Examples for Open Office Automation with Scripting Languages

Walter Augustin
Vienna University of Economics and Business Administration,
Reg. No. 9250416

E-Mail: h9250416@wu-wien.ac.at

January 10, 2005

Bachelor Course Paper
Department of Business Informatics
Prof. Dr. Rony Flatscher
Department Chair "Information Systems and Operations"

Contents

1 Basics

1.1About this paper

As the variety of open source software keeps growing, more and more developers spend their time developing competitive business applications. In the field of the standard "office" applications such as text processors, spreadsheets and presentation tools, Microsoft's most serious rival – although still far smaller – seems to be the Open Office suite.

Microsoft Office scripting has been popular for many years now. VBA (Visual Basic for Applications) is a comprehensive yet easy scripting language that comes with MS Office, but via the OLE automation bridge other scripting languages can be used as well.

For the purpose of writing "macros", Open Office offers a Java programming interface (Java UNO). This allows the professional programmer to take extensive control of the suite – but an average user, even with some programming knowledge – can hardly keep up.

Now the idea for using "simple" scripting languages for Open Office automation came up. Luckily, IBM donated its "Bean Scripting Framework" (BSF) to the open source community, namely the Apache Software Foundation. This set of classes allows to control arbitrary Java classes via scripting languages. It is the perfect solution for adding "macro" capability to existing applications.

Rainer Hahnekamp dedicated himself to the task of connecting scripting languages to Open Office via BSF [Hahnek05]. In his paper, he describes how to use languages such as Javascript, Rexx, Tcl, Python, or Prolog, for Open Office scripting - under Windows as well as under Linux.

This document, on the other hand, demonstrates in more detail what "every-day" work with those macro programs could be like. For this reason, the author adapted some of the Java examples that come along with the Open Office Developer's Kit to Javascript and Rexx. These "nutshell" examples are meant to serve as a starting point in automating Open Office. Besides that, they demonstrate the technical boundaries which a script programmer could encounter, as well as the advantages and drawbacks which one technique may have over another.

1.2Using the examples

Most of the examples listed in this paper are taken from the Open Office SDK (Java) Developer's Guide (see chapter 7, "Download Links") and translated to the respective scripting language. Some of them have been modified, many of them have been complemented with additional comments.

In the "Download Links" chapter you will find a download URL for the whole set of examples in compressed format. It is strongly recommended that you use these files

instead of typing them from the course paper. The downloadable example files come complete with connection headers and proper line breaking.

Note: Rexx uses the comma for indicating that the command continues on the next line. If you copy Rexx code from this document and merge two lines, you will have to delete the comma!

In this document, the "universal" connection headers will be presented in the beginning of the according sections. *The required connection parts as well as auxiliary functions are not repeated throughout the examples in this paper.*

For better orientation, scripts are marked correspondingly in their header comment:

[*script no.*][*scripting approach*][*scripting language*]

For example:
[88][BSF][Rhino] Script No. 88, approach: BSF, language: Rhino (JavaScript)
[99][OLE][VBS] Script No. 99, approach: OLE, language: Visual Basic Script

If your system meets the technical requirements (chapter 1.3), you can execute them as follows.

1.2.1BSF-based examples

The BSF scripts are provided in Rhino (= Javascript) and Object Rexx (with bsf4rexx installed). All these examples can be comfortably executed using the ScriptRunner class which is provided in the script files download package, as well. Here is the listing of the ScriptRunner class.

```
/*
 * Created on January 03, 2005
 * @author matto (Walter Augustin)
 *
 * ScriptRunner is a simple class for executing the sample scripts provided
 * in the course paper of Walter Augustin, 9250416,
 * Vienna University of Economics and Business Administration
 *
 * E-Mail: h9250416@wu-wien.ac.at
 *
 * Usage:
 *      1. Put into a directory together with the sample scripts
 *      2. Change "pathToScripts" in the main method to that directory
 *      3. Change "file" in the main method to the filename of your sample script
 *         (e.g.: Example_01.js)
 *      4. Uncomment either runRhino(code) or runRexx(code)
 *         (depending on which language the script you wish to run is written in)
 *      5. Recompile and run this class.
 *
 * Note:
 *      Requires an IBM BSF installation as well as the Open Office classes and
 *     BSF4Rexx (see paper).
 *
 */

import com.ibm.bsf.*;
import java.io.*;

public class ScriptRunner
{
        public static void main(String[] args) throws IOException
        {
                String file = "Example 01.js";
                String pathToScripts = "C:\\oo-scripts\\";
                String code = loadFile(pathToScripts + file);
```

```
                runRhino(code);
                //runRexx(code);
        }

        public static void runRexx(String code)
        {
                run("rexx", code);
        }

        public static void runRhino(String code)
        {
                run("javascript", code);
        }

        public static void run(String language, String code)
        {
                try
                {
                        BSFManager mgr = new BSFManager();

                        mgr.exec(language, "debug infos", 0, 0, code);

                        System.out.println("... DONE ...");
                        System.exit(0);
                }
                catch(BSFException ex)
                {
                        ex.printStackTrace();
                }
        }

        public static String loadFile(String fileName)
        {
                String resultStr = "";

                File file = new File(fileName);

                try
                {
                        FileReader reader = new FileReader(file);
                        BufferedReader in = new BufferedReader(reader);
                        String line = "";

                        while((line = in.readLine()) != null)
                                resultStr += line + "\n";

                        in.close();
                        reader.close();
                }
                catch(IOException ex)
                {
                        ex.printStackTrace();
                }

                return resultStr;
        }
}
```

1.2.2OLE-based examples

OLE Example No. 29 (Visual Basic Script) can be started by double-clicking it or by calling it from the command line.

OLE Example No. 30 (Object Rexx) is executed either from the command line (e.g. using “rexxj Example_30.rex”) or via the Object Rexx Workbench (Windows: double-click).

OLE Example No. 31 (PHP) can be called from the command line (depending on the platform and PHP installation; modifications are necessary, see below) or via a web browser.

1.3Technical requirements

In this section, a brief summary of the requirements will be given. For further reference, especially regarding the installation of Java, BSF, and Rexx., see Rainer Hahnekamp's paper on BSF and Open Office [Hahnek05].

1.3.1Java

The BSF-based examples require – as BSF is written in Java – a Java environment (for example, J2SE).

1.3.2BSF

There are two flavors of BSF: IBM's distribution and the Apache distribution. Apache's version is more current, but on the other hand, the IBM version is better suited for use with the Rexx programming language.

1.3.3Open Office

If you would like to have the Open Office Developer's guide on your hard disk, download the Open Office SDK (see Download Links chapter). The Developer's guide provides a whole lot of scripting examples in Java (i.e., without the need for BSF).

In order to execute the scripts, you have to install Open Office, then make it "listen" (see

[Your local SDK path]/
docs/DevelopersGuide/FirstSteps/FirstSteps.htm#1+1+Programming+with+UNO

) by modifying an XML settings file, and include its .jar files in your Java classpath.

1.3.4Operating System

The BSF-based examples can be run on virtually every platform ("virtually" meaning, any platform for which a Java SDK as well as Open Office are available). OLE techniques are MS-Windows-only.

If you plan on running your Open Office scripts on Windows, OLE may be a serious alternative for you since it is a little easier than via BSF/Java. But consider two issues here: First, your scripts will remain windows-only and may be difficult to adapt to a Java environment. Second, there are more – and different – scripting languages available for use with the BSF approach. Choose wisely.

Java-based approach	OLE (Windows) approach
JavaScript (Rhino) Python (Jython, JPython) Tcl NetRexx Object Rexx Java JRuby JudoScript Groovy ObjectScript	Visual Basic Script Delphi PHP Object Rexx

Fig. 1: A selection of available programming languages for the different scripting approaches; underlined are those treated in this paper.

1.3.5Suggested Development Environments

1.3.5.1Development Environments for Java/BSF

You are free to use your favorite developing environment, if you have one. From command-line tools such as emacs, pico, or edit, to highly sophisticated applications like Eclipse and NetBeans, every editor will do.

For most efficient work, however, a more comfortable solution is recommended. Note that you will be programming in two different environments: Java on the one hand and the scripting language of your choice on the other.

Furthermore, if you are an advanced user, you will probably want to examine the .jar files provided by Open Office. Or, you could like to write your scripts with the comfort of syntax highlighting, each script in one different file – and not in form of a concatenated string hard-coded into a Java class's "main" function. Therefore, your programming environment should be able to handle files in different languages within the same project.

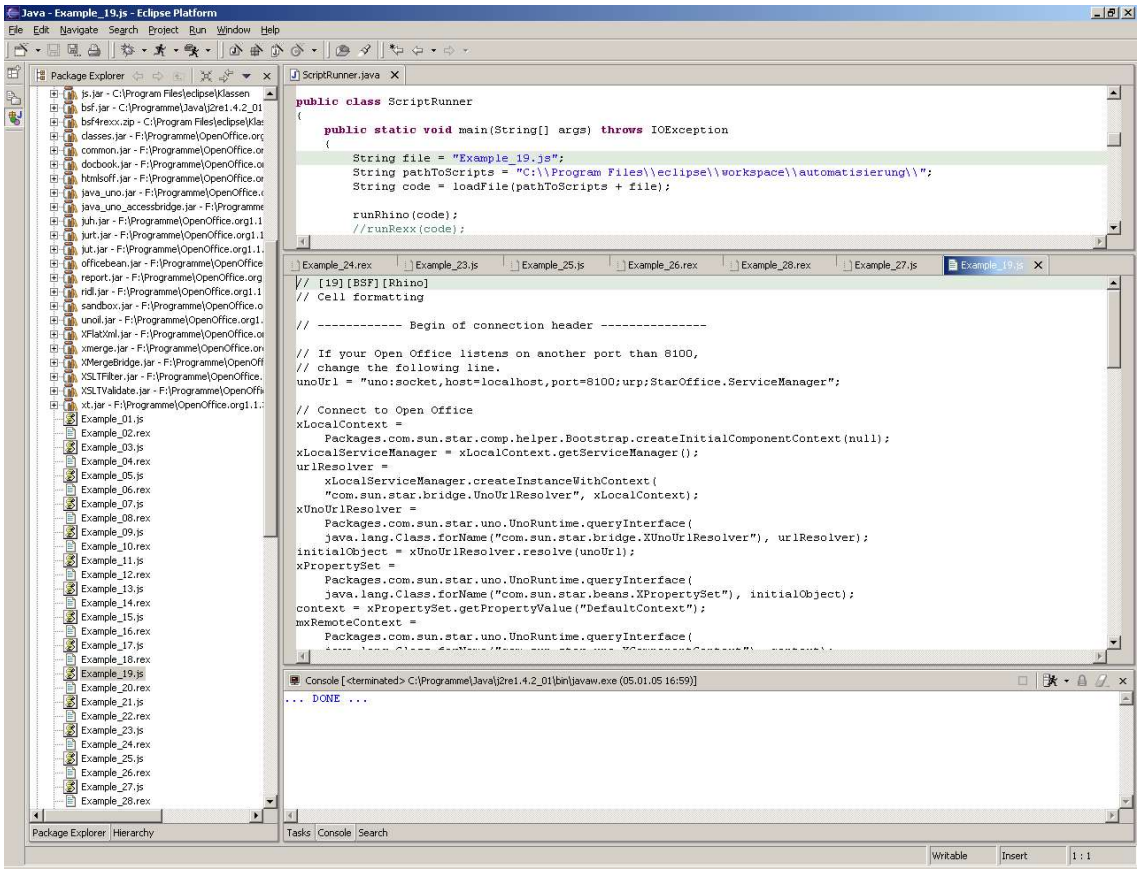

Fig. 2: The Eclipse development environment in use for running the examples

When you start your script, you run the main method of a Java wrapper class (such as the ScriptRunner class from chapter 1.2.1) which will integrate BSF, read in your script file and finally execute it.

1.3.5.2Development Environments for OLE

With OLE, you can basically use the same environments as mentioned under 1.3.5.1. But keep in mind that, as you will not be using Java but instead programming the scripting language of your choice “directly”, you are free to choose scripting-language-specific IDEs such as the Zend Studio for PHP.

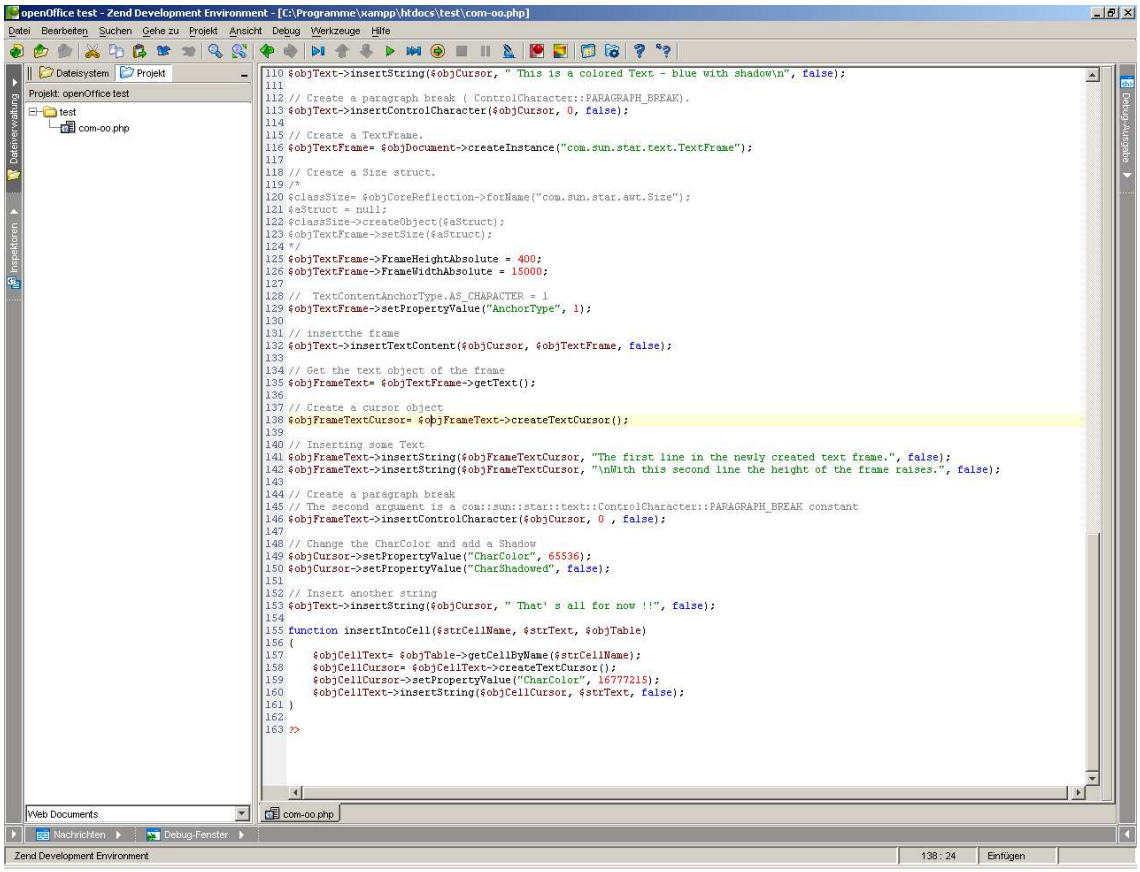

Fig. 3: Zend Studio: Ideal for PHP development

Currently, heavyweight applications such as NetBeans and Eclipse tend to slow down systems. Furthermore, Java-based applications still may have problems with garbage collection – also depending on how they are implemented – and therefore may waste your computer's memory requiring annoying IDE or system restarts. Thus, if you can use an IDE which does *not* build on Java, you are probably better off with that one.

1.3.6Using the documentation

If you are an experienced programmer, you might want an extensive reference at hand. The Java UNO interface which you will be programming is well documented in the Open Office SDK:

[Your local SDK path]/docs/common/ref/com/sun/star/module-ix.html

Also, you can use the Developer's guide (which the examples in this paper are mostly based upon):

[Your local SDK path]/docs/DevelopersGuide/DevelopersGuide.htm

The Developer's Guide is also available in PDF format (stored in the same directory).

1.4Concepts

1.4.1Java UNO

UNO (Universal Network Objects) is implemented in Open Office and provides a powerful bridge between Open Office and programming languages, especially Java. Where Java can be used to control an application, the Java BSF classes can be used too. BSF extends the horizon to a large variety of available scripting languages. Via the scripting languages' object-oriented capabilities, you are able to program the Java UNO interface without having to use a strong-typed, heavyweight programming language such as Java.

UNO can be accessed via Java, as in the Java/BSF approach, or directly via Microsoft's COM (Component Object Model) technology and OLE (Object Linking and Embedding) objects.

The drawback of (more or less) direct use of UNO classes is that you should have an idea of advanced object-oriented software design. The use of scripting languages for this task simplifies the handling of these objects but not the object model itself. A UNO user should be able to learn to deal with e.g. the factory pattern which is thoroughly used in the interface. The "new" operator is hardly ever used (because you usually get sub-objects from provider objects), and you will have to use structs.

1.4.2Reflection

However, of all the scripting languages, PHP has a major drawback here: Neither in Version 4 nor in Version 5 the instantiation of a struct via the Reflection API has succeeded. PHP programmers have to work around this issue by making use of the conventional set methods, where available (see Chapter 3.3).

Reflection mechanisms are also needed if the scripting language of your choice cannot be used to work with objects like Java, especially where Java simplifies things providing special mechanisms. For example, in some languages you cannot create an array like this:

```
import com.sun.star.beans.*;

PropertyValue myArray = new PropertyValue[10];
```

Instead, you might have to use (in Javascript):

```
myArray =
  java.lang.reflect.Array.newInstance(Packages.com.sun.star.beans.PropertyValue, 10);
```

Or, where Java lets you indicate a class name like this:

```
import com.sun.star.uno.*;

xUnoUrlResolver =
 UnoRuntime.queryInterface(com.sun.star.bridge.XunoUrlResolver, urlResolver);
```

You have to work around it like this (in Javascript):

```
xUnoUrlResolver =
 Packages.com.sun.star.uno.UnoRuntime.queryInterface(
```

java.lang.Class.forName("com.sun.star.bridge.XUnoUrlResolver"), urlResolver);

or like this (in Rexx):

```
unoRuntime = .unoRuntime.class~new()
unoResolverName =
 .bsf4rexx~Class.class~forName("com.sun.star.bridge.XUnoUrlResolver")
urlResolver = unoRuntime~queryInterface(unoResolverName, xUrlResolver)
```

When you are restricted to the OLE world, you can rely on the Open Office reflection classes:

[Your local SDK path]/
docs/DevelopersGuide/AdvancedUNO/AdvancedUNO.htm#
1+2+3+UNO+Reflection+API

They can be used for instantiating Open Office objects such as structs, provide information on UNO types and also – mind the scripting languages' automatic type casting! – do explicit type conversions (especially with OLE programming where Java type conversions are not available).

1.4.3Programming obstacles

When using scripting languages for Java UNO programming, most obstacles derive from getting information on how to

- Access external Java classes
- Access fields vs. methods
- Access static methods or fields vs. normal methods or fields
- Point the scripting language which method to choose (when there are multiple method signatures due to method overloading in an external Java class)

1.4.3.1Javascript solutions

In Javascript, these issues are implemented in a more comfortable way:

You can call external Java classes using (for instance) *java.lang.System* (for Java's on-board packages) or *Packages.com.sun.star.beans.PropertyValue* (for external packages).

Fields and methods - static or non-static – are accessed as they are in Java.

1.4.3.2Rexx solutions

Object Rexx with the bsf4rexx extension provides you with these possibilities:

Accessing external java classes:

myObject = .bsf~new("com.sun.star.beans.PropertyValue")

Accessing fields (one of various ways to do this):

theName = bsf("getFieldValue", myJavaClass, "name")
staticValue = bsf("getStaticValue", "com.xyz.abc.ClassName", "STATIC_FIELD")

Type hinting:

cellProperties~bsf.invokeStrict("setPropertyValue", "ST", "CellStyle", "ST", "My Style")

In this case, setPropertyValue has different signatures, and therefore the automatic casting mechanism is unable to decide which one to choose. By using the *invokeStrict* method, you explicitly declare that “CellStyle” and “My Style” are to be regarded as Strings (“ST”).

For more information on Object Rexx, see [IBM05] and [RexxLA05].

Details on BSF4Rexx is available from the University of Essen [Flats01] and, more recent information, in the course slides of Prof. Dr. Flatscher’s Java automation class [Flats05].

2Examples for scripting with the Bean Scripting Framework

The scripts in this section use this header code to connect to Open Office via UNO. Include the corresponding lines in the beginning of your files.

Rhino connection header:

```
// [1][BSF][Rhino]
// JavaScript connection header

// If your Open Office listens on another port than 8100,
// change the following line.
unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.ServiceManager";

// Connect to Open Office
xLocalContext =
        Packages.com.sun.star.comp.helper.Bootstrap.createInitialComponentContext(null);
xLocalServiceManager = xLocalContext.getServiceManager();
urlResolver =
        xLocalServiceManager.createInstanceWithContext(
        "com.sun.star.bridge.UnoUrlResolver", xLocalContext);
xUnoUrlResolver =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.bridge.XUnoUrlResolver"), urlResolver);
initialObject = xUnoUrlResolver.resolve(unoUrl);
xPropertySet =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.beans.XPropertySet"), initialObject);
context = xPropertySet.getPropertyValue("DefaultContext");
mxRemoteContext =
Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.uno.XComponentContext"), context);
mxRemoteServiceManager = mxRemoteContext.getServiceManager();

// Retrieve the Desktop object, we need its XComponentLoader interface
// to load a new document
desktop =
        mxRemoteServiceManager.createInstanceWithContext(
        "com.sun.star.frame.Desktop", mxRemoteContext);
xComponentLoader =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.frame.XComponentLoader"), desktop);

// Ready for action ...
```

```
-- [2][BSF][Rexx]
-- Rexx connection header

-- If your Open Office listens on another port than 8100,
-- change the following line.
unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.NamingService"

-- Load UNO Runtime for later use

.bsf~import("unoRuntime.class", "com.sun.star.uno.UnoRuntime")
unoRuntime = .unoRuntime.class~new()

-- Connect to Open Office

.bsf~import("bootstrap.class", "com.sun.star.comp.helper.Bootstrap")
bootstrap = .bootstrap.class~new()
xComponentContext = bootstrap~createInitialComponentContext(.NIL)

xLocalServiceManager = xComponentContext~getServiceManager()
xUrlResolver = ,
        xLocalServiceManager~createInstanceWithContext(,
        "com.sun.star.bridge.UnoUrlResolver", xComponentContext)

unoResolverName = .bsf4rexx~Class.class~forName("com.sun.star.bridge.XUnoUrlResolver")
urlResolver = unoRuntime~queryInterface(unoResolverName, xUrlResolver)

rInitialObject = urlResolver~resolve(unoUrl)
```

```
namingServiceName = .bsf4rexx~Class.class~forName("com.sun.star.uno.XNamingService")
rName = unoRuntime~queryInterface(namingServiceName, rInitialObject)

rXsmgr = rName~getRegisteredObject("StarOffice.ServiceManager")
msfName = .bsf4rexx~Class.class~forName("com.sun.star.lang.XMultiServiceFactory")
xMsf = unoRuntime~queryInterface(msfName, rXsmgr)

-- Retrieve the Desktop object, we need its XComponentLoader interface
-- to load a new document

oInterface = xMsf~createInstance("com.sun.star.frame.Desktop")
xDesktopName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XDesktop")
oDesktop = unoRuntime~queryInterface(xDesktopName, oInterface)
xComponentLoaderName = ,
       .bsf4rexx~Class.class~forName("com.sun.star.frame.XComponentLoader")
xComponentLoader = unoRuntime~queryInterface(xComponentLoaderName, oDesktop)

-- Ready for action ...


::requires "BSF.cls"
```

2.1General document operations

2.1.1Loading

2.1.1.1Opening a blank document

In this case you have to specify which application to open. This is done by using the URL scheme “private:factory/*application*”, where *application* may be:

scalc	for the spreadsheet application
swriter	for the word processor
simpress	for the presentation designer
sdraw	for the drawing tool

```
// [3][BSF][Rhino]
// Open a blank text document

loadProps = new Array();      // We need no properties
xWriterComponent = xComponentLoader.loadComponentFromURL(
       "private:factory/swriter", "_blank", 0, loadProps);
```

```
-- [4][BSF][Rexx]
-- Open a blank text document

-- We need no properties
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0)

xWriterComponent = xComponentLoader~loadComponentFromURL( ,
       "private:factory/swriter", "_blank", 0, loadProps)


::requires "BSF.cls"
```

2.1.1.2Opening an existing document

Here we do not need to specify the application because it can be guessed from the document we want to be loaded. The URL scheme is “file:///*path*”, where *path* consists of the directory path and the file name.

```
// [5][BSF][Rhino]
// Open an existing text document
```

```
loadProps = new Array();      // We need no properties
xWriterComponent = xComponentLoader.loadComponentFromURL(
       "file:///c:/documents/myfile.sxw", "_blank", 0, loadProps);
```

```
-- [6][BSF][Rexx]
-- Open an existing text document

-- We need no properties
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0)

xWriterComponent = xComponentLoader~loadComponentFromURL( ,
       "file:///c:/documents/myfile.sxw", "_blank", 0, loadProps)


::requires "BSF.cls"
```

Alternatively to “file:///”, the URL may also be an “http://” or an “ftp://” URL. Note that there are three slashes in “file:///”.

2.1.2Saving

You can save to every file format for which a filter is defined in the “Filters” section of

[Open Office path]share/registry/data/org/openoffice/Office/TypeDetection.xcu.

The filter’s identifying string is value that comes after “<node oor:name=”.

The following example saves the current document to a MS Word 97 file.

First we need to get a Storable interface in the Writer context. The Writer was returned when we opened up the word processor window in example 4 or 6 (resp. 5 or 7). The Storable provides the method “storeAsURL” which can be given an array of PropertyValues. We define a “FilterName” property and give it the name of the desired filter according to the .xcu file above.

```
// [7][BSF][Rhino][F]
// This script is a continuation of script #3.
// Save current text document in Word 97 format

xStorable =    // get a "Storable" interface in the context of the Writer application
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.frame.XStorable"), xWriterComponent);

storeProps = new Array();     // This time we need to define a property: The filter name
storeProps[0] = new Packages.com.sun.star.beans.PropertyValue();
storeProps[0].Name = "FilterName";
storeProps[0].Value = "MS Word 97";

storeUrl = "file:///c:/test.doc";    // Linux/UNIX users change the filename!
xStorable.storeAsURL(storeUrl, storeProps);
```

```
-- [8][BSF][Rexx]
-- Save current text document in Word 97 format

-- Get a "Storable" interface in the context of the Writer application
xStorableName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XStorable")
xStorable = unoRuntime~queryInterface(xStorableName, xWriterComponent)

-- This time we need to define a property: The filter name
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
storeProps = .bsf~createArray(propertyValueName, 1)
storeProps[1] = .bsf~new("com.sun.star.beans.PropertyValue")
storeProps[1]~bsf.setFieldValue("Name", "FilterName")
storeProps[1]~bsf.setFieldValue("Value", "MS Word 97")

storeUrl = "file:///c:/test.doc"     -- Linux/UNIX users change the filename!
xStorable~storeAsURL(storeUrl, storeProps)
```

```
::requires "BSF.cls"
```

2.1.3Printing

Printing works in analogy to saving: First, we get a Printable interface in the application's context (here: Writer), then we define properties.

The first properties array defines the printer "Name", determined by its name in the operating system environment. The array is set via the Printable's "setPrinter" method.

The second properties array holds the print options such as the number of pages ("Pages"). Note that despite the fact that we are programming in a loosely typed language, the underlying Java engine expects us to hand over the (numeric) value as a String (printOpts[0].Value = "1").

Finally, we start the printing process with the Printable's "print" method which is given the print options as an argument.

Note that the array index enumeration in Rexx starts with 1 (where in Javascript, it is 0).

```
// [9][BSF][Rhino]
// Print a text document

// Get a "Printable" interface in the context of the Writer application
xPrintable =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.view.XPrintable"), xWriterComponent);

// We need to define a property: The printer name
// This is a printer's name as known in your system.
printerDesc = new Array();
printerDesc[0] = new Packages.com.sun.star.beans.PropertyValue();
printerDesc[0].Name = "Name";
printerDesc[0].Value = "HP PSC 950";  // Change this to your printer name!
xPrintable.setPrinter(printerDesc);

// Print options are properties, too.
// We want to define which pages to print
printOpts = new Array();
printOpts[0] = new Packages.com.sun.star.beans.PropertyValue();
printOpts[0].Name = "Pages";
printOpts[0].Value = "1";

// This prints the (empty) test page
xPrintable.print(printOpts);
```

```
-- [10][BSF][Rexx]
-- Print a text document

-- Get a "Printable" interface in the context of the Writer application
xPrintableName = .bsf4rexx~Class.class~forName("com.sun.star.view.XPrintable")
xPrintable = unoRuntime~queryInterface(xPrintableName, xWriterComponent)

-- We need to define a property: The printer name
-- This is a printer's name as known in your system.
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
printerDesc = .bsf~createArray(propertyValueName, 1)
printerDesc[1] = .bsf~new("com.sun.star.beans.PropertyValue")
printerDesc[1]~bsf.setFieldValue("Name", "Name")
printerDesc[1]~bsf.setFieldValue("Value", "HP PSC 950")
                                       -- ^ Change this to your printer name!
xPrintable~setPrinter(printerDesc)

-- Print options are properties, too.
-- We want to define which pages to print
printOpts = .bsf~createArray(propertyValueName, 1)
printOpts[1] = .bsf~new("com.sun.star.beans.PropertyValue")
```

```
printOpts[1]~bsf.setFieldValue("Name", "Pages")
printOpts[1]~bsf.setFieldValue("Value", "1")

-- This prints the (empty) test page
xPrintable~print(printOpts);


::requires "BSF.cls"
```

2.2Spreadsheet

In this chapter you will find a couple of spreadsheet-specific examples. As a prerequisite, you will need the following header lines in the beginning of your files:

```
// [11][BSF][Rhino]
// If your Open Office listens on another port than 8100,
// change the following line.
unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.ServiceManager";

// Connect to Open Office
xLocalContext =
        Packages.com.sun.star.comp.helper.Bootstrap.createInitialComponentContext(null);
xLocalServiceManager = xLocalContext.getServiceManager();
urlResolver =
        xLocalServiceManager.createInstanceWithContext(
        "com.sun.star.bridge.UnoUrlResolver", xLocalContext);
xUnoUrlResolver =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.bridge.XUnoUrlResolver"), urlResolver);
initialObject = xUnoUrlResolver.resolve(unoUrl);
xPropertySet =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.beans.XPropertySet"), initialObject);
context = xPropertySet.getPropertyValue("DefaultContext");
mxRemoteContext =
Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.uno.XComponentContext"), context);
mxRemoteServiceManager = mxRemoteContext.getServiceManager();

// Retrieve the Desktop object, we need its XComponentLoader interface
// to load a new document
desktop =
        mxRemoteServiceManager.createInstanceWithContext(
        "com.sun.star.frame.Desktop", mxRemoteContext);
xComponentLoader =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.frame.XComponentLoader"), desktop);

// Open a blank spreadsheet document
// If you wish to open an existing document, use a different URL (see chapter 2.1.1.2)
// You can address single sheets in your existing document by indicating a different
// index number (see below ”oIndexSheets.getByIndex(0)”).
loadProps = new Array();      // We need no properties
xSheetComponent = xComponentLoader.loadComponentFromURL(
        "private:factory/scalc", "_blank", 0, loadProps);
xDocument = UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.sheet.XSpreadsheetDocument”),
        xSheetComponent);
xSheets = xDocument.getSheets();
xIndexSheets =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.container.XIndexAccess"), xSheets);
xSheet =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.sheet.XSpreadsheet"),
        xIndexSheets.getByIndex(0));

// Ready for action ...
```

```
-- [12][BSF][Rexx]
-- Object Rexx connection header for spreadsheet examples

-- If your Open Office listens on another port than 8100,
-- change the following line.
unoUrl = "uno:socket,host=localhost,port=8100;urp;StarOffice.NamingService"

-- Load UNO Runtime for later use
```

```
unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")

-- Connect to Open Office
xComponentContext = .bsf~new("com.sun.star.comp.helper.Bootstrap")
~createInitialComponentContext(.nil)
xUrlResolver = xComponentContext~getServiceManager() ~createInstanceWithContext
("com.sun.star.bridge.UnoUrlResolver", xComponentContext)

unoResolverName = .bsf4rexx~Class.class~forName("com.sun.star.bridge.XUnoUrlResolver")
urlResolver = unoRuntime~queryInterface(unoResolverName, xUrlResolver)

rInitialObject = urlResolver~resolve(unoUrl)
namingServiceName = .bsf4rexx~Class.class~forName("com.sun.star.uno.XNamingService")
rName = unoRuntime~queryInterface(namingServiceName, rInitialObject)

rXsmgr = rName~getRegisteredObject("StarOffice.ServiceManager")
msfName = .bsf4rexx~Class.class~forName("com.sun.star.lang.XMultiServiceFactory")
xMsf = unoRuntime~queryInterface(msfName, rXsmgr)

-- Retrieve the Desktop object, we need its XComponentLoader interface
-- to load a new document
oInterface = xMsf~createInstance("com.sun.star.frame.Desktop")
xDesktopName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XDesktop")
oDesktop = unoRuntime~queryInterface(xDesktopName, oInterface)
xComponentLoaderName = .bsf4rexx~Class.class~forName
("com.sun.star.frame.XComponentLoader")
xComponentLoader = unoRuntime~queryInterface(xComponentLoaderName, oDesktop)

-- Open a blank spreadsheet document
-- If you wish to open an existing document, use a different URL (see chapter 2.1.1.2)
-- You can address single sheets in your existing document by indicating a different
-- index number (see below "oIndexSheets~getByIndex(0)").

propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0)  -- We need no properties
xSheetComponent = ,
       xComponentLoader~loadComponentFromURL("private:factory/scalc", "_blank", 0,
loadProps)
xSpreadsheetDocumentName = .bsf4rexx~Class.class~forName
("com.sun.star.sheet.XSpreadsheetDocument")
xDocument = unoRuntime~queryInterface(xSpreadsheetDocumentName, xSheetComponent)
xSheets = xDocument~getSheets

xIndexAccessName = .bsf4rexx~Class.class~forName("com.sun.star.container.XIndexAccess")
xIndexSheets = unoRuntime~queryInterface(xIndexAccessName, xSheets)
xSheetName = .bsf4rexx~Class.class~forName("com.sun.star.sheet.XSpreadsheet")
xSheet = unoRuntime~queryInterface(xSheetName, xIndexSheets~getByIndex(0))

-- Ready for action ...


::requires "BSF.cls"
```

2.2.1Inserting formulas and numeric values

In order to work with spreadsheet cell contents, we have to keep two things in mind (this explanation is simplified):

1. A cell can contain a (numeric) *value* or something else (i.e. any character string) which would be called a *formula* (without regard to whether it really is a formula). This distinction is essential for choosing the right write method.
2. Addressing cells:
   You can get cells out of
   a. a spreadsheet
   b. a range of cells within a spreadsheet.
   This implies that you can get a *cell range* out of a spreadsheet.

   Cells can be addressed by
   a. their cell names, such as “B5”, as in the spreadsheet application
   b. their numeric indices, such as “2”/”6”

Cell ranges can be addressed just as in the spreadsheet application, e.g. like "A1:C5".

If you do a (perhaps nested) loop over a range of cells, thereby addressing each cell in the cell range, you will most likely prefer to use the numeric cell addressing, such as:

```
for(i=0; i<10; i++)
{
        for(j=0; j<10; j++)
        {
                // do something with cell [i][j]
        }
}
```

Note: In Rexx, Visual Basic and some other languages, index enumeration starts at 1, not at 0.

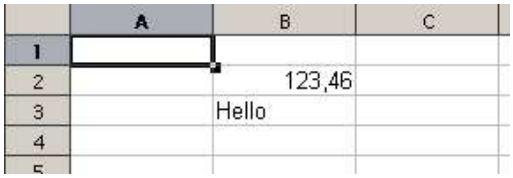

Fig. 4: Insertion of both numbers and text

On the other hand, if you want to grab cells, you might prefer the spreadsheet user's notation: "get cell by *position F2*" or "get the cell range by *name B12:H20*".

In the example, we create an "insert" function in order to simplify future cell insertions. This function will be used in some later examples.

```
// [13][BSF][Rhino]
// A function for inserting values or formulas into cells

// Test for the insert function:
// (1) insert a value (number)
insert(1, 1, 123.456, xSheet, true);
// (2) insert a text string (formula)
insert(1, 2, "Hello", xSheet, false);

// A function for inserting values or formulas into cells
// Parameters:
//      x, y            cell coordinates
//      content                 value or formula to insert
//      container       the spreadsheet or cell range where the cell resides
//      isValue                 (true or false) indicates whether the content should be
//                      regarded as a number or a character string
function insert(x, y, content, container, isValue)
{
        oCell = container.getCellByPosition(x, y);
        if(isValue)
                oCell.setValue(new java.lang.Float(content).floatValue());
        else
                oCell.setFormula(content);
}
```

```
-- [14][BSF][Rexx]
-- A function for inserting values or formulas into cells

-- Test for the insert function:
-- (1) insert a value (number)
call insert 1, 1, 123.456, xSheet, 1
-- (2) insert a text string (formula)
call insert 1, 2, "Hello", xSheet, 0

EXIT
```

```
-- A function for inserting values or formulas into cells
-- Parameters:
--      x, y           cell coordinates
--      content               value or formula to insert
--      container      the spreadsheet or cell range where the cell resides
--      isValue               (1 [true] or 0) indicates whether the content should be
--                     regarded as a number or a character string
insert: PROCEDURE
use arg x, y, content, container, isValue
        oCell = container~getCellByPosition(x, y)
        if isValue then
                oCell~setValue(.bsf~new("java.lang.Float", content) ~floatValue)
        else
                oCell~setFormula(content)
        RETURN


::requires "BSF.cls"
```

We need to insert content into cells in this relatively complicated manner because numeric values (which you want to be able to do calculations with) need to be floating point numbers. In this case, we have to explicitly instantiate a Java Float object. As script programmers may not like to think in Java terms, we will use this function for later examples. It just needs to be told whether a value is meant as a numeric value or not (fifth argument).

Now we insert some values and some data into the spreadsheet:

```
// [15][BSF][Rhino]
// Insert formulas using the insert function

// First, insert some numeric data
for(i=0; i<10; i++)
{
        for(j=0; j<10; j++)
                insert(i, j, i*j, xSheet, true);
}

// Insert "formulas" using the insert function
// First, insert a real formula:
insert(1, 10, "=SUM(B1:B10)", xSheet, false);

// Second, insert a text string
insert(2, 10, "-> This is the sum!", xSheet, false);
```

```
-- [16][BSF][Rexx]
-- Insert formulas using the insert function

-- first, insert some numeric data
do i=0 to 9
        do j=0 to 9
                call insert i, j, i*j, xSheet, 1
        end
end

-- Insert "formulas" using the insert function
-- First, insert a real formula:
call insert 1, 10, "=SUM(B1:B10)", xSheet, 0

-- Second, insert a text string
call insert 2, 10, "-> This is the sum!", xSheet, 0


::requires "BSF.cls"
```

2.2.2Using your own formulas

As for formulas, you do not depend on built-in spreadsheet formulas. You can write your own formulas in a scripting language, too. In the following example we use a function that will calculate the average of all non-empty cells in a given range:

```
// [17][BSF][Rhino]
// A function to calculate the average not considering empty fields

// Test for function avgNonEmpty:

// (1) Insert dummy data
insert(0, 0, 5.2, xSheet, true);
insert(0, 2, 2.3, xSheet, true);

// (2) Calculate and print average
result = avgNonEmpty(0, 0, 0, 2, xSheet);
insert(0, 3, result, xSheet, true);

// (3) Insert some labels
insert(1, 1, "<- Remains empty", xSheet, false);
insert(1, 3, "<- The average", xSheet, false);

// A function to calculate the average not considering empty fields
function avgNonEmpty(fromx, fromy, tox, toy, container)
{
        sum = 0.0;
        fieldCount = 0;

        for(i=fromx; i<=tox; i++)
        {
                for(j=fromy; j<=toy; j++)
                {
                        currentCell = container.getCellByPosition(i, j);
                        currentValue = currentCell.getValue();
                        if(currentValue != "")
                        {
                                sum += currentValue;
                                fieldCount++;
                        }
                }
        }

        if(fieldCount > 0)
                return (sum/fieldCount);
        else
                return 0;
}
```

```
-- [18][BSF][Rexx]
-- A function to calculate the average not considering empty fields

-- Test for function avgNonEmpty:

-- (1) Insert dummy data
call insert 0, 0, 5.2, xSheet, 1
call insert 0, 2, 2.3, xSheet, 1

-- (2) Calculate and print average
result = avgNonEmpty(0, 0, 0, 2, xSheet)
call insert 0, 3, result, xSheet, 1

-- (3) Insert some labels
call insert 1, 1, "<- Remains empty", xSheet, 0
call insert 1, 3, "<- The average", xSheet, 0

EXIT

-- A function to calculate the average not considering empty fields
avgNonEmpty: PROCEDURE
use arg fromx, fromy, tox, toy, container
        sum = 0
        fieldCount = 0

        do i=fromx to tox
                do j=fromy to toy
                        currentValue = container~getCellByPosition(i, j) ~getValue
                        if currentValue <> "0.0" then
                        do
                                sum = sum + currentValue
                                fieldCount = fieldCount + 1
                        end
                end
        end

        if fieldCount > 0 then
                RETURN (sum/fieldCount)
        else
```

```
            RETURN 0

::requires "BSF.cls"
```

2.2.3Formatting cells

Unfortunately, formatting cells is not a simple task. Though, things get easier when you understand the underlying concept:

- A *cell style* is a *set* of definitions for a cell (e.g. a certain background color plus a font color plus a transparency setting).
- You can give every of your cell styles a name.
- All the *cell* styles together are the "CellStyles" family. There are other style families, too.
- All the style families reside in the document and are given out to you by the document's "StyleFamiliesSupplier".

The following example gets the cell style family, defines one new style and applies it to a cell range. (The numeric color values are the same RGB codes as used in HTML, just converted to decimal numbers: White = (hex) FFFFFF = (dec) 16777215.)

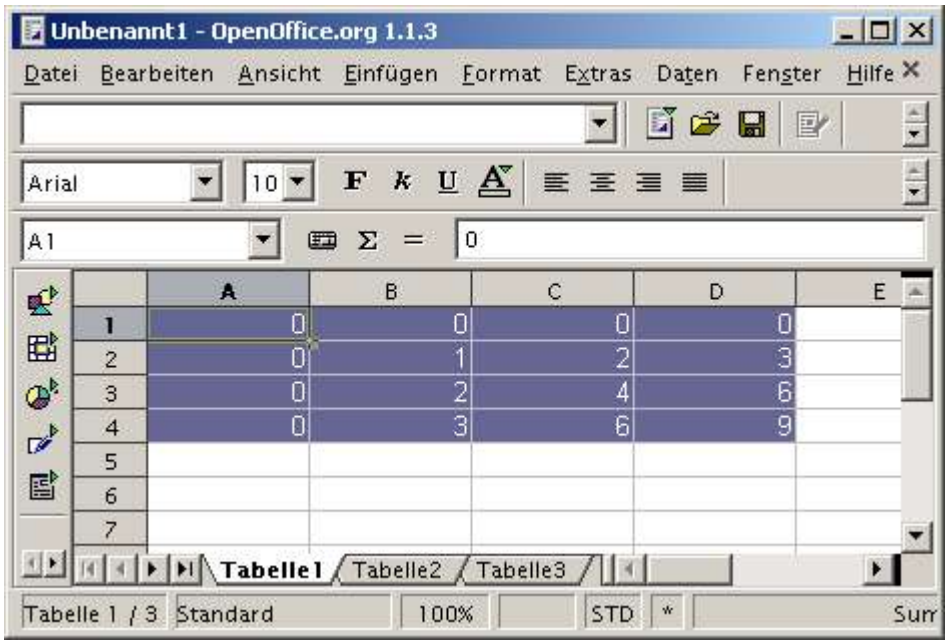

Fig. 5: Sample application of cell styles

```
// [19][BSF][Rhino]
// Cell formatting

// Insert some dummy data
for(i=0; i<4; i++)
{
        for(j=0; j<4; j++)
                insert(i, j, i*j, xSheet, true);
}

// Set style
xStyleFamiliesSupplier =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.style.XStyleFamiliesSupplier"), xDocument);
xStyleFamilies = xStyleFamiliesSupplier.getStyleFamilies();
xCellStyles =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.container.XNameAccess"),
xStyleFamilies.getByName("CellStyles"));
xDocumentMultiServiceFactory =
```

```
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.lang.XMultiServiceFactory"), xDocument);
oStyleFamilyNameContainer =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.container.XNameContainer"), xCellStyles);
style = xDocumentMultiServiceFactory.createInstance("com.sun.star.style.CellStyle");
oStyleFamilyNameContainer.insertByName("My Style", style);

oCellProperties1 =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), style);
oCellProperties1.setPropertyValue(
       "IsCellBackgroundTransparent", new java.lang.Boolean(false));
oCellProperties1.setPropertyValue(
       "CellBackColor", new java.lang.Integer(6710932));   // = hex 66 66 94
oCellProperties1.setPropertyValue(
       "CharColor", new java.lang.Integer(16777215));      // = hex FF FF FF

// Apply the defined style to a cell range (cells 0/0 to 3/3 = A1:C4)
cellRange = xSheet.getCellRangeByPosition(0, 0, 3, 3);
// get existing properties of cells in range
cellProperties =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), cellRange);
// redefine properties of cells in range
cellProperties.setPropertyValue("CellStyle", "My Style");
```

```
-- [20][BSF][Rexx]
-- Cell formatting

-- Insert some dummy data
do i=0 to 3
       do j=0 to 3
              call insert i, j, i*j, xSheet, 1
       end
end

-- Set style
xStyleFamiliesSupplierName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.style.XStyleFamiliesSupplier")
xStyleFamiliesSupplier = unoRuntime~queryInterface( ,
       xStyleFamiliesSupplierName, xDocument)
xStyleFamilies = xStyleFamiliesSupplier~getStyleFamilies
xNameAccessName = .bsf4rexx~Class.class~forName("com.sun.star.container.XNameAccess")
xCellStyles = unoRuntime~queryInterface(xNameAccessName, xStyleFamilies~getByName
("CellStyles"))
xDocumentMultiServiceFactoryName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.lang.XMultiServiceFactory")
xDocumentMultiServiceFactory = unoRuntime~queryInterface
(xDocumentMultiServiceFactoryName, xDocument)
oStyleFamilyNameContainerName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.container.XNameContainer")
oStyleFamilyNameContainer = unoRuntime~queryInterface( ,
       oStyleFamilyNameContainerName, xCellStyles)
style = xDocumentMultiServiceFactory~createInstance("com.sun.star.style.CellStyle")
oStyleFamilyNameContainer~insertByName("My Style", style)

oCellProperties1Name = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
oCellProperties1 = unoRuntime~queryInterface(oCellProperties1Name, style)
oCellProperties1~setPropertyValue( ,
       "IsCellBackgroundTransparent", .bsf~new("java.lang.Boolean", false))
oCellProperties1~setPropertyValue( ,
       "CellBackColor", .bsf~new("java.lang.Integer", 6710932))   -- = hex 66 66 94
oCellProperties1~setPropertyValue( ,
       "CharColor", .bsf~new("java.lang.Integer", 16777215))      -- = hex FF FF FF

-- Apply the defined style to a cell range (cells 0/0 to 3/3 = A1:C4)
cellRange = xSheet~getCellRangeByPosition(0, 0, 3, 3)

-- get existing properties of cells in range
cellPropertiesName = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
cellProperties = unoRuntime~queryInterface(cellPropertiesName, cellRange)

-- redefine properties of cells in range
--  !! Note for Rexx users: You must point BSF4Rexx to the right method
--     using the invokeStrict function with type indicators ("ST" for String)
cellProperties~bsf.invokeStrict("setPropertyValue", "ST", "CellStyle", "ST", "My Style")

EXIT


::requires "BSF.cls"
```

2.2.4Creating a 3-D chart

One of the spreadsheet application's most powerful and most impressing capabilities is that of generating 3-D style charts from numbers. Consider the in-line comments in the example.

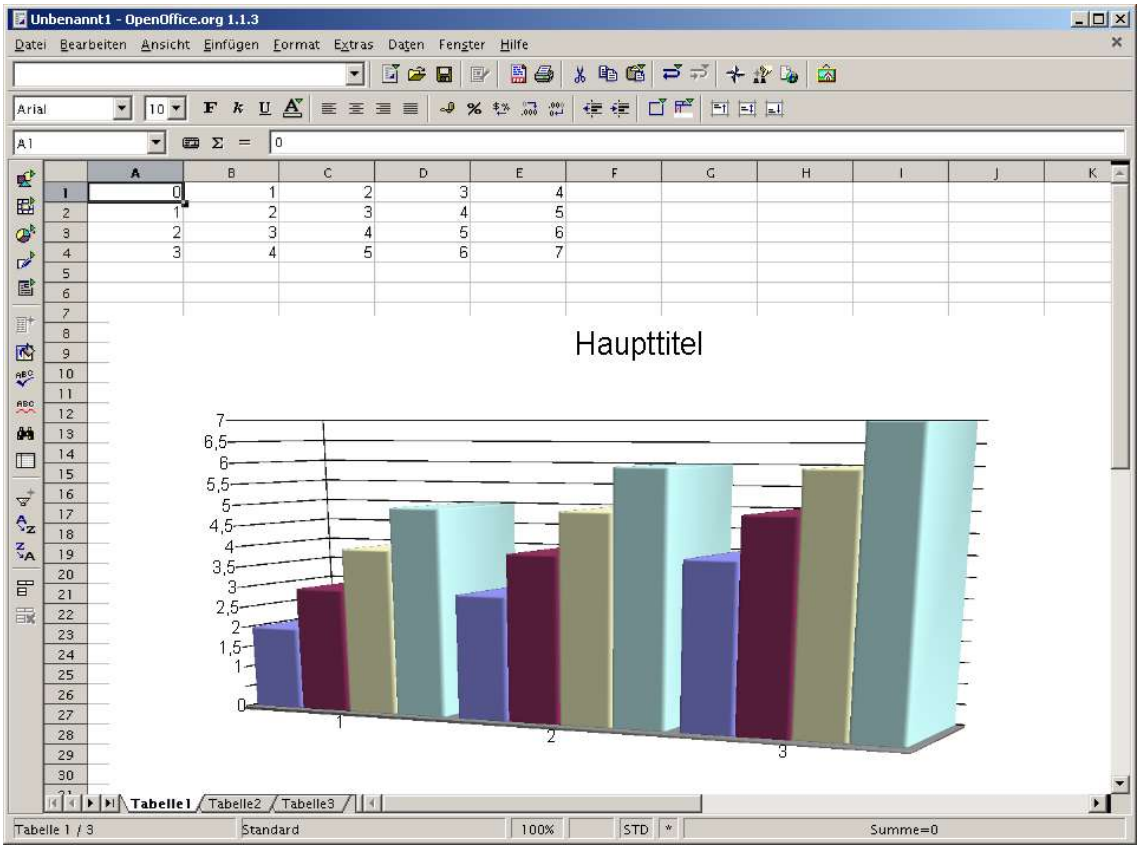

Fig. 6: Creation of an impressive 3-D chart with relatively little coding effort

```
// [21][BSF][Rhino]
// Create a chart in a spreadsheet

// As a prerequisite, we populate the spreadsheet with some numeric values:
for(i=0; i<5; i++)
{
        for(j=0; j<4; j++)
                insert(i, j, i+j, xSheet, true);
}

// First create the elements the chart consists of:
// (1) a frame (made from the Rectangle class)
oRect = new Packages.com.sun.star.awt.Rectangle();
oRect.X = 500;
oRect.Y = 3000;
oRect.Width = 25000;
oRect.Height = 11000;

// (2) the underlying data (the numeric values come from a cell range)
oRange =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.table.XCellRange"), xSheet);
myRange = oRange.getCellRangeByName("A1:E4");
oRangeAddr =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.sheet.XCellRangeAddressable"), myRange);

// Now get the chart collection from the Sheet's charts supplier and add a new chart

// Get the address of the underlying data (in our case a cell range)
```

```
myAddr = oRangeAddr.getRangeAddress();
oAddr = new Array();
oAddr[0] = myAddr;
// Get the supplier and its charts
oSupp =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.table.XTableChartsSupplier"), xSheet);
oCharts = oSupp.getCharts();
// Append a new chart to the collection
oCharts.addNewByName("Example", oRect, oAddr, true, true);

// change diagram properties: 2-D to 3-D
tempName =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.container.XNameAccess"), oCharts)
       .getByName("Example");
oChart =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.table.XTableChart"), tempName);
oEOS =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.document.XEmbeddedObjectSupplier"),
       oChart);
oInt = oEOS.getEmbeddedObject();
xChart =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.chart.XChartDocument"), oInt);
oDiag = xChart.getDiagram();

// change diagram to 3-D
oCPS =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), oDiag);
oCPS.setPropertyValue("Dim3D", new java.lang.Boolean(true));
```

```
-- [22][BSF][Rexx]
-- Create a chart in a spreadsheet

-- As a prerequisite, we populate the spreadsheet with some numeric values:
do i=0 to 4
       do j=0 to 3
              call insert i, j, i+j, xSheet, 1
       end
end

-- First create the elements the chart consists of:
-- (1) a frame (made from the Rectangle class)
oRect = .bsf~new("com.sun.star.awt.Rectangle")
oRect~bsf.setFieldValue("X", 500)
oRect~bsf.setFieldValue("Y", 3000)
oRect~bsf.setFieldValue("Width", 25000)
oRect~bsf.setFieldValue("Height", 11000)

-- (2) the underlying data (the numeric values come from a cell range)
xCellRangeName = .bsf4rexx~Class.class~forName("com.sun.star.table.XCellRange")
oRange = unoRuntime~queryInterface(xCellRangeName, xSheet)
myRange = oRange~getCellRangeByName("A1:E4")
xCellRangeAddressableName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.sheet.XCellRangeAddressable")
oRangeAddr = unoRuntime~queryInterface(xCellRangeAddressableName, myRange)

-- Now get the chart collection from the Sheet's charts supplier and add a new chart

-- Get the address of the underlying data (in our case a cell range)
myAddr = oRangeAddr~getRangeAddress
cellRangeAddressName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.table.CellRangeAddress")
oAddr = .bsf~createArray(cellRangeAddressName, 1)
oAddr[1] = myAddr
-- Get the supplier and its charts
xTableChartsSupplierName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.table.XTableChartsSupplier")
oSupp = unoRuntime~queryInterface(xTableChartsSupplierName, xSheet)
oCharts = oSupp~getCharts
-- Append a new chart to the collection
oCharts~addNewByName("Example", oRect, oAddr, true, true)

-- change diagram properties: 2-D to 3-D
xNameAccessName = .bsf4rexx~Class.class~forName("com.sun.star.container.XNameAccess")
tempName = unoRuntime~queryInterface(xNameAccessName, oCharts) ~getByName("Example")
xTableChartName = .bsf4rexx~Class.class~forName("com.sun.star.table.XTableChart")
oChart = unoRuntime~queryInterface(xTableChartName , tempName)
```

```
xEmbeddedObjectSupplierName = .bsf4rexx~Class.class~forName( ,
       "com.sun.star.document.XEmbeddedObjectSupplier")
oInt = unoRuntime~queryInterface(xEmbeddedObjectSupplierName, oChart) ~getEmbeddedObject
xChartDocumentName = .bsf4rexx~Class.class~forName("com.sun.star.chart.XChartDocument")
oDiag = unoRuntime~queryInterface(xChartDocumentName, oInt) ~getDiagram

-- change diagram to 3-D
xPropertySetName = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
oCPS = unoRuntime~queryInterface(xPropertySetName, oDiag)
oCPS~setPropertyValue("Dim3D", .bsf~new("java.lang.Boolean", true))


::requires "BSF.cls"
```

2.3Text processor

Working with the text processor is much like working with spreadsheets. The main difference is that you do not have cells where most of the content is placed but the “text” area which can be accessed in different ways.

2.3.1Text document navigation and manipulation

This example demonstrates how a text cursor is used for navigation in a text document. There are different types of cursors, depending on how you want to navigate. Consider the in-line comments.

Note that this example uses the standard connection header (not the spreadsheet-specific one).

```
// [23][BSF][Rhino]
// Text manipulation

// Open a blank text document
loadProps = new Array();      // We need no properties
xWriterComponent =
       xComponentLoader.loadComponentFromURL("private:factory/swriter", "_blank", 0,
loadProps);

// Get the document's text interface
xTextDocument =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.text.XTextDocument"), xWriterComponent);
xText = xTextDocument.getText();

// Now we could start writing into the document, for example:
// xText.setString("A few words ...");

// In order to set paragraph properties, we need a cursor for navigation.
// A cursor can be retrieved from the controller,
// the controller comes from the model,
// and the model comes from the component, i.e. the application.

// Get model from component
xModel =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.frame.XModel"), xWriterComponent);

// The model knows its controller
xController = xModel.getCurrentController();

// The controller gives us the TextViewCursor
// Query the viewcursor supplier interface
xViewCursorSupplier =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.text.XTextViewCursorSupplier"),
       xController);
```

```
// Get the cursor
xViewCursor = xViewCursorSupplier.getViewCursor();

// Set the appropriate property for paragraph style
xCursorPropertySet =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xViewCursor);
xCursorPropertySet.setPropertyValue("ParaStyleName", "Quotations");

// Print the current page number - we need the XPageCursor interface for this
xPageCursor =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.text.XPageCursor"), xViewCursor);

// This is written directly into the open document (see above).
xText.setString("The current page number is " + xPageCursor.getPage());

// If we had not retrieved the text interface from the xTextDocument (see above),
// we could now get it from the TextViewCursor.
// The cursor is an XTextRange and has therefore a method getText():
// xDocumentText = xViewCursor.getText();

// The text creates a model cursor from the viewcursor
xModelCursor = xText.createTextCursorByRange(xViewCursor.getStart());

// Now we could query XWordCursor, XSentenceCursor and XParagraphCursor
// or XDocumentInsertable, XSortable or XContentEnumerationAccess
// and work with the properties of com.sun.star.text.TextCursor

// In this case we just go to the end of the paragraph and add some text.
// Now get a paragraph cursor first.
xParagraphCursor =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.text.XParagraphCursor"), xModelCursor);

// Go to the end of the paragraph
xParagraphCursor.gotoEndOfParagraph(false);
xParagraphCursor.setString(" ***** Fin de semana! ******");
```

```
-- [24][BSF][Rexx]
-- Text manipulation

-- Open a blank text document
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0)  -- We need no properties
xWriterComponent = xComponentLoader~loadComponentFromURL("private:factory/swriter",
"_blank", 0, loadProps)

-- Get the document's text interface
xTextDocumentName = .bsf4rexx~Class.class~forName("com.sun.star.text.XTextDocument")
xTextDocument = unoRuntime~queryInterface(xTextDocumentName, xWriterComponent)
xText = xTextDocument~getText

-- Now we could start writing into the document, for example:
-- xText~setString("A few words ...")

-- In order to set paragraph properties, we need a cursor for navigation.
-- A cursor can be retrieved from the controller,
-- the controller comes from the model,
-- and the model comes from the component, i.e. the application.

-- Get model from component
xModelName = .bsf4rexx~Class.class~forName("com.sun.star.frame.XModel")
xModel = unoRuntime~queryInterface(xModelName, xWriterComponent)

-- The model knows its controller
xController = xModel~getCurrentController

-- The controller gives us the TextViewCursor
-- Query the viewcursor supplier interface
xTextViewCursorSupplierName = .bsf4rexx~Class.class~forName
("com.sun.star.text.XTextViewCursorSupplier")
xViewCursorSupplier = unoRuntime~queryInterface(xTextViewCursorSupplierName,
xController)

-- Get the cursor
xViewCursor = xViewCursorSupplier~getViewCursor

-- Set the appropriate property for paragraph style
xPropertySetName = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
xCursorPropertySet = unoRuntime~queryInterface(xPropertySetName, xViewCursor)
--  !! Note for Rexx users: You must point BSF4Rexx to the right method
```

```
--     using the invokeStrict function with type indicators ("ST" for String)
xCursorPropertySet~bsf.invokeStrict("setPropertyValue", "ST", "ParaStyleName", "ST",
"Quotations")

-- Print the current page number - we need the XPageCursor interface for this
xPageCursorName = .bsf4rexx~Class.class~forName("com.sun.star.text.XPageCursor")
xPageCursor = unoRuntime~queryInterface(xPageCursorName, xViewCursor)

-- This is written directly into the open document (see above).
xText~setString("The current page number is " xPageCursor~getPage)

-- If we had not retrieved the text interface from the xTextDocument (see above),
-- we could now get it from the TextViewCursor.
-- The cursor is an XTextRange and has therefore a method getText():
-- xDocumentText = xViewCursor~getText

-- The text creates a model cursor from the viewcursor
xModelCursor = xText~createTextCursorByRange(xViewCursor~getStart)

-- Now we could query XWordCursor, XSentenceCursor and XParagraphCursor
-- or XDocumentInsertable, XSortable or XContentEnumerationAccess
-- and work with the properties of com.sun.star.text.TextCursor

-- In this case we just go to the end of the paragraph and add some text.
-- Now get a paragraph cursor first.
xParagraphCursorName = .bsf4rexx~Class.class~forName
("com.sun.star.text.XParagraphCursor")
xParagraphCursor = unoRuntime~queryInterface(xParagraphCursorName, xModelCursor)

-- Go to the end of the paragraph
xParagraphCursor~gotoEndOfParagraph(false)
xParagraphCursor~setString(" ***** Fin de semana! ******")


::requires "BSF.cls"
```

2.4Drawings and presentations

Open Office provides a drawing application and a presentation designer which both work with the same drawing object classes. When translating the following example from Java to the scripting languages, the major obstacle was using the numbers provided by Java objects for calculations in the script and sending them back to Java objects.

In Javascript, a solution was to explicitly convert the “standard” data type values to Java Integers like this:

pageWidth = java.lang.Integer.parseInt(xPageProps.getPropertyValue("Width"));

While in Rexx, it was necessary to convert calculated values to pseudo-integers using Rexx’s “format” function:

shapeX = format(shapeX, , 0)

For an insight into the Java UNO object hierarchy, have a look at the in-line comments in the examples.

2.4.1Creating a drawing

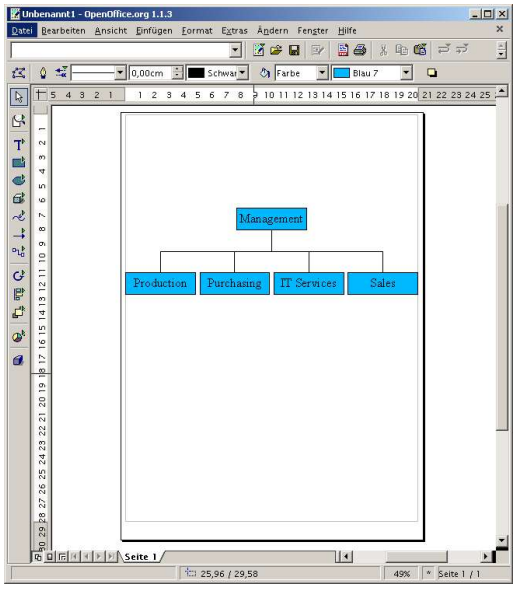

Fig. 7: The result of script 25/26

```
// [25][BSF][Rhino]
// Draw an organigram

// Open the "sdraw" application
loadProps = new Array();

xDrawComponent = xComponentLoader.loadComponentFromURL(
        "private:factory/sdraw", "_blank", 0, loadProps);

// Get draw page by index
xDrawPagesSupplier =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.drawing.XDrawPagesSupplier"),
       xDrawComponent);
xDrawPages = xDrawPagesSupplier.getDrawPages();
drawPage = xDrawPages.getByIndex(0);
```

```
xDrawPage =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.drawing.XDrawPage"), drawPage);

// Find out page dimensions
// Important: Integer values may not be regarded as Strings,
// therefore explicitly convert to integers via Integer.parseInt()!
xPageProps =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xDrawPage);
pageWidth =
        java.lang.Integer.parseInt(xPageProps.getPropertyValue("Width"));
pageHeight =
        java.lang.Integer.parseInt(xPageProps.getPropertyValue("Height"));
pageBorderTop =
        java.lang.Integer.parseInt(xPageProps.getPropertyValue("BorderTop"));
pageBorderLeft =
        java.lang.Integer.parseInt(xPageProps.getPropertyValue("BorderLeft"));
pageBorderRight =
        java.lang.Integer.parseInt(xPageProps.getPropertyValue("BorderRight"));
drawWidth = pageWidth - pageBorderLeft - pageBorderRight;
horCenter = pageBorderLeft + drawWidth / 2;

// Make up some data for organigram
orgUnits = new Array(2);
orgUnits[0] = new Array("Management"); // level 0
orgUnits[1] = new Array("Production", "Purchasing", "IT Services", "Sales"); // level 1
levelCount = new Array(1, 4);

// Calculate shape, sizes and positions
horSpace = 300;
verSpace = 3000;
shapeWidth = (drawWidth - (levelCount[1] - 1) * horSpace) / levelCount[1];
shapeHeight = pageHeight / 20;
shapeX = pageWidth / 2 - shapeWidth / 2;
levelY = 0;
xStartShape = null;


// Get document factory
xDocumentFactory =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.lang.XMultiServiceFactory"),
xDrawComponent);

// Creating and adding RectangleShapes and Connectors
for(level=0; level <= 1; level++)
{
        levelY = pageBorderTop + 2000 + level * (shapeHeight + verSpace);

        for(i = levelCount[level] - 1; i > -1; i--)
        {
                shapeX = horCenter -
                                (levelCount[level] * shapeWidth + (levelCount[level] - 1) *
                                horSpace) / 2 +
                                i * shapeWidth + i * horSpace;
                shape =
                        xDocumentFactory.createInstance(
                        "com.sun.star.drawing.RectangleShape");
                xShape =
                        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                        java.lang.Class.forName("com.sun.star.drawing.XShape"), shape);
                xShape.setPosition(new Packages.com.sun.star.awt.Point(shapeX, levelY));
                xShape.setSize(
                        new Packages.com.sun.star.awt.Size(shapeWidth, shapeHeight));
                xDrawPage.add(xShape);

                // Set the text
                xText =
                        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                        java.lang.Class.forName("com.sun.star.text.XText"), xShape);
                xText.setString(orgUnits[level][i]);

                // Memorize the root shape, for connectors
                if(level == 0 && i == 0)
                        xStartShape = xShape;

                // Add connectors for level 1
                if(level == 1)
                {
                        connector =
                                xDocumentFactory.createInstance(
                                "com.sun.star.drawing.ConnectorShape");
                        xConnector =
```

```
                                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                                java.lang.Class.forName("com.sun.star.drawing.XShape"),
                                connector);
                        xDrawPage.add(xConnector);
                        xConnectorProps =
                                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                                java.lang.Class.forName("com.sun.star.beans.XPropertySet"),
                                connector);
                        xConnectorProps.setPropertyValue("StartShape", xStartShape);
                        xConnectorProps.setPropertyValue("EndShape", xShape);

                        // glue point positions: 0=top 1=left 2=bottom 3=right
                        xConnectorProps.setPropertyValue(
                                "StartGluePointIndex", new java.lang.Integer(2));
                        xConnectorProps.setPropertyValue(
                                "EndGluePointIndex", new java.lang.Integer(0));
                }
        }
}
```

```
-- [26][BSF][Rexx]
-- Draw an organigram

-- Open the "sdraw" application
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
loadProps = .bsf~createArray(propertyValueName, 0)  -- We need no properties
xDrawComponent = xComponentLoader~loadComponentFromURL( ,
        "private:factory/sdraw", "_blank", 0, loadProps)

-- Get draw page by index
xDrawPagesSupplierName = .bsf4rexx~Class.class~forName( ,
        "com.sun.star.drawing.XDrawPagesSupplier")
xDrawPagesSupplier = unoRuntime~queryInterface(xDrawPagesSupplierName, xDrawComponent)
xDrawPages = xDrawPagesSupplier~getDrawPages
drawPage = xDrawPages~getByIndex(0)
xDrawPageName = .bsf4rexx~Class.class~forName("com.sun.star.drawing.XDrawPage")
xDrawPage = unoRuntime~queryInterface(xDrawPageName, drawPage)

-- Find out page dimensions
xPropertySetName = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
xPageProps = unoRuntime~queryInterface(xPropertySetName, xDrawPage)
pageWidth = xPageProps~getPropertyValue("Width")
pageHeight = xPageProps~getPropertyValue("Height")
pageBorderTop = xPageProps~getPropertyValue("BorderTop")
pageBorderLeft = xPageProps~getPropertyValue("BorderLeft")
pageBorderRight = xPageProps~getPropertyValue("BorderRight")
drawWidth = pageWidth - pageBorderLeft - pageBorderRight
horCenter = pageBorderLeft + drawWidth / 2

-- Make up some data for organigram
orgUnits.1.1 = "Management" -- level 0
orgUnits.2.1 = "Production"
orgUnits.2.2 = "Purchasing"
orgUnits.2.3 = "IT Services"
orgUnits.2.4 = "Sales"    -- level 1
levelCount.1 = 1
levelCount.2 = 4

-- Calculate shape, sizes and positions
horSpace = 300
verSpace = 3000
shapeWidth = (drawWidth - (levelCount.2 - 1) * horSpace) / levelCount.2
shapeHeight = pageHeight / 20
shapeX = pageWidth / 2 - shapeWidth / 2
levelY = 0
xStartShape = .nil

-- Get document factory

xMultiServiceFactoryName = .bsf4rexx~Class.class~forName( ,
        "com.sun.star.lang.XMultiServiceFactory")
xDocumentFactory = unoRuntime~queryInterface(xMultiServiceFactoryName, xDrawComponent)

-- Creating and adding RectangleShapes and Connectors
do level=1 to 2
        levelY = pageBorderTop + 2000 + level * (shapeHeight + verSpace)

        do i=levelCount.level to 1 by -1
                shapeX = horCenter - ,
                                (levelCount.level * shapeWidth + (levelCount.level - 1) * ,
                                horSpace) / 2 + ,
                                (i-1) * shapeWidth + (i-1) * horSpace
```

```
                shape = xDocumentFactory~createInstance( ,
                        "com.sun.star.drawing.RectangleShape")

                xShapeName = .bsf4rexx~Class.class~forName("com.sun.star.drawing.XShape")
                xShape = unoRuntime~queryInterface(xShapeName, shape)
                shapeX = format(shapeX, , 0)

                xShape~setPosition(.bsf~new("com.sun.star.awt.Point", shapeX, levelY))
                shapeWidth = format(shapeWidth, , 0)
                shapeHeight = format(shapeHeight, , 0)
                xShape~setSize( ,
                        .bsf~new("com.sun.star.awt.Size", shapeWidth, shapeHeight))
                xDrawPage~add(xShape)

                -- Set the text
                xTextName = .bsf4rexx~Class.class~forName("com.sun.star.text.XText")
                xText = unoRuntime~queryInterface(xTextName, xShape)
                xText~setString(orgUnits.level.i)

                -- Memorize the root shape, for connectors
                if level = 1 & i = 1 then xStartShape = xShape

                -- Add connectors for level 1
                if level = 2 then
                do
                        connector = xDocumentFactory~createInstance( ,
                                "com.sun.star.drawing.ConnectorShape")
                        xConnector = unoRuntime~queryInterface(xShapeName, connector)
                        xDrawPage~add(xConnector)
                        xConnectorProps = unoRuntime~queryInterface( ,
                                xPropertySetName, connector)
                        xConnectorProps~bsf.invokeStrict( ,
                                "setPropertyValue", "ST", "StartShape", "O", xStartShape)
                        xConnectorProps~bsf.invokeStrict(,
                                "setPropertyValue", "ST", "EndShape", "O", xShape)

                        -- glue point positions: 0=top 1=left 2=bottom 3=right
                        xConnectorProps~setPropertyValue( ,
                                "StartGluePointIndex", .bsf~new("java.lang.Integer", 2))
                        xConnectorProps~setPropertyValue( ,
                                "EndGluePointIndex", .bsf~new("java.lang.Integer", 0))
                end
        end
end


::requires "BSF.cls"
```

2.4.2Creating a presentation

Working even with simple presentations can get quite complex. The following example from the Developer's Guide was chosen because it introduces a couple of useful functions for handling shapes and transitions (see end of the scripts).

Also, it makes extensive use of constants which can be confusing, especially in Rexx scripts. Due to this fact, constants are stored in temporary variables for more transparency in the Rexx example, e.g.:

hide = bsf("getStaticValue", "com.sun.star.presentation.AnimationEffect", "HIDE")
xShapePropSet~setPropertyValue("Effect", hide)

Fig. 8: An interactive presentation created by a script

```
// [27][BSF][Rhino]
// Create a sample presentation

// Open the "impress" application
pPropValues =
        java.lang.reflect.Array.newInstance(
        Packages.com.sun.star.beans.PropertyValue, 1);
pPropValues[0] =              new Packages.com.sun.star.beans.PropertyValue();
pPropValues[0].Name =                 "Silent";
pPropValues[0].Value =        new java.lang.Boolean(true);

xDrawComponent =
        xComponentLoader.loadComponentFromURL(
                "private:factory/simpress", "_blank", 0, pPropValues);

// Create three presentation pages
xDrawPagesSupplier =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.drawing.XDrawPagesSupplier"),
xDrawComponent);
xDrawPages = xDrawPagesSupplier.getDrawPages();

for(i=0; i<3; i++)
        xDrawPages.insertNewByIndex(0);

// Set the slide transition for the first page
xPage =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.drawing.XDrawPage"), xDrawPages.getByIndex
(0));

xShapes =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.drawing.XShapes"), xPage );

// Set slide transition effect
setSlideTransition(
        xPage,
        Packages.com.sun.star.presentation.FadeEffect.FADE_FROM_RIGHT,
        Packages.com.sun.star.presentation.AnimationSpeed.FAST, 1, 0 );
        // automatic object and slide transition

// Create a rectangle that is placed on the top left of the page
xShapePropSet =
        createAndInsertShape(
                xDrawComponent,
                xShapes,
                new Packages.com.sun.star.awt.Point(1000, 1000),
                new Packages.com.sun.star.awt.Size(5000, 5000),
                "com.sun.star.drawing.RectangleShape");
xShapePropSet.setPropertyValue(
        "Effect",
        Packages.com.sun.star.presentation.AnimationEffect.WAVYLINE_FROM_BOTTOM);

// The following three properties provoke that the shape is dimmed to red
// after the animation has been finished
xShapePropSet.setPropertyValue("DimHide", new java.lang.Boolean(false));
xShapePropSet.setPropertyValue("DimPrevious", new java.lang.Boolean(true));
xShapePropSet.setPropertyValue("DimColor", new java.lang.Integer(0xff0000));

// Set the slide transition for the second page
xPage =
        Packages.com.sun.star.uno.UnoRuntime.queryInterface(
        java.lang.Class.forName("com.sun.star.drawing.XDrawPage"),
xDrawPages.getByIndex(1));
```

```
xShapes =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.drawing.XShapes"), xPage);

setSlideTransition(
       xPage,
       Packages.com.sun.star.presentation.FadeEffect.FADE_FROM_RIGHT,
       Packages.com.sun.star.presentation.AnimationSpeed.SLOW, 1, 0 );
       // automatic object and slide transition

// Create an ellipse that is placed on the bottom right of second page
xShapePropSet =
       createAndInsertShape(
       xDrawComponent,
       xShapes,
       new Packages.com.sun.star.awt.Point(21000, 15000),
       new Packages.com.sun.star.awt.Size(5000, 5000),
       "com.sun.star.drawing.EllipseShape");
xShapePropSet.setPropertyValue(
       "Effect",
       Packages.com.sun.star.presentation.AnimationEffect.HIDE);

// Create two objects for the third page:
// Clicking the first object lets the presentation jump
// to page one by using ClickAction.FIRSTPAGE,
// the second object lets the presentation jump to page two
// by using a ClickAction.BOOKMARK.
xPage =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.drawing.XDrawPage"),
xDrawPages.getByIndex(2));
xShapes =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.drawing.XShapes"), xPage);
setSlideTransition(
       xPage,
       Packages.com.sun.star.presentation.FadeEffect.ROLL_FROM_LEFT,
       Packages.com.sun.star.presentation.AnimationSpeed.MEDIUM, 2, 0 );
xShape =
       createShape(
              xDrawComponent,
              new Packages.com.sun.star.awt.Point(1000, 8000),
              new Packages.com.sun.star.awt.Size(5000, 5000),
              "com.sun.star.drawing.EllipseShape");
xShapes.add(xShape);
addPortion(xShape, "click to go", false);
addPortion(xShape, "to first page", true);
xShapePropSet =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xShape);
xShapePropSet.setPropertyValue(
       "Effect",
       Packages.com.sun.star.presentation.AnimationEffect.FADE_FROM_BOTTOM);
xShapePropSet.setPropertyValue(
       "OnClick",
       Packages.com.sun.star.presentation.ClickAction.FIRSTPAGE);
xShape =
       createShape(
              xDrawComponent,
              new Packages.com.sun.star.awt.Point(22000, 8000),
              new Packages.com.sun.star.awt.Size(5000, 5000),
              "com.sun.star.drawing.RectangleShape");
xShapes.add(xShape);
addPortion(xShape, "click to go", false);
addPortion(xShape, "to the second page", true);
xShapePropSet =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xShape);
xShapePropSet.setPropertyValue(
       "Effect",
       Packages.com.sun.star.presentation.AnimationEffect.FADE_FROM_BOTTOM);
xShapePropSet.setPropertyValue(
       "OnClick",
       Packages.com.sun.star.presentation.ClickAction.BOOKMARK);

// Set the name of page two, and use it with the bookmark action
xPage =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.drawing.XDrawPage"),
xDrawPages.getByIndex(1));
xPageName =
       Packages.com.sun.star.uno.UnoRuntime.queryInterface(
       java.lang.Class.forName("com.sun.star.container.XNamed"), xPage);
xPageName.setName("page two");
```

```
xShapePropSet.setPropertyValue("Bookmark", xPageName.getName());

// Start an endless presentation which is displayed in
// full-screen mode and placed on top
xPresSupplier = Packages.com.sun.star.uno.UnoRuntime.queryInterface
(java.lang.Class.forName("com.sun.star.presentation.XPresentationSupplier"),
xDrawComponent);
xPresentation = xPresSupplier.getPresentation();
xPresPropSet = Packages.com.sun.star.uno.UnoRuntime.queryInterface
(java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xPresentation);
xPresPropSet.setPropertyValue("IsEndless", new java.lang.Boolean(true));
xPresPropSet.setPropertyValue("IsAlwaysOnTop", new java.lang.Boolean(true));
xPresPropSet.setPropertyValue("Pause", new java.lang.Integer(0));
xPresentation.start();

// Auxiliary functions used in the example: ------------------------------

// This simple method applies the slide transition to a page
function setSlideTransition(xPage, eEffect, eSpeed, nChange, nDuration)
{
        // The following test is only sensible if you do not exactly know
        // what type of page xPage is, for this purpose it can been tested
        // if the com.sun.star.presentation.DrawPage service is supported
        xInfo =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.lang.XServiceInfo"), xPage );
        if(xInfo.supportsService("com.sun.star.presentation.DrawPage") == true)
        {
                try
                {
                        xPropSet =
                                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                                java.lang.Class.forName("com.sun.star.beans.XPropertySet"),
                                xPage );
                        xPropSet.setPropertyValue("Effect", eEffect );
                        xPropSet.setPropertyValue("Speed", eSpeed );
                        xPropSet.setPropertyValue("Change", new
                                java.lang.Integer(nChange));
                        xPropSet.setPropertyValue("Duration", new
                                java.lang.Integer(nDuration));
                }
                catch(ex)
                {
                        java.lang.System.err.println("setSlideTransition Exception:\n" +
                                ex.printStackTrace());
                }
        }
}

function createAndInsertShape(xDrawDoc, xShapes, aPos, aSize, sShapeType)
{
        xShape = createShape(xDrawDoc, aPos, aSize, sShapeType);
        xShapes.add(xShape);
        xPropSet =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xShape);

        return xPropSet;
}

function createShape(xDrawDoc, aPos, aSize, sShapeType)
{
        xFactory =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.lang.XMultiServiceFactory"),
                xDrawDoc);
        xObj = xFactory.createInstance(sShapeType);
        xShape =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.drawing.XShape"), xObj);
        xShape.setPosition(aPos);
        xShape.setSize(aSize);

        return xShape;
}

function addPortion(xShape, sText, bNewParagraph)
{
        xText =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.text.XText"), xShape);

        xTextCursor = xText.createTextCursor();
        xTextCursor.gotoEnd(false);
```

```
        if(bNewParagraph == true)
        {
                xText.insertControlCharacter(
                        xTextCursor,
                        Packages.com.sun.star.text.ControlCharacter.PARAGRAPH_BREAK,
                        false );
                xTextCursor.gotoEnd(false);
        }

        xTextRange =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.text.XTextRange"), xTextCursor);
        xTextRange.setString(sText);
        xTextCursor.gotoEnd(true);
        xPropSet =
                Packages.com.sun.star.uno.UnoRuntime.queryInterface(
                java.lang.Class.forName("com.sun.star.beans.XPropertySet"), xTextRange);

        return xPropSet;
}
```

```
-- [28][BSF][Rexx]
-- Create a sample presentation

-- Open the "sdraw" application
propertyValueName = .bsf4rexx~Class.class~forName("com.sun.star.beans.PropertyValue")
pPropValues = .bsf~createArray(propertyValueName, 1)

pPropValues[1] = .bsf~new("com.sun.star.beans.PropertyValue")
pPropValues[1]~bsf.setFieldValue("Name", "Silent")
pPropValues[1]~bsf.setFieldValue("Value", .bsf~new("java.lang.Boolean", true))

xDrawComponent = xComponentLoader~loadComponentFromURL( ,
        "private:factory/simpress", "_blank", 0, pPropValues)

-- Create three presentation pages
xDrawPagesSupplierName = .bsf4rexx~Class.class~forName( ,
        "com.sun.star.drawing.XDrawPagesSupplier")
xDrawPagesSupplier = unoRuntime~queryInterface(xDrawPagesSupplierName, xDrawComponent)
xDrawPages = xDrawPagesSupplier~getDrawPages

do 3
        xDrawPages~insertNewByIndex(0)
end

-- Set the slide transition for the first page
xDrawPageName = .bsf4rexx~Class.class~forName( ,
        "com.sun.star.drawing.XDrawPage")
xPage = unoRuntime~queryInterface(xDrawPageName, xDrawPages~getByIndex(0))

xShapesName = .bsf4rexx~Class.class~forName("com.sun.star.drawing.XShapes")
xShapes = unoRuntime~queryInterface(xShapesName, xPage)

-- Set slide transition effect
fade = bsf("getStaticValue", "com.sun.star.presentation.FadeEffect", "FADE_FROM_RIGHT")
fast = bsf("getStaticValue", "com.sun.star.presentation.AnimationSpeed", "FAST")
call setSlideTransition xPage, fade, fast, 1, 0
        -- automatic object and slide transition

-- Create a rectangle that is placed on the top left of the page
point = .bsf~new("com.sun.star.awt.Point", 1000, 1000)
size = .bsf~new("com.sun.star.awt.Size", 5000, 5000)
xShapePropSet = createAndInsertShape( ,
        xDrawComponent, xShapes, point, size, "com.sun.star.drawing.RectangleShape")
line = bsf( ,
"getStaticValue", "com.sun.star.presentation.AnimationEffect", "WAVYLINE_FROM_BOTTOM")
xShapePropSet~setPropertyValue("Effect", line)

-- The following three properties provoke that the shape is dimmed to red
-- after the animation has been finished
xShapePropSet~setPropertyValue("DimHide", .bsf~new("java.lang.Boolean", false))
xShapePropSet~setPropertyValue("DimPrevious", .bsf~new("java.lang.Boolean", true))
xShapePropSet~setPropertyValue("DimColor", .bsf~new("java.lang.Integer", 16711680))

-- Set the slide transition for the second page
xPage = unoRuntime~queryInterface(xDrawPageName, xDrawPages~getByIndex(1))
xShapes = unoRuntime~queryInterface(xShapesName, xPage)

slow = bsf("getStaticValue", "com.sun.star.presentation.AnimationSpeed", "SLOW")
call setSlideTransition xPage, fade, slow, 1, 0
        -- automatic object and slide transition
```

```
-- Create an ellipse that is placed on the bottom right of second page
point = .bsf~new("com.sun.star.awt.Point", 21000, 15000)
xShapePropSet = createAndInsertShape(,
       xDrawComponent, xShapes, point, size, "com.sun.star.drawing.EllipseShape")
hide = bsf("getStaticValue", "com.sun.star.presentation.AnimationEffect", "HIDE")
xShapePropSet~setPropertyValue("Effect", hide)

-- Create two objects for the third page:
-- Clicking the first object lets the presentation jump
-- to page one by using ClickAction.FIRSTPAGE,
-- the second object lets the presentation jump to page two
-- by using a ClickAction.BOOKMARK.
xPage = unoRuntime~queryInterface(xDrawPageName, xDrawPages~getByIndex(2))
xShapes = unoRuntime~queryInterface(xShapesName, xPage)
roll = bsf("getStaticValue", "com.sun.star.presentation.FadeEffect", "ROLL FROM LEFT")
medium = bsf("getStaticValue", "com.sun.star.presentation.AnimationSpeed", "MEDIUM")
call setSlideTransition xPage, roll, medium, 2, 0
point = .bsf~new("com.sun.star.awt.Point", 1000, 8000)
xShape = createShape(xDrawComponent, point, size, "com.sun.star.drawing.EllipseShape")

xShapes~add(xShape)
call addPortion xShape, "click to go", 0
call addPortion xShape, "to first page", 1

xPropertySetName = .bsf4rexx~Class.class~forName("com.sun.star.beans.XPropertySet")
xShapePropSet = unoRuntime~queryInterface(xPropertySetName, xShape)
fadebottom = bsf(,
"getStaticValue", "com.sun.star.presentation.AnimationEffect", "FADE_FROM_BOTTOM")
xShapePropSet~setPropertyValue("Effect", fadebottom)
firstpage = bsf("getStaticValue", "com.sun.star.presentation.ClickAction", "FIRSTPAGE")
xShapePropSet~setPropertyValue("OnClick", firstpage)
point = .bsf~new("com.sun.star.awt.Point", 22000, 8000)
xShape = createShape(xDrawComponent, point, size, "com.sun.star.drawing.RectangleShape")

xShapes~add(xShape)
call addPortion xShape, "click to go", 0
call addPortion xShape, "to the second page", 1
xShapePropSet = unoRuntime~queryInterface(xPropertySetName, xShape)
xShapePropSet~setPropertyValue("Effect", fadebottom)
bookmark = bsf("getStaticValue", "com.sun.star.presentation.ClickAction", "BOOKMARK")
xShapePropSet~setPropertyValue("OnClick", bookmark)

-- Set the name of page two, and use it with the bookmark action
xPage = unoRuntime~queryInterface(xDrawPageName, xDrawPages~getByIndex(1))
xNamedName = .bsf4rexx~Class.class~forName("com.sun.star.container.XNamed")
xPageName = unoRuntime~queryInterface(xNamedName, xPage)
xPageName~setName("page two")
xShapePropSet~bsf.invokeStrict( ,
       "setPropertyValue", "ST", "Bookmark", "ST", xPageName~getName)

-- Start an endless presentation which is displayed in
-- full-screen mode and placed on top
xPresentationSupplierName = .bsf4rexx~Class.class~forName
("com.sun.star.presentation.XPresentationSupplier")
xPresSupplier = unoRuntime~queryInterface(xPresentationSupplierName, xDrawComponent)
xPresentation = xPresSupplier~getPresentation
xPresPropSet = unoRuntime~queryInterface(xPropertySetName, xPresentation)
xPresPropSet~setPropertyValue("IsEndless", .bsf~new("java.lang.Boolean", true))
xPresPropSet~setPropertyValue("IsAlwaysOnTop", .bsf~new("java.lang.Boolean", true))
xPresPropSet~setPropertyValue("Pause", .bsf~new("java.lang.Integer", 0))
xPresentation~bsf.invoke("start")

EXIT

-- Auxiliary functions used in the example: -----------------------------

-- This simple method applies the slide transition to a page
setSlideTransition: PROCEDURE
use arg xPage, eEffect, eSpeed, nChange, nDuration
       unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")
       -- The following test is only sensible if you do not exactly know
       -- what type of page xPage is, for this purpose it can been tested
       -- if the com.sun.star.presentation.DrawPage service is supported
       xServiceInfoName = .bsf4rexx~Class.class~forName(,
              "com.sun.star.lang.XServiceInfo")
       xInfo = unoRuntime~queryInterface(xServiceInfoName, xPage)

       if xInfo~supportsService("com.sun.star.presentation.DrawPage") then
       do
              xPropertySetName = .bsf4rexx~Class.class~forName( ,
                     "com.sun.star.beans.XPropertySet")
              xPropSet = unoRuntime~queryInterface(xPropertySetName, xPage)
              xPropSet~setPropertyValue("Effect", eEffect)
              xPropSet~setPropertyValue("Speed", eSpeed)
```

```
                xPropSet~setPropertyValue( ,
                        "Change", .bsf~new("java.lang.Integer", nChange))
                xPropSet~setPropertyValue( ,
                        "Duration", .bsf~new("java.lang.Integer", nDuration))
        end
        RETURN

createAndInsertShape: PROCEDURE
use arg xDrawDoc, xShapes, aPos, aSize, sShapeType
        unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")
        xShape = createShape(xDrawDoc, aPos, aSize, sShapeType)
        xShapes~add(xShape)
        xPropertySetName = .bsf4rexx~Class.class~forName( ,
                "com.sun.star.beans.XPropertySet")
        xPropSet = unoRuntime~queryInterface(xPropertySetName, xShape)
        RETURN xPropSet

createShape: PROCEDURE
use arg xDrawDoc, aPos, aSize, sShapeType
        unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")
        xMultiServiceFactoryName = .bsf4rexx~Class.class~forName( ,
                "com.sun.star.lang.XMultiServiceFactory")
        xFactory = unoRuntime~queryInterface(xMultiServiceFactoryName, xDrawDoc)
        xObj = xFactory~createInstance(sShapeType);
        xShapeName = .bsf4rexx~Class.class~forName("com.sun.star.drawing.XShape")
        xShape = unoRuntime~queryInterface(xShapeName, xObj)

        xShape~setPosition(aPos)
        xShape~setSize(aSize)

        RETURN xShape

addPortion: PROCEDURE
use arg xShape, sText, bNewParagraph
        unoRuntime = .bsf~new("com.sun.star.uno.UnoRuntime")
        xTextName = .bsf4rexx~Class.class~forName("com.sun.star.text.XText")
        xText = unoRuntime~queryInterface(xTextName, xShape)

        xTextCursor = xText~createTextCursor
        xTextCursor~gotoEnd(false)
        if bNewParagraph = 1 then
        do
                pbreak = bsf( ,
        "getStaticValue", "com.sun.star.text.ControlCharacter", "PARAGRAPH_BREAK")
                xText~insertControlCharacter(xTextCursor, pbreak, false)
                xTextCursor~gotoEnd(false)
        end

        xTextRangeName = .bsf4rexx~Class.class~forName("com.sun.star.text.XTextRange")
        xTextRange = unoRuntime~queryInterface(xTextRangeName, xTextCursor)
        xTextRange~setString(sText)
        xTextCursor~gotoEnd(true)
        xPropertySetName = .bsf4rexx~Class.class~forName( ,
                "com.sun.star.beans.XPropertySet")
        xPropSet = unoRuntime~queryInterface(xPropertySetName, xTextRange)

        RETURN xPropSet


::requires "BSF.cls"
```

3Examples for scripting with OLE

OLE/COM examples require Microsoft Windows.

3.1Creating a text document using Visual Basic

The following example is taken from the Open Office SDK. It demonstrates how Swrite can be controlled from a Visual Basic Script via OLE. The script can be executed directly by double-clicking it in Windows Explorer.

```
'[29][OLE][VB]

'************************************************************************
'*
'*  $RCSfile: WriterDemo.vbs,v $
'*
'*  $Revision: 1.2 $
'*
'*  last change: $Author: hr $ $Date: 2003/06/30 15:51:54 $
'*
'*  The Contents of this file are made available subject to the terms of
'*  the BSD license.
'*
'*  Copyright (c) 2003 by Sun Microsystems, Inc.
'*  All rights reserved.
'*
'*  Redistribution and use in source and binary forms, with or without
'*  modification, are permitted provided that the following conditions
'*  are met:
'*  1. Redistributions of source code must retain the above copyright
'*     notice, this list of conditions and the following disclaimer.
'*  2. Redistributions in binary form must reproduce the above copyright
'*     notice, this list of conditions and the following disclaimer in the
'*     documentation and/or other materials provided with the distribution.
'*  3. Neither the name of Sun Microsystems, Inc. nor the names of its
'*     contributors may be used to endorse or promote products derived
'*     from this software without specific prior written permission.
'*
'*  THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS
'*  "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT
'*  LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS
'*  FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE
'*  COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT,
'*  INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING,
'*  BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS
'*  OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND
'*  ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR
'*  TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE
'*  USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.
'*
'************************************************************************

'The service manager is always the starting point
'If there is no office running then an office is started up
Set objServiceManager= WScript.CreateObject("com.sun.star.ServiceManager")

'Create the CoreReflection service that is later used to create structs
Set objCoreReflection= objServiceManager.createInstance
("com.sun.star.reflection.CoreReflection")

'Create the Desktop
Set objDesktop= objServiceManager.createInstance("com.sun.star.frame.Desktop")

'Open a new empty writer document
Dim args()
Set objDocument= objDesktop.loadComponentFromURL("private:factory/swriter", "_blank", 0,
args)

'Create a text object
Set objText= objDocument.getText

'Create a cursor object
Set objCursor= objText.createTextCursor
```

```
'Inserting some Text
objText.insertString objCursor, "The first line in the newly created text document." &
vbLf, false

'Inserting a second line
objText.insertString objCursor, "Now we're in the second line", false

'Create instance of a text table with 4 columns and 4 rows
Set objTable= objDocument.createInstance( "com.sun.star.text.TextTable")
objTable.initialize 4, 4

'Insert the table
objText.insertTextContent objCursor, objTable, false

'Get first row
Set objRows= objTable.getRows
Set objRow= objRows.getByIndex( 0)

'Set the table background color
objTable.setPropertyValue "BackTransparent", false
objTable.setPropertyValue                "BackColor", 13421823

'Set a different background color for the first row
objRow.setPropertyValue "BackTransparent", false
objRow.setPropertyValue "BackColor", 6710932

'Fill the first table row
insertIntoCell "A1","FirstColumn", objTable
insertIntoCell "B1","SecondColumn", objTable
insertIntoCell "C1","ThirdColumn", objTable
insertIntoCell "D1","SUM", objTable

objTable.getCellByName("A2").setValue 22.5
objTable.getCellByName("B2").setValue 5615.3
objTable.getCellByName("C2").setValue -2315.7
objTable.getCellByName("D2").setFormula"sum <A2:C2>"

objTable.getCellByName("A3").setValue 21.5
objTable.getCellByName("B3").setValue 615.3
objTable.getCellByName("C3").setValue -315.7
objTable.getCellByName("D3").setFormula "sum <A3:C3>"

objTable.getCellByName("A4").setValue 121.5
objTable.getCellByName("B4").setValue -615.3
objTable.getCellByName("C4").setValue 415.7
objTable.getCellByName("D4").setFormula "sum <A4:C4>"

'Change the CharColor and add a Shadow
objCursor.setPropertyValue "CharColor", 255
objCursor.setPropertyValue "CharShadowed", true

'Create a paragraph break
'The second argument is a com::sun::star::text::ControlCharacter::PARAGRAPH_BREAK
constant
objText.insertControlCharacter objCursor, 0 , false

'Inserting colored Text.
objText.insertString objCursor, " This is a colored Text - blue with shadow" & vbLf,
false

'Create a paragraph break ( ControlCharacter::PARAGRAPH_BREAK).
objText.insertControlCharacter objCursor, 0, false

'Create a TextFrame.
Set objTextFrame= objDocument.createInstance("com.sun.star.text.TextFrame")

'Create a Size struct.
Set objSize= createStruct("com.sun.star.awt.Size")
objSize.Width= 15000
objSize.Height= 400
objTextFrame.setSize( objSize)

' TextContentAnchorType.AS CHARACTER = 1
objTextFrame.setPropertyValue "AnchorType", 1

'insert the frame
objText.insertTextContent objCursor, objTextFrame, false

'Get the text object of the frame
Set objFrameText= objTextFrame.getText


'Create a cursor object
Set objFrameTextCursor= objFrameText.createTextCursor
```

```
'Inserting some Text
objFrameText.insertString objFrameTextCursor, "The first line in the newly created text
frame.", _
                          false
objFrameText.insertString objFrameTextCursor,
                          vbLf & "With this second line the height of the frame
raises.", false

'Create a paragraph break
'The second argument is a com::sun::star::text::ControlCharacter::PARAGRAPH_BREAK
constant
objFrameText.insertControlCharacter objCursor, 0 , false

'Change the CharColor and add a Shadow
objCursor.setPropertyValue "CharColor", 65536
objCursor.setPropertyValue "CharShadowed", false

'Insert another string
objText.insertString objCursor, " That's all for now !!", false

On Error Resume Next
If Err Then
  MsgBox "An error occurred"
End If


Sub insertIntoCell( strCellName, strText, objTable)
    Set objCellText= objTable.getCellByName( strCellName)
    Set objCellCursor= objCellText.createTextCursor
    objCellCursor.setPropertyValue "CharColor",16777215
    objCellText.insertString objCellCursor, strText, false
End Sub

Function createStruct( strTypeName)
    Set classSize= objCoreReflection.forName( strTypeName)
    Dim aStruct
    classSize.createObject aStruct
    Set createStruct= aStruct
End Function
```

3.2Creating a text document using Object Rexx

This script is the same as above, translated to Object Rexx, using the .OLEObject class. You can find some examples for the usage of the .OLEObject at [Inform05].

As we cannot rely on the BSF/BSF4Rexx objects, we cannot use the Java reflection mechanisms, or Java data/object types.

```
-- [30][OLE][Rexx]
-- The service manager is always the starting point
-- If there is no office running then an office is started up
objServiceManager = .OLEObject~new("com.sun.star.ServiceManager")

-- Create the Desktop
objDesktop= objServiceManager~createInstance("com.sun.star.frame.Desktop")

-- Open a new empty writer document
args = .array~new(0)
objDocument= objDesktop~loadComponentFromURL("private:factory/swriter", "_blank", 0,
args)

-- Create a text object
objText= objDocument~getText()

-- Create a cursor object
objCursor= objText~createTextCursor()

-- Inserting some Text
objText~insertString(objCursor, "The first line in the newly created text document."|
|"0A"X, .false)

-- Inserting a second line
objText~insertString(objCursor, "Now we're in the second line", .false)

-- Create instance of a text table with 4 columns and 4 rows
```

```
objTable= objDocument~createInstance("com.sun.star.text.TextTable")
objTable~initialize(4, 4)

-- Insert the table
objText~insertTextContent(objCursor, objTable, .false)

-- Get first row
objRows = objTable~getRows()
objRow = objRows~getByIndex(0)

-- Set the table background color
objTable~setPropertyValue("BackTransparent", false)
objTable~setPropertyValue("BackColor", 13421823)

-- Set a different background color for the first row
objRow~setPropertyValue("BackTransparent", .false)
objRow~setPropertyValue("BackColor", 6710932)

-- Fill the first table row
call insertIntoCell "A1", "FirstColumn", objTable
call insertIntoCell "B1","SecondColumn", objTable
call insertIntoCell "C1","ThirdColumn", objTable
call insertIntoCell "D1", "SUM", objTable

objTable~getCellByName("A2")~setValue(22.5)
objTable~getCellByName("B2")~setValue(5615.3)
objTable~getCellByName("C2")~setValue(-2315.7)
objTable~getCellByName("D2")~setFormula("sum <A2:C2>")

objTable~getCellByName("A3")~setValue(21.5)
objTable~getCellByName("B3")~setValue(615.3)
objTable~getCellByName("C3")~setValue(-315.7)
objTable~getCellByName("D3")~setFormula("sum <A3:C3>")

objTable~getCellByName("A4")~setValue(121.5)
objTable~getCellByName("B4")~setValue(-615.3)
objTable~getCellByName("C4")~setValue(415.7)
objTable~getCellByName("D4")~setFormula("sum <A4:C4>")

-- Change the CharColor and add a Shadow
objCursor~setPropertyValue("CharColor", 255)
--objCursor~setPropertyValue("CharShadowed", true)

-- Create a paragraph break
PARAGRAPH_BREAK = 0
objText~insertControlCharacter(objCursor, PARAGRAPH_BREAK, .false)

-- Inserting colored Text.
objText~insertString(objCursor, "This is a colored Text - blue with shadow" ||"0A"X, .
false)

-- Create a paragraph break (ControlCharacter::PARAGRAPH_BREAK).
objText~insertControlCharacter(objCursor, PARAGRAPH_BREAK, .false)

-- Create a TextFrame.
objTextFrame = objDocument~createInstance("com.sun.star.text.TextFrame")

-- Create a Size struct.
objTextFrame~FrameHeightAbsolute = 400
objTextFrame~FrameWidthAbsolute = 15000

-- TextContentAnchorType.AS_CHARACTER = 1
AS_CHARACTER = 1
objTextFrame~setPropertyValue("AnchorType", AS_CHARACTER)

-- insert the frame
objText~insertTextContent(objCursor, objTextFrame, .false)

-- Get the text object of the frame
objFrameText = objTextFrame~getText

-- Create a cursor object
objFrameTextCursor = objFrameText~createTextCursor

-- Inserting some Text
objFrameText~insertString(objFrameTextCursor, "The first line in the newly created text
frame.", .false)
objFrameText~insertString(objFrameTextCursor, "0A"X || "With this second line the height
of the frame raises.", .false)

-- Create a paragraph break
objFrameText~insertControlCharacter(objCursor, PARAGRAPH_BREAK , .false)

-- Change the CharColor and add a Shadow
```

```
objCursor~setPropertyValue("CharColor", 65536)
objCursor~setPropertyValue("CharShadowed", .false)


-- Insert another string
objText~insertString(objCursor, " That' s all for now !!", .false)

EXIT

insertIntoCell: PROCEDURE
       use arg strCellName, strText, objTable
       objCellText = objTable~getCellByName(strCellName)
       objCellCursor = objCellText~createTextCursor()
       objCellCursor~setPropertyValue("CharColor", 16777215)
       objCellText~insertString(objCellCursor, strText, .false)
       RETURN


:: requires "OREXXOLE.CLS"
```

3.3Creating a text document using PHP

This script is the same as above, translated to PHP.

If you have PHP installed, you can call this script from the command line (DOS box) like this:

```
C:\> [PHP path]php [script path]Example31.php
```

Or, if you have a web server on your system as well as PHP, start it, put the script in a web documents folder and open it in your browser using the URL:

```
http://localhost/[Path on webserver]Example32.php
```

(Actually, if you do not open it in the web browser, you will have to ignore the HTML tags in the console output. However this is no problem since console or web page output is not essential in this example.)

Should you like to prefer a web server plus PHP with little effort under Microsoft Windows, consider the XAMPP package by the Apache Friends (see Chapter 7: Download Links).

```
<?

// [31][OLE][PHP]
// Testing COM/OLE

// Adapted from the Open Office SDK writerdemo.vbs example

echo "<h1>Testing COM/OLE</h1>\n";

// The service manager is always the starting point
// If there is no office running then an office is started up
$objServiceManager = new COM("com.sun.star.ServiceManager");

// Create the CoreReflection service that is later used to create structs
$objCoreReflection = $objServiceManager->createInstance
("com.sun.star.reflection.CoreReflection");

// Create the Desktop
$objDesktop= $objServiceManager->createInstance("com.sun.star.frame.Desktop");

// Open a new empty writer document
$args = array();
$objDocument= $objDesktop->loadComponentFromURL("private:factory/swriter", "_blank", 0,
$args);

// Create a text object
$objText= $objDocument->getText();
```

```
// Create a cursor object
$objCursor= $objText->createTextCursor();

// Inserting some Text
$objText->insertString($objCursor, "The first line in the newly created text
document.\n", false);

// Inserting a second line
$objText->insertString($objCursor, "Now we're in the second line", false);

// Create instance of a text table with 4 columns and 4 rows
$objTable= $objDocument->createInstance("com.sun.star.text.TextTable");
$objTable->initialize(4, 4);

// Insert the table
$objText->insertTextContent($objCursor, $objTable, false);

// Get first row
$objRows= $objTable->getRows();
$objRow= $objRows->getByIndex(0);

// Set the table background color
$objTable->setPropertyValue("BackTransparent", false);
$objTable->setPropertyValue("BackColor", 13421823);

// Set a different background color for the first row
$objRow->setPropertyValue("BackTransparent", false);
$objRow->setPropertyValue("BackColor", 6710932);

// Fill the first table row
insertIntoCell("A1", "FirstColumn", $objTable);
insertIntoCell("B1", "SecondColumn", $objTable);
insertIntoCell("C1", "ThirdColumn", $objTable);
insertIntoCell("D1", "SUM", $objTable);

$objCell = $objTable->getCellByName("A2");
$objCell->setValue(22.5);
$objCell = $objTable->getCellByName("B2");
$objCell->setValue(5615.3);
$objCell = $objTable->getCellByName("C2");
$objCell->setValue(-2315.7);
$objCell = $objTable->getCellByName("D2");
$objCell->setFormula("sum <A2:C2>");

$objCell = $objTable->getCellByName("A3");
$objCell->setValue(21.5);
$objCell = $objTable->getCellByName("B3");
$objCell->setValue(615.3);
$objCell = $objTable->getCellByName("C3");
$objCell->setValue(-315.7);
$objCell = $objTable->getCellByName("D3");
$objCell->setFormula("sum <A3:C3>");

$objCell = $objTable->getCellByName("A4");
$objCell->setValue(121.5);
$objCell = $objTable->getCellByName("B4");
$objCell->setValue(-615.3);
$objCell = $objTable->getCellByName("C4");
$objCell->setValue(415.7);
$objCell = $objTable->getCellByName("D4");
$objCell->setFormula("sum <A4:C4>");

// Create a paragraph break
// The second argument is a com::sun::star::text::ControlCharacter::PARAGRAPH_BREAK
constant
$objText->insertControlCharacter($objCursor, 0 , false);

// Inserting colored Text.
$objText->insertString($objCursor, " This is a colored Text - blue with shadow\n",
false);

// Create a paragraph break ( ControlCharacter::PARAGRAPH_BREAK).
$objText->insertControlCharacter($objCursor, 0, false);

// Create a TextFrame.
$objTextFrame= $objDocument->createInstance("com.sun.star.text.TextFrame");

// Set size
$objTextFrame->FrameHeightAbsolute = 400;
$objTextFrame->FrameWidthAbsolute = 15000;

// TextContentAnchorType.AS_CHARACTER = 1
$objTextFrame->setPropertyValue("AnchorType", 1);
```

```
// insert the frame
$objText->insertTextContent($objCursor, $objTextFrame, false);

// Get the text object of the frame
$objFrameText= $objTextFrame->getText();

// Create a cursor object
$objFrameTextCursor= $objFrameText->createTextCursor();

// Inserting some Text
$objFrameText->insertString($objFrameTextCursor, "The first line in the newly created
text frame.", false);
$objFrameText->insertString($objFrameTextCursor, "\nWith this second line the height of
the frame raises.", false);

// Create a paragraph break
// The second argument is a com::sun::star::text::ControlCharacter::PARAGRAPH_BREAK
constant
$objFrameText->insertControlCharacter($objCursor, 0 , false);

// Change the CharColor and add a Shadow
$objCursor->setPropertyValue("CharColor", 65536);
$objCursor->setPropertyValue("CharShadowed", false);

// Insert another string
$objText->insertString($objCursor, " That' s all for now !!", false);

function insertIntoCell($strCellName, $strText, $objTable)
{
    $objCellText= $objTable->getCellByName($strCellName);
    $objCellCursor= $objCellText->createTextCursor();
    $objCellCursor->setPropertyValue("CharColor", 16777215);
    $objCellText->insertString($objCellCursor, $strText, false);
}

?>
```

4Using Java LiveDoc via BSF

In [Hahnek05], Rainer Hahnekamp introduces his Java LiveDoc component which generates javadoc-style XML/HTML documentation using Java's reflection classes.

Of course this feature can be used via BSF, too. Example 32 generates a Java LiveDoc page from Rhino and invokes the Web Browser in order to display it. Do not forget to modify the configuration settings at the beginning.

```
// [32][BSF][Rhino]
// Using the Java LiveDoc from Rhino

// Requires
//      (1) Rainer Hahnekamp's ljd.jar (see [Hahnek05])
//      (2) The Saxon XML processing tools: saxon.jar, from
//          http://saxon.sourceforge.net
// to be available for BSF
// For details, see [Hahnek05] (Rainer Hahnekamp's course paper)
// (This example does not require further components of OOBabylon.)

// IMPORTANT: Some configuration is required: =================

// The class to analyze
className = "java.lang.Math";
// How to call your browser
browserPath = "C:\\Programme\\Mozilla Firefox\\firefox.exe";
// Where to put generated files
outputDir = "C:\\";

// End of configuration =======================================

try
{
        // Instantiate Java LiveDoc object
        ljd = new Packages.org.ljd.LJD(className);
        // Create XML output file (contains the class description)
        fWriter = new java.io.FileWriter(outputDir + className + ".xml");
        // Write XML output
        fWriter.write(ljd.outputter.getOutput());
        fWriter.flush();
        fWriter.close();
}
catch(ex)
{
        java.lang.System.err.println(ex);
}

try
{
        // Instantiate an XML transformer factory
        factory = new Packages.com.icl.saxon.TransformerFactoryImpl();

        // Read in the XML file containing the class description from above
        sourceFile = new java.io.File(outputDir + className + ".xml");
        eis = new Packages.com.icl.saxon.ExtendedInputSource(sourceFile);
        sourceInput = new
                Packages.javax.xml.transform.sax.SAXSource(
                factory.getSourceParser(), eis);

        // Read in the XSL stylesheet
        sheetFile = new java.io.File("transform.xsl");
        eis = new Packages.com.icl.saxon.ExtendedInputSource(sheetFile);
        styleSource = new
                Packages.javax.xml.transform.sax.SAXSource(
                factory.getSourceParser(), eis);
        sheet = factory.newTemplates(styleSource);

        // Apply stylesheet and write output to HTML file
        outputFile = new java.io.File(outputDir + className + ".html");
        params = new Packages.com.icl.saxon.ParameterSet();
        styleSheet = new Packages.com.icl.saxon.StyleSheet();
        styleSheet.processFile(sourceInput, sheet, outputFile, params);

        // Finally, open up a browser window and display the HTML file
        runtime = java.lang.Runtime.getRuntime();
        runtime.exec(browserPath + " " + outputFile.getAbsolutePath());
}
```

```
catch(ex)
{
        java.lang.System.err.println(ex);
}
```

```
-- [33][BSF][Rexx]
-- Using the Java LiveDoc from Rexx

-- Requires
--      (1) Rainer Hahnekamp's ljd.jar (see [Hahnek05])
--      (2) The Saxon XML processing tools: saxon.jar, from http://saxon.sourceforge.net
-- to be available for BSF
-- For details, see [Hahnek05] (Rainer Hahnekamp's course paper)
-- (This example does not require further components of OOBabylon.)

-- IMPORTANT: Some configuration is required: =================

-- The class to analyze
className = "java.lang.Math"
-- How to call your browser
browserPath = "C:\Programme\Mozilla Firefox\firefox.exe"
-- Where to put generated files
outputDir = "C:\"

-- End of configuration ======================================

-- Instantiate Java LiveDoc object
ljd = .bsf~new("org.ljd.LJD", className)
-- Create XML output file (contains the class description)
fWriter = .bsf~new("java.io.FileWriter", outputDir || className || ".xml")
-- Write XML output
outputter = ljd~bsf.getFieldValue("outputter")
output = outputter~getOutput
fWriter~bsf.invokeStrict("write", "ST", output)
--fWriter~write(output)
fWriter~flush()
fWriter~close()

-- Instantiate an XML transformer factory
factory = .bsf~new("com.icl.saxon.TransformerFactoryImpl")

-- Read in the XML file containing the class description from above
sourceFile = .bsf~new("java.io.File", outputDir || className || ".xml")
eis = .bsf~new("com.icl.saxon.ExtendedInputSource", sourceFile)
sourceInput = .bsf~new("javax.xml.transform.sax.SAXSource", factory~getSourceParser,
eis)

-- Read in the XSL stylesheet
sheetFile = .bsf~new("java.io.File", "transform.xsl")
eis = .bsf~new("com.icl.saxon.ExtendedInputSource", sheetFile)
styleSource = .bsf~new("javax.xml.transform.sax.SAXSource", factory~getSourceParser,
eis)
sheet = factory~newTemplates(styleSource)

-- Apply stylesheet and write output to HTML file
outputFile = .bsf~new("java.io.File", outputDir || className || ".html")
params = .bsf~new("com.icl.saxon.ParameterSet")
styleSheet = .bsf~new("com.icl.saxon.StyleSheet")
styleSheet~processFile(sourceInput, sheet, outputFile, params)

-- Finally, open up a browser window and display the HTML file
runtimeName = .bsf4rexx~Class.class~forName("java.lang.Runtime")
runtime = bsf("invoke", runtimeName, getRuntime)
abspath = outputFile~getAbsolutePath
bsf("invokeStrict", runtime, "exec", "ST", browserPath || " " || abspath)


::requires "BSF.cls"
```

5Conclusion

Open Office scripting is an important issue since there is (a) a growing need for it and (b) no proprietary standard settled yet, which makes way for the variety of open-source technology.

If you work with BSF, you can choose virtually every scripting language to code. All the (originally Java) examples chosen from the Developer's Guide could be translated to Rhino and even Object Rexx (with underlying BSF4Rexx).

If you however choose to make use of the OLE scripting interface, you could well encounter insuperable hurdles, too, since you would depend on (a) Microsoft Windows and (b) the scripting language's COM implementation.

Plus, Java UNO programming is no task for beginners or hobby macro programmers. Its complexity cannot be easily mastered by the typical, occasional VB script programmer. A simplified "Meta API", as used in the DHTML object model, would be helpful. Imagine you could write

```
spreadsheet.insertIntoCell(0, 0, 123.45);
spreadsheet.colorizeCell(0, 0, "blue");
```

instead of connecting to Open Office, retrieving the objects with respect to the object hierarchy in their appropriate contexts, doing type conversions, handling address objects and ranges, working with style family suppliers etc.

> An alternative to coding a Java-side "Meta API" is to write function libraries for the specific scripting languages that abstract Java UNO's complexity.
>
> Another alternative is to write an abstraction library in one scripting language and make that available via BSF so there is no need to translate it to other scripting languages.

In a nutshell, while the experienced programmer is able to profit a lot from the possibilities that come with Open Office and BSF, less skilled users, or users with lesser time, should wait for eventual simplification efforts from the Open Source developers' community.

6List of references

[Flats01] Prof. Dr. Rony Flatscher: “bsf4rexx" - "Bean Scripting Framework" (BSF) for Rexx, at: http://nestroy.wi-inf.uni-essen.de:8080/Forschung/rgf/Entwicklung.html; 2000; retrieved on January 7, 2005

[Flat05] Prof. Dr. Rony Flatscher: Java Automation - Course slides (in German), at: http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/folien; 2004; retrieved on January 7, 2005

[Hahnek05] Rainer Hahnekamp: Extending the scripting abilities of OpenOffice.org with BSF and JSR-223; course paper, Vienna University of Economics and Business Administration, Information Systems and Operations (Prof. Dr. Rony Flatscher); January, 2005

[IBM05] IBM: IBM Rexx Family Homepage, at: http://www-306.ibm.com/software/awdtools/rexx/; retrieved on January 7, 2005

[Inform05] Informit Network Homepage - Windows Scripting and Objects – Using Objects with Object REXX, at: http://www.informit.com/articles/article.asp?p=29570&seqNum=7; retrieved on January 7, 2005

[PHP05] Zend Company: PHP Manual - X. COM and .Net (Windows), at: http://www.php.net/com; retrieved on January 7, 2005

[RexxLA05] The Rexx Language Association Homepage, at: http://www.rexxla.org; retrieved on January 7, 2005

7Download Links

BSF – Apache Foundation: Bean Scripting Framework (most recent version)
http://jakarta.apache.org/site/binindex.cgi#bsf

BSF4Rexx – Prof. Dr. Rony Flatscher: Bean Scripting Framework for Rexx
(official download site:) http://wi.wu-wien.ac.at/rgf/rexx/
(Beta of "Viennese version" used in this paper:)
http://wwwi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/bsf.upd/

Eclipse – Eclipse.org: Eclipse Project Downloads
http://www.eclipse.org/downloads/index.php

J2SE (Java 2 Standard Edition) – Sun: J2SE 1.4.2 download page
http://java.sun.com/j2se/1.4.2/download.html

NetBeans – NetBeans.org: Downloads
http://www.netbeans.org/downloads/

Open Office – OpenOffice.org – Download Central
(including .jar files:) http://download.openoffice.org

Open Office SDK – OpenOffice.org – SDK download page
(including Developer's Guide:)
http://www.openoffice.org/dev_docs/source/sdk/index.html

Rhino – Mozilla.org: JavaScript for Java
http://www.mozilla.org/rhino/download.html

XAMPP – The Apache Group: Integrated open-source software web server package (Apache, PHP, MySQL etc.)
http://www.apachefriends.org

Zend Studio – Zend: Zend Studio 3.5 Beta 2 (demo version download page)
http://www.zend.com/store/products/zend-studio-beta.php

Example scripts from this document:
http://www.matto.at/oo_examples/

8Index