```
/* 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" (cont'd), 6

```
java ScriptedUI ui.rex
```



C++ ↔ Java (BSF) ↔ C++

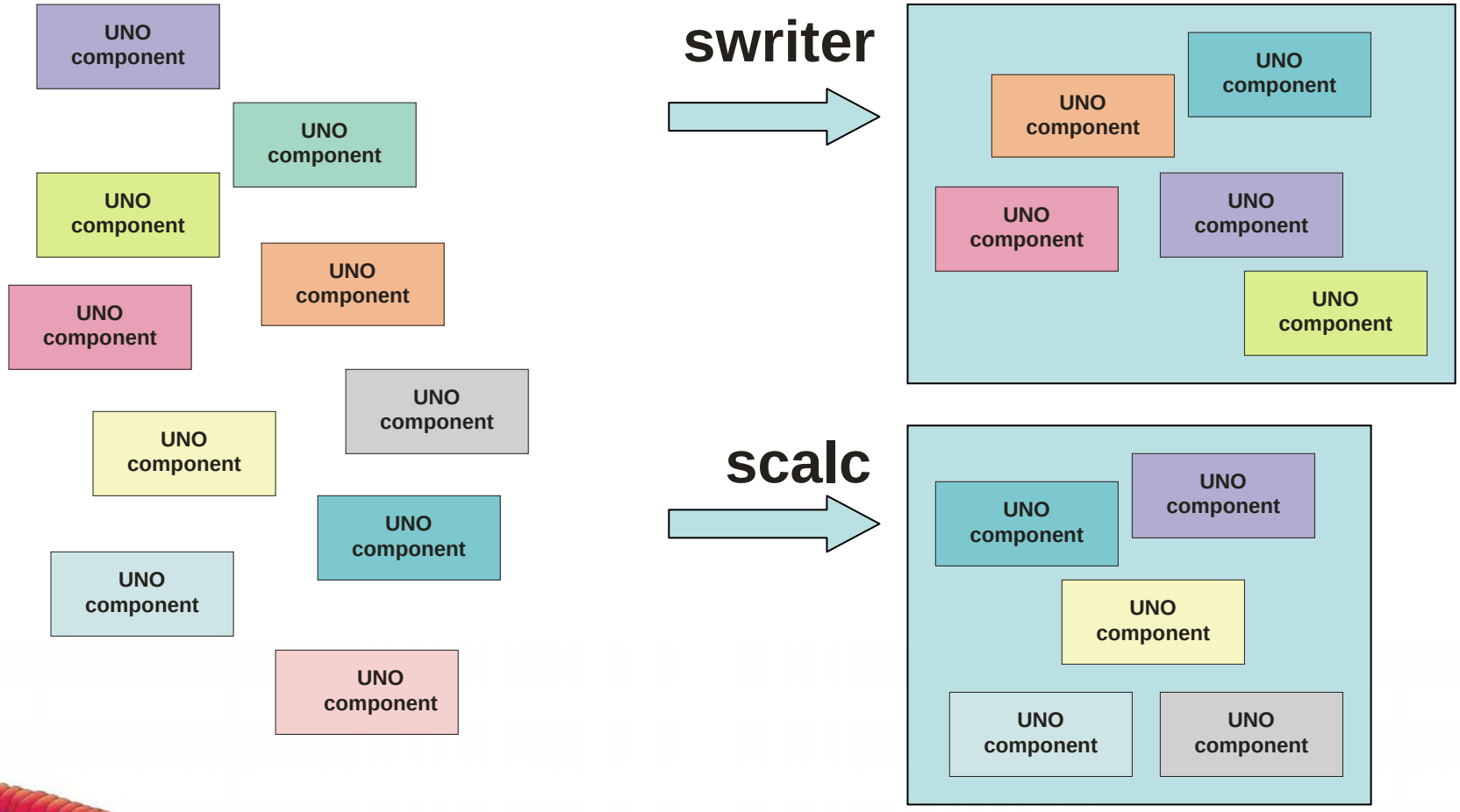
- Observations
  - Java has become one of the most important programming languages
  - Non-Java applications start to supply Java interfaces to service Java programs
- Using BSF one could use *any* supported scripting language to interface with such applications without a need for a specific function library!

Case Study StarOffice/OpenOffice.org

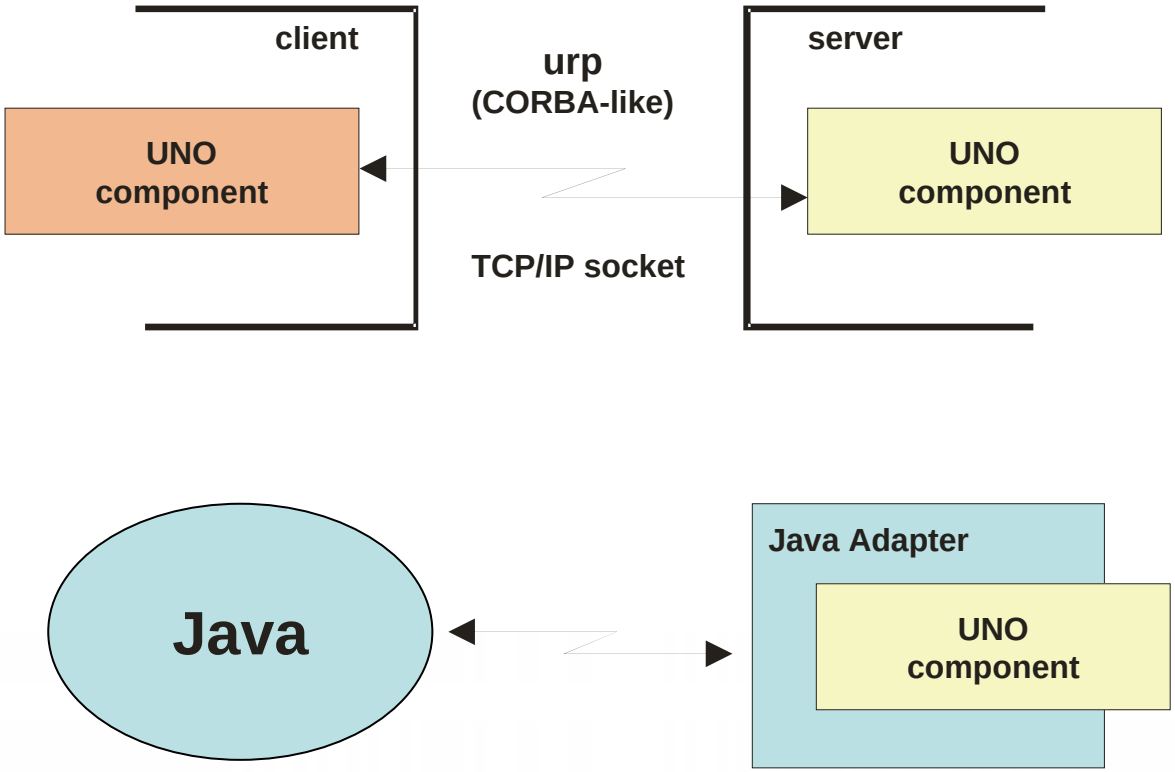
- Office-suite with OO-design to the core
  - Written in C++
  - Multiplatform
  - *Many* filters (eg. Microsoft Office file formats)
- Sun bought StarDivision in 1999
  - Opensource version in 2001 released
  - http://www.**O**pen**O**ffice.**o**rg ("OOo")

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

OOo "UNO" Architecture



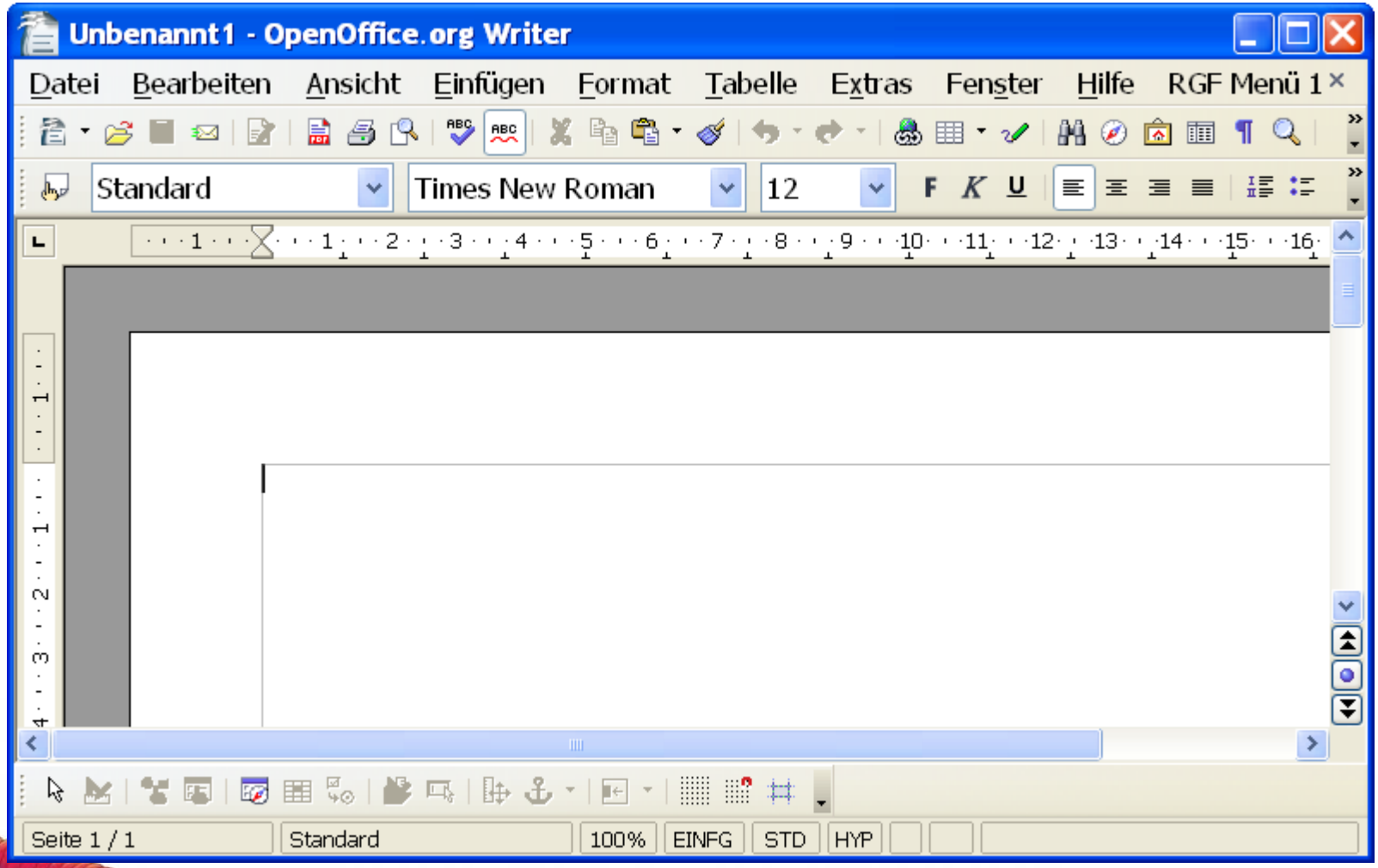
UNO Interaction



```
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);
   }
}
```

A Java Program to Create an Empty OOo Word Processor Document

An Empty Text Document



```
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, which uses BSF */
```

An ooRexx Program to Create an Empty OOo Word Processor Document Taking Advantage of BSF

Some Observations

- The ooRexx program is considerably smaller and as a result more ledgible, because
  - No explicit need for type casting
    - A strongly typed language like Java needs to employ "queryInterface(interfaceClass, UNO_Object)" over and over again
  - ooRexx Module "UNO.CLS" includes
    - Simple routines for recurrent tasks like creating the OOo desktop,
    - A Mechanism to allow for sending interface queries as plain (unqualified) ooRexx messages

OOo: Creating a Text Document With Content

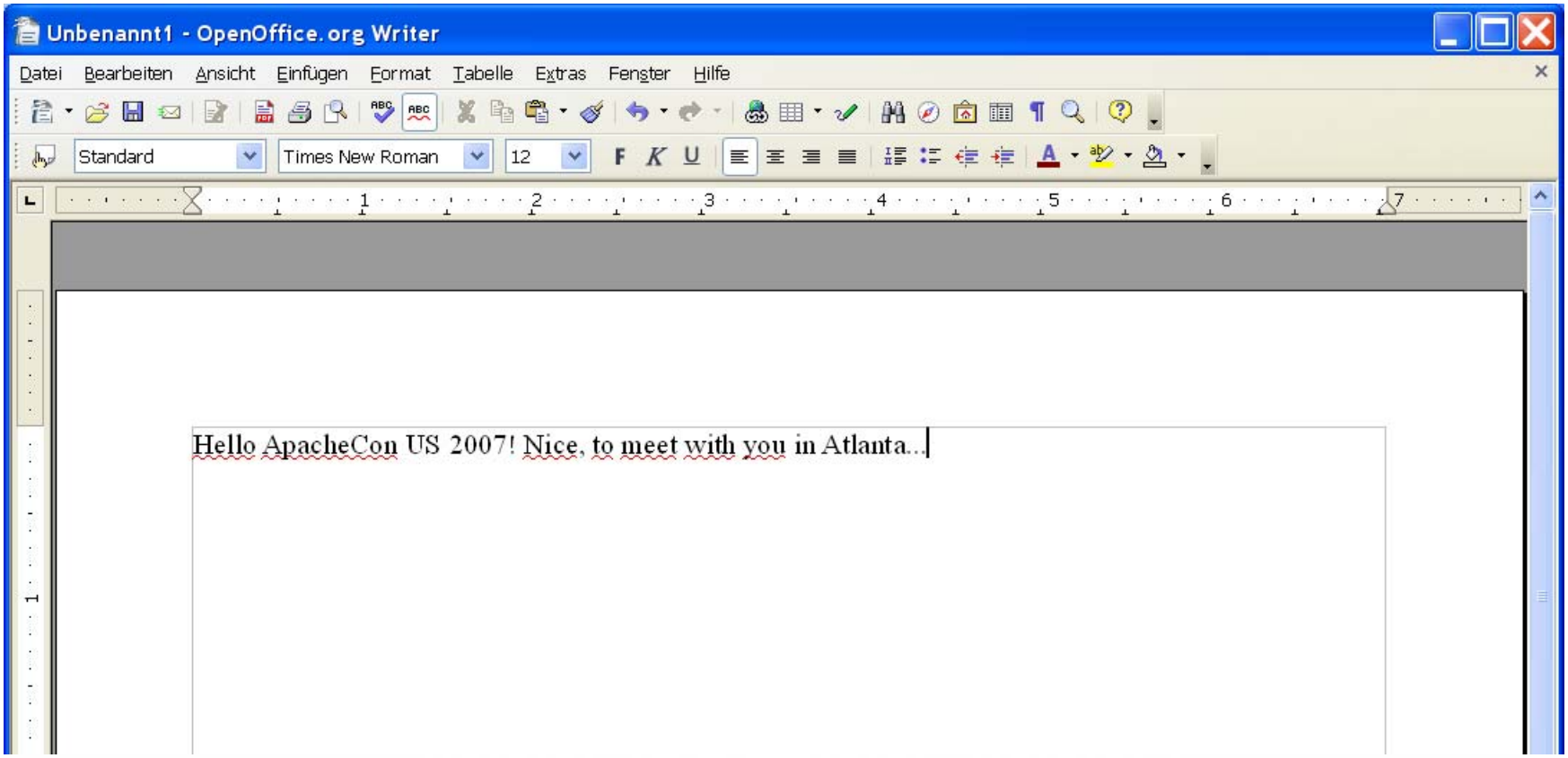
```
 /* get the OOo Desktop service object */
oDesktop = UNO.createDesktop()

   /* 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 US 2007! Nice, to meet with you in Atlanta...")
```

```
::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

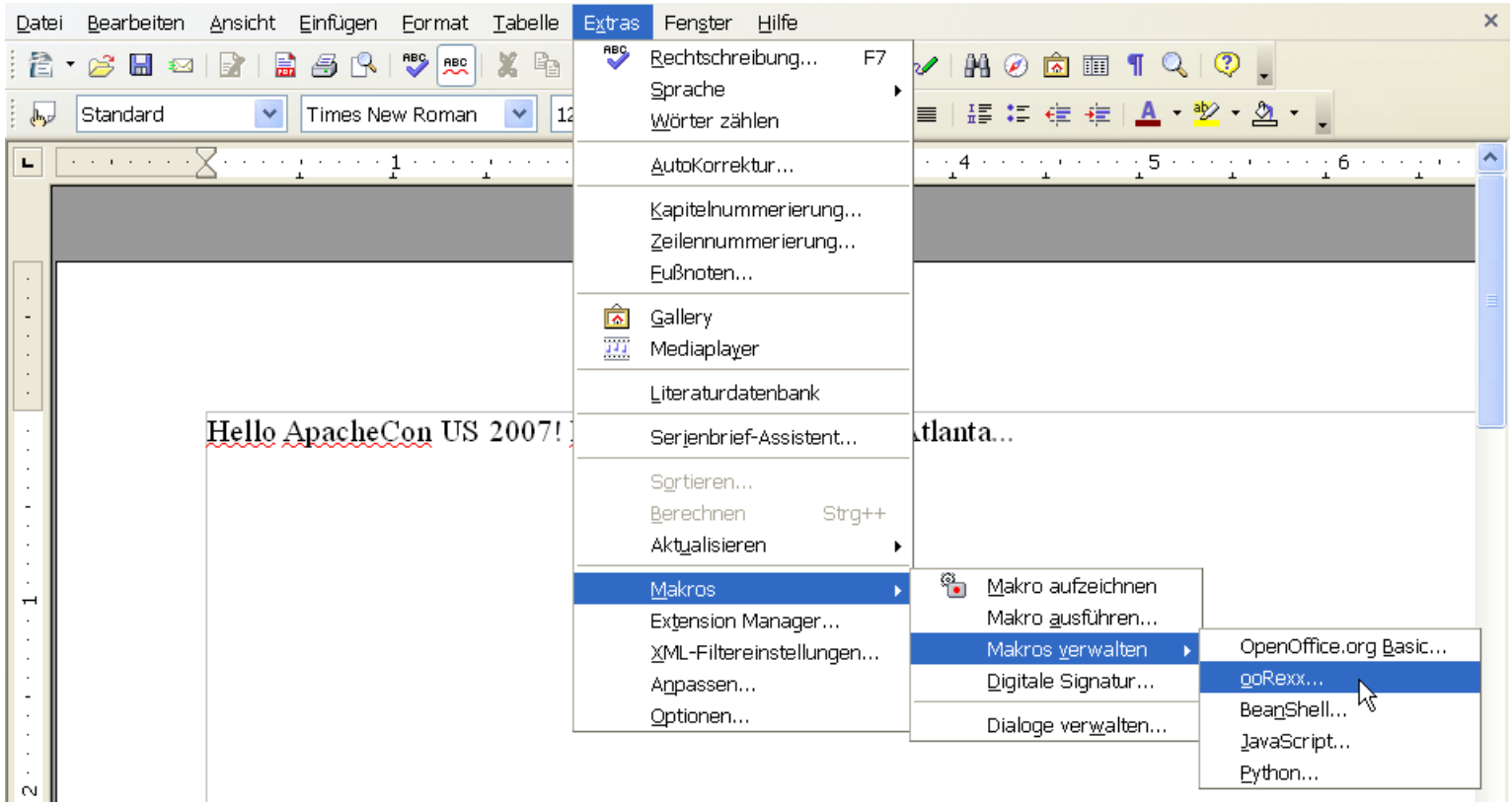
OOo: Creating a Text Document

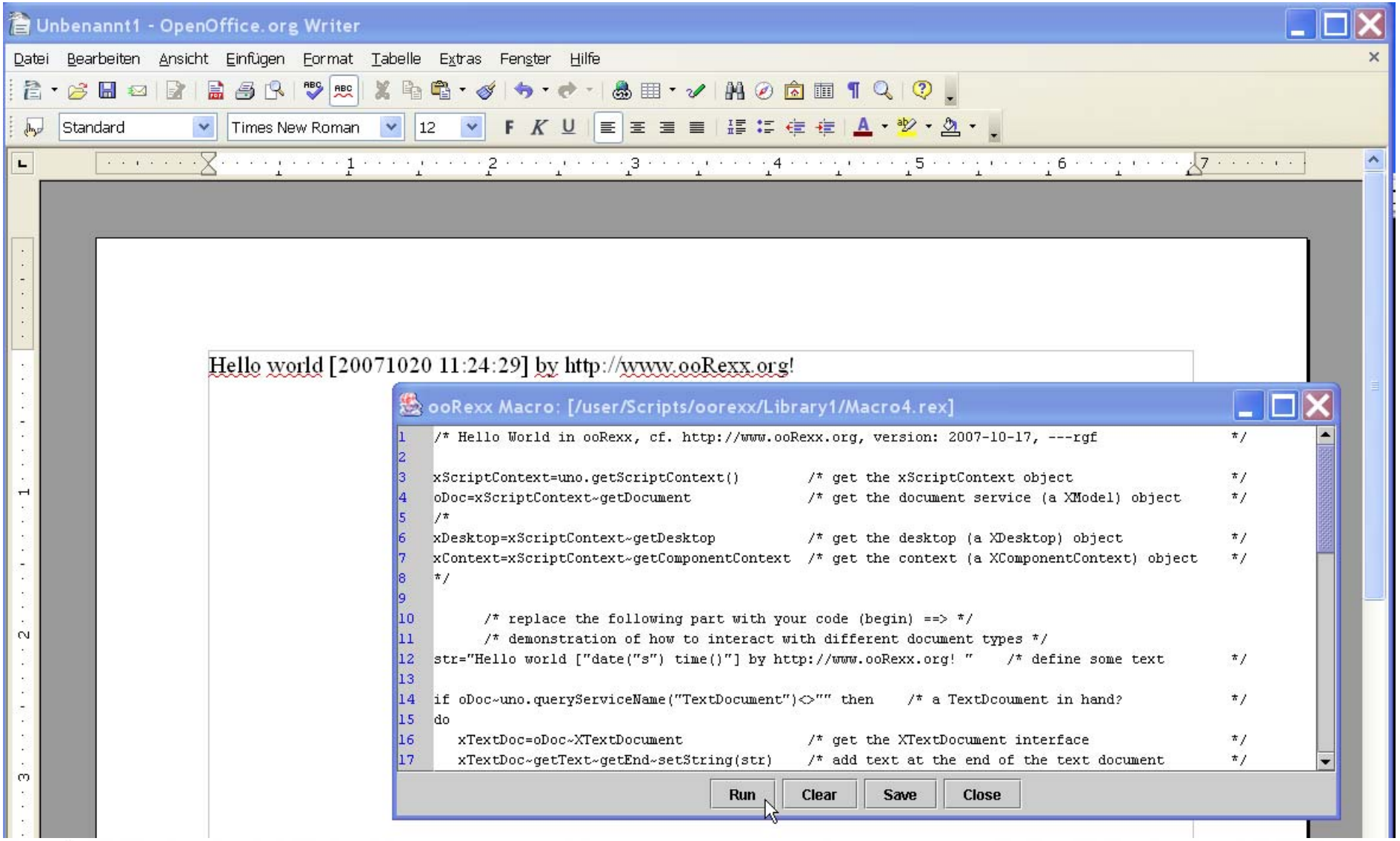


OpenOffice.org – BSF Macros!

- Since OOo version 2.0 (Introduced 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




Unbenannt1 - OpenOffice.org Writer
Datei Bearbeiten Ansicht Einfügen Format Tabelle Extras Fenster Hilfe
Standard
Times New Roman
12
Hello world [20071020 11:24:29] by http://www.ooRexx.org!
ooRexx Macro: [/user/Scripts/oorexx/Library1/Macro4.rex]
/* Hello World in ooRexx, cf. http://www.ooRexx.org, version: 2007-10-17, ---rgf */
xScriptContext=uno.getScriptContext() /* get the xScriptContext object */
oDoc=xScriptContext~getDocument /* get the document service (a XModel) object */
/*
xDesktop=xScriptContext~getDesktop /* get the desktop (a XDesktop) object */
xContext=xScriptContext~getComponentContext /* get the context (a XComponentContext) object */
*/
/* replace the following part with your code (begin) ==> */
/* demonstration of how to interact with different document types */
str="Hello world ["date("s") time()"] by http://www.ooRexx.org! " /* define some text */
if oDoc~uno.queryServiceName("TextDocument")<>"" then /* a TextDcoument in hand? */
do
xTextDoc=oDoc~XTextDocument /* get the XTextDocument interface */
xTextDoc~getText~getEnd~setString(str) /* add text at the end of the text document */
Run
Clear
Save
Close

Add *Your* BSF Scripting Language as a Macro Languages to OOo

- If you are interested in using your favorite scripting language in OpenOffice, you can! And quite easily so, just
  - Look at the current Rexx implementation for the OOo scripting framework which uses BSF (pure Java)
  - Copy the few existing files to a new subdirectory
    - Java programs plus one template file containing a template program in your scripting language of choice
  - Change the name of the scripting language in a few places to match your scripting language of choice, adapt the Manifest file accordingly, and
  - … you should be done!

Upcoming BSF 3.0

JSR-223/Java 1.6 Scripting

Upcoming BSF 3.0
An Implementation of JSR-223, 1

- JSR-223
  - Task: create a scripting framework for Java
    - Sun's original motivation: create support for PHP
      - "JSR-000223: Scripting Pages in Java Web Applications"
    - Final title: "JSR 223: Scripting for the Java Platform"
    - Apache Jakarta BSF committers (Victor Orlikowski, IBM; Rony G. Flatscher, WU Wien) served as experts
  - Work started in 2003, ended last year (2006)

Upcoming BSF 3.0
An Implementation of JSR-223, 2

- JSR-223
  - Results (reference implementation) used by Sun to implement the Java scripting framework for Java 1.6
    - Cf. Java 1.6 package named "javax.script"
    - At the end of the development changes got introduced to the interfaces to "take advantage of Java 1.6 features"
      - Results therefore suggest that JSR-223 scripting would only be available in Java 1.6 and up … however …

Upcoming BSF 3.0
An Implementation of JSR-223, 3

- BSF 3.0 (currently in beta)
  - Opensource implemention of JSR-223
  - Allows deploying "javax.script" with *earlier versions of Java*, starting with Java 1.4
    - → No need to upgrade deployed Java installations to Java 1.6
    - → Ability to enhance existing, deployed installation with scripting capabilities without being forced to upgrade to 1.6
    - → Java applications that employ JSR-223 scripting can be deployed also among Java 1.4 and 1.5 installations!
    - → No dependency on Java 1.6 at all
      - However, Java 1.4 and 1.5 programs using BSF 3.0 will run under Java 1.6 and higher by using Java 1.6 "javax.script" package

BSF 3.0/JSR-223 in a Nutshell: Java Invoking a Javascript Script

```
import javax.script.*;  // BSF 3.0 support using JSR-223/Java 1.6 package name

/** Java program that shows the easyness of deploying scripts with BSF 3.0/JSR-223. */
public class Test_BSF3
{
  /** Running in-line defined scripts. */
  public static void main (String[] args) {
    try {
        ScriptEngineManager mgr = new ScriptEngineManager ();

        ScriptEngine jsEngine = mgr.getEngineByName("javascript");
        String scriptCode = "java.lang.System.out.println(\"JavaScript was here!\");";
        jsEngine.eval(scriptCode);
    }
    catch (Exception e) {e.printStackTrace();}
  }
}
```

JavaScript was here!

"ScriptedUI_BSF3.java", 1 (Java Host)

```
public class ScriptedUI_BSF3 {

   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_BSF3.java", 2 (Javascript Client)

```
/* filename "ui_bsf3.js" (JavaScript)    */

   /* picking up the Java Panel object not necessary anymore!
      "p" (a Java Panel object) gets registered with the
      JavaScript engine right before the script gets invoked! */

   /* 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 \"ui_bsf3.js\" (JavaScript)");
```

"ScriptedUI_BSF3.java", 4

```
java ScriptedUI_BSF3 ui_bsf3.js
```

Roundup

Roundup, 1

- Jakarta BSF 2.4
  - **Great** enabling technology
  - Easy to use from Java
    - Couldn't possibly get easier
    - ***No excuse*** *that Java applications do not supply (open platform) scripting to their users/customers!*
  - Quite a number of available scripting languages
    - Easy to add new scripting languages thanks to the BSF framework
      - Cf. <http://jakarta.apache.org/bsf>
    - You can even use JSR-223/Java 1.6 scripting engines!
      - Cf. Apache Incubator "Tuscany" sandbox ("JSR223BSFEngine.java", author: Ant Elder)

Roundup, 2

- Jakarta BSF 2.4
  - Apache projects allow to take advantage of Jakarta BSF (e.g. ant, Taglibs, Xalan) for *your* benefit!
    - Just look for the documentation and examples
  - The BSF 2.4 scripting framework could be even compiled for Java 1.3
- Upcoming BSF 3.0
  - Opensource implementation of JSR-223
  - Allows deploying "javax.script" on earlier Java version
  - Java applications can take advantage of JSR-223 scripting without being forced to upgrade to Java 1.6 !

Links

- http://jakarta.apache.org/bsf
- 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
- http://jcp.org/en/jsr/detail?id=223
- http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current/
- http://www.ooRexx.org
- news:comp.lang.rexx
- 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