Extending The Scripting Abilities Of OpenOffice.org With BSF And JSR-223

Rainer Hahnekamp
Matrikel#: 0250658
Vienna University of Economics and Business Administration
Rainer.Hahnekamp@aon.at

Prof. Dr. Rony G. Flatscher
Department for Business Administration and IS
Rony.Flatscher@wu-wien.ac.at

WIRTSCHAFTS
UNIVERSITÄT
WIEN

Contents

Figures

1 Introduction

1.1 Research Question

The Research Question was defined as follows:

How can the scripting abilities of OpenOffice.org with BSF and JSR-233 be extended that macro developers can use of technology with minimalistic effort?

By analysing the question it can be divided up into 4 main tasks:

- use the Bean Scripting Framework to write macros in different scripting languages
- do the same with JSR-223 as underlying technology
- develop a concept that supports macro developers using BSF/JSR-223 in different ways
- implement that concept

1.2 Audience

As already mentioned above before the main audience of this article are macro developers of OpenOffice.org.

But also other interested programmers or just everybody who is interested in BSF/JSR-223 in general or just wants to know how to connect scripting languages with Java is invited to read this document.

If you have further questions that are not answered in this document or criticism, suggestions or feedback in any kind of way I would be happy if you contact me at Rainer-.Hahnekamp@aon.at.

1.3 Requirements

You should have experience in programming in general especially in Java, scripting languages and web development. Further more you have to be familiar with technologies like XML, XSLT, DOM and how are they used. They are not explained in detail since the document's intention is to focus on OOBabylon and its functionality.

1.4 How this document is organized

This document consists of 3 main parts.

After the introduction comprising this chapter and the next chapter namely the Executive Summary which gives a quick answer to the research question respectively describes the result OOBabylon the first part starts with the theoretical aspects. You will get basic knowledge about scripting languages in general, OpenOffice.org, BSF and JSR223. If you already familiar with them you can just skip the chapter. Otherwise I strongly recommend that you read it because otherwise you will not understand the fundamentals of OOBabylon. If you have still questions after reading that chapter you can take a closer look at the link list at the end of the document.

The second part describes the course of action during the semester where OOBabylon where developed in concept and also implemented in its first version.

Finally the third part deals with the OOBabylon application and the "Live JavaDoc" which are both the result of our researches.

1.5 Acknowledgements

I want to thank Prof. Flatscher for his support during the whole semester and his insider information he told us especially at the „Skiseminar". Eventually Walter and I should also be thankful that he stopped us from developing a Meta-API which would have not led to a satisfying result.
Many thanks goes also to Walter who accompanied me the whole semester and has been a very helpful and competent colleague.

1.6 Finally

I hope that OOBabylon has the potential to become a worthy addition to OpenOffice.org since it opens the API for scripting languages. As a result it attracts more developers since they have the possibility to program with their favourite language.

2 Executive Summary

In the last years the Open-Source project GNU/Linux with its applications got more and more attention from companies that tried to complement or even substitute it with the predominant commercial software system. These applications were mainly – but not all – developed by Microsoft. MS Office can be count to one the top products where Microsoft has a lock-in. OpenOffice.org is a new challenger on the market which is on the one hand an Open-Source project and on the other hand has the quality to challenge MS Office.
A superior support for macro development could be a competitive advantage for an office application. Concerning this aspect one can see that in both programs macro development is always bound to one and only programming language.
This document describes the development and functionality of OOBabylon which allows programmers to develop applications in different scripting languages and execute them in a Java environment. Its main usage is the development of macros for OpenOffice.org in different scripting languages.

OOBabylon uses the Bean Scripting Framework from the Apache project and JSR-223 as fundamental technologies. It is designed to hide the whole complexity of BSF/JSR-223 from the user and offers some very helpful tools to simplify the development process. In addition it has some nutshell examples, especially for macro developers programming in OpenOffice.org.

The great advantage of OOBabylon is that you have a bundle with pre-configured scripting languages in BSF like Jython (=Python) or Rhino (=Javascript) that allows you a much faster development process and means therefore less costs.

The other advantage is that you can write your macros in every scripting language which is available for BSF. If a macro developer therefore knows how to program in Javascript until now he was not able to write OpenOffice.org scripts. With OOBabylon and BSF he has no the ability to chose from a pool of many scripting languages to write his macros.

3 Basics

This chapter is the theoretical part of my work. I am going to explain the four main topics you should be familiar with if you want to use OOBabylon efficiently.

3.1 Scripting Languages

In this chapter we are going to deal with scripting languages. We start with the definitions go on with the characteristics and its areas of use. At the end we will get to know some popular script languages which forms an integral part in OOBabylon.

3.1.1 Introduction

Some years ago it was very easy to classify programming languages. Usually languages were classified according to their age and therefore one spoke of different generations of programming languages. Some languages which where in the beginning conceived for a special purpose got extended with lots of functionalities to use them in other areas. So these got a little bit mixed up and it was getting more and more difficult to assign them to a specific generation. So one decided not to classify them into generations but according to their usage. Therefore today languages are classified in categories like procedural, logistic, functional, object-oriented or scripting languages. [HaNe01]
What are they now really? In contrast to „big“ languages like C++ or Java they were in the beginning developed to automatize regular tasks. So the syntax was more oriented to code fast. In general scripting languages have following characteristics:

- Extensibility: It is generally very easy to write extensions for scripting languages.
- 2 Layers: Usually the different commands of a scripting language are implemented in another programming language like C and can be therefore executed very fast.
- Interpretative and dynamical typing: This means that a variable can have different types. In one command row it can be a String an in another it is used for calculation as an Integer. Please notice that at one moment the variable can have only one type.[HaNe01]

3.1.2 Areas

As in the beginning already mentioned languages and especially scripting languages evolved in the course of the years and got more and more functionalities so that they could be used in different areas. Please do not take these statement as a strict one. Some languages can be used in more than one area (like Perl):

- Command Line Interpreters: These languages can be used within a console of an operating system. Popular ones are Tclsh, bash respectively cs (classical Unix shell) or in the MS Windows world the older command.com or its newer version the cmd.exe.
- used in different programs: Here the language is embedded in an application where you can program macros (like OpenOffice.org) for instance. Examples are AppleScript for the Mac OS or VBA for the MS Office suite or a similar basic dialect for OpenOffice.org.
- World Wide Web: As applications based on a web interface are getting more and more it is clear that there must also be some languages for this area as well on the client-side as on the server-side. Example are PHP, Javascript or VBScript.
- Powerful languages: Such scripting languages are suited for developing even large-scale applications, like it is the case with Python and Gnome. Usually they are object-oriented. This is also the reason why this group is mainly supported from BSF like Python, Perl, Rexx or Ruby.[Wiki05a]

3.1.3 Examples

Jacl

Jacl is the reimplementation of Tcl for Java. Tcl is the acronym for „Tool Command Language“ and its main usage is in areas of GUI (Tcl/Tk), unit testing or rapid prototyping.

There also exists an extended version named XOTcl which has been developing by Prof. Neumann and Dr. Zdun who teach on our university at the department of Information Systems – New Media Lab. XOTcl has features like the concept of a Meta-Class or the dynamic change of a class during runtime.[NeZd05]

Jython

Jython is like Jacl also a reimplementation of a scripting languages in Java. In this case it is

Python which is an extreme popular language. Formerly known as JPython, Jython has the ability to use the Java API. Therefore GUIs in Jython are programmed with the swing or awt classes instead of Tkinter.

Python was created in 1990 by Guido van Rossum. Its intention was designed to program in different ways. You have the opportunities to program for example object-oriented, procedural-oriented or even aspect-oriented. Popular programs are the application server Zope or popular BitTorrent system. Python is an Open-Source Software and therefore available freely at www.python.org in its actual version 2.4.[Wiki05b]

Rhino

Many people do not really know what Rhino stands for but they all know its descent language namely ECMAScript or also known as Javascript. ECMAScript is the real name. The only reason it is known as Javascript is that when it was launched everybody spoke of Java. As the other two scripting languages mentioned above Rhino is also a reimplementation in Java. It is Open-Source Software and developed under the Mozilla project. Javascript has not very much in common with Java except the similarity of the name.

Javascript is mainly used for client scripting in HTML pages. Since HTML is very easy to learn many computer users know it. As a consequence their first contact with a real programming language is the bridge Javascript over HTML. Because of its popularity and spreading most of the nutshells shipped with OOBabylon are written in Rhino (and OORexx).

Perl

Perl stands for „Practical Extraction and Report Language“ and was written by Larry Wall in 1987. It is one of the most popular used scripting languages and is developed under OSS (=Open-Soure Software). It has been using for CGI (=Common Gateway Interface). Since other technologies like Servlets or Application Servers are pushing more and more into the market and CGI's drawbacks like functionality or scalability its usage is decreasing. Nevertheless Perl can be found in many other areas of applications.

Perl's best known features are the regular expressions which can be traced back to its original usage.[Wall00]

Perl is not as the first three mentioned scripting languages reimplemented in Java. Therefore it is not that easy to use it with BSF. Nevertheless there exists a version for BSF from the company ActiveState but it is windows only

Therefore I decided to use an alternative namely BSFPerl at http://sourceforge.net/projects/bsfperl in its actual version 0.2 which is platform-independent to integrate into OOBabylon. BSFPerl can be seen as a bridge between BSF and Perl since it needs a Perl interpreter.

OORexx

Open Object Rexx is the newly „open sourced" version of Object Rexx. Rexx can be traced back to the year 1979 where it was developed by Mike Cowlishaw at IBM to be a scripting programming language. This means that it was from the beginning designed to be used in wide areas of programming. It is said that it is the precursor for such popular languages like Tcl or Python.[Wiki04]

There are many different variants available. The newer ones are NetRexx and ObjectRexx. ObjectRexx – as the name already says – allows the developer to program his applications using the object-oriented paradigm.
Last year IBM put ObjectRexx under its Open-Source licence CPL (=Common Public License). So ORexx is now OORexx which stands for Open Object Rexx and can be downloaded from the site of the Rexx Language Associaton at http://www.rexxla.org/oorexx/.

Professor Flatscher is very deeply involved since he was in the negotiation team during the license-phase and is a member of the development team. Further more he is the developer of BSF4Rexx [Flat03] [Flat04a] which has been improved a lot since its first release in 2001 [Kale01]. For more information please take a look at the link list at the end of this document.

It is by far more complexer to develop a BSF-Engine for a programming language which has not a reimplementation in Java and must therefore be used in another way. Usually you write a dll and access it via JNI. As an example here is the architecture how ORexx is linked with BSF.

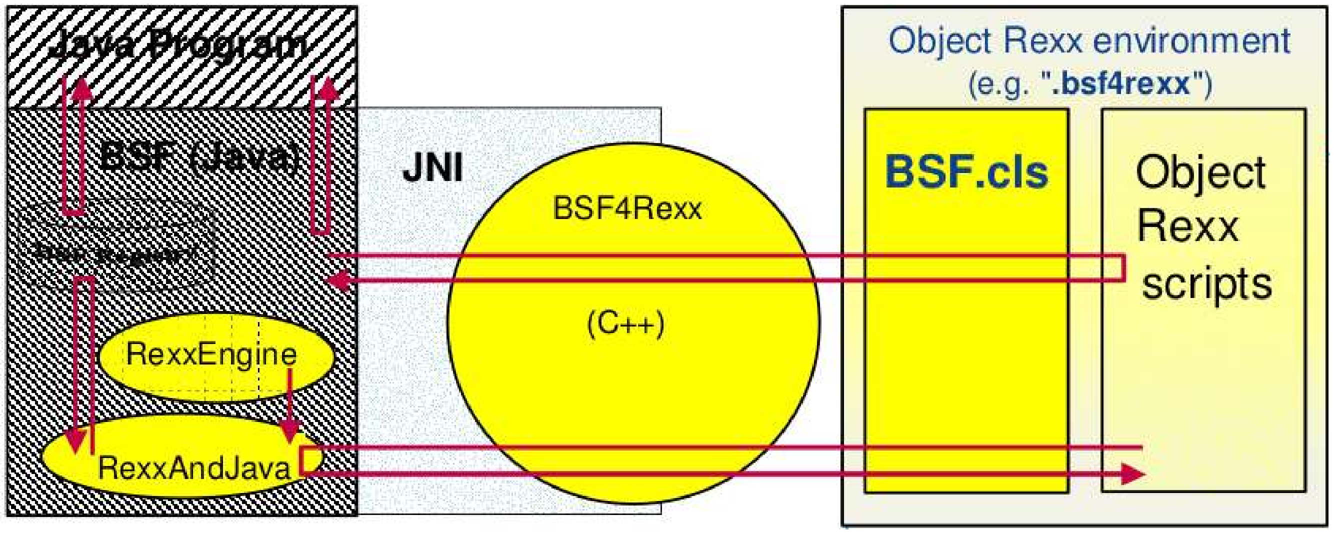


Figure 1 BSF4Rexx, [Flat04b]

3.2 OpenOffice.org

As most readers certainly are familiar with OpenOffice.org I will deal with the topic only shortly and set the main focus on UNO.

3.2.1 Introduction

OpenOffice.org is an office suite comparable to MS Office. In contrast to the latter is Open-Source and therefore freely available for Windows, Mac OS X and Linux systems. OpenOffice.org comprises following core applications:

- Writer: word processor
- Calc: spreadsheet
- Draw: graphics program

It has lots of other useful tools like a mathematical formula editor or a PDF-export function. Further more OpenOffice.org is known for its excellent XML-Format. OpenOffice.org files are stored in XML format and can be therefore be easily transformed into another format. Nevertheless OpenOffice.org was often criticised because of its very slow behaviour.

At present OpenOffice.org is available in version 1.1.4 and it is planned to release OpenOffice.org 2.0 in march or April 2005.

OpenOffice.org has been licensed under LGPL by Sun Microsystems.[Wiki05c]

3.2.2 History

OpenOffice.org's predecessor Star Office was first released by the company StarDivison which was founded in 1984 by the 16-year old Marc Börris. Until 1999 StarDivision has sold 25,000,000 units. In the same year Sun purchased the whole company and offered StarOffice in its version 5.2 for free. 1 year later OpenOffice.org was pronounced.[Wiki05c]

3.2.3 API

As API OpenOffice.org uses the powerful „Unified Network Object“ model. According to the UNO Development Kit Project it is defined as:

> *„UNO offers interoperability between different programming languages, different object models, different machine architectures, and different processes; either in a local network or even via the Internet. UNO components can be implemented in and accessed from any programming language for which a UNO language binding exists.“[Udkp05]*

Interoperability means that the model and its access is not bound to a special language, but was designed in such a way that nearly every programming language can use its classes, methods and so on. As you can already imagine every programming language needs some access or some kind of a bridge which is called binding. In our case we use the Java bindings to access it via BSF:

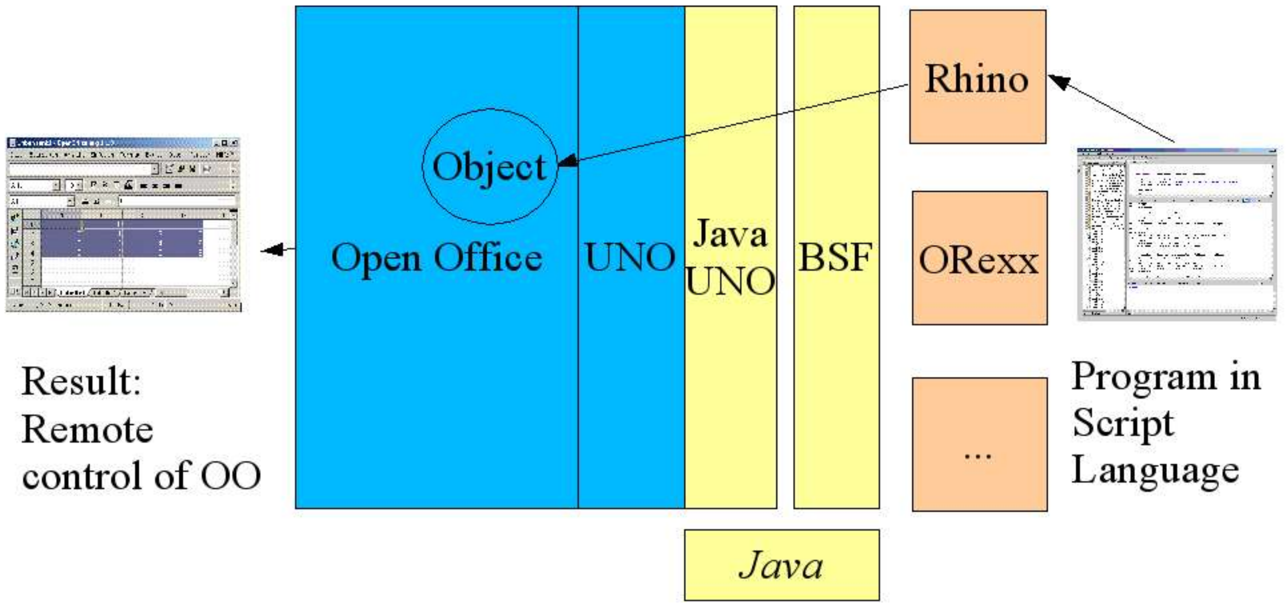


Figure 2 Java UNO, Walter Augustin

In Windows you have also the possibility to access UNO via OLE. If you are interested in this methodology please take a closer look at the document of my colleague Walter Augustin. [Augu05]

As you can already imagine UNO is a client/server system. This means that you have at first to initiate a client application that connects via TCP/IP to a host with a specific host where OpenOffice.org's UNO is listening. For security reasons you have to enable the server manually by editing the file Setup.xcu in your OpenOffice.org directory. For more information see the first steps in the OpenOffice.org Developers Guide at http://api.openoffice.org/docs/DevelopersGuide/FirstSteps/FirstSteps.htm [Open03]

Since the code which opens the connection to the server is always the same, there exists a pre-registered object in BSF that returns an opened connection in form of the XComponentLoader class under the name oobComponentLoader.

3.3 Bean Scripting Framework

In this chapter you will get information about the Bean Scripting Framework which is the core component of OOBabylon and can in short be described as the bridge between a scripting language and Java.

3.3.1 Introduction

The name Bean in BSF does not mean that it can only be used in context with Java Beans. The real reason is that at the time it was first developed the term „Bean“ was a hotshot-word in the scene and therefore one decided it was named “Bean Scripting Framework” because marketing reasons.[Bsf02]
According to its developers BSF is defined as:

> *„...a set of Java classes that enables the use of scripting languages such as Javascript and Tcl, within Java applications and that permits the use of Java objects and functions within the supported scripting languages.“ [Orli02]*

This definitions implies 2 main functionalities or forms of use with BSF:

1. Embedding Script code into a Java-based program or environment: This was the main intention of BSF. You can create servlets written in any scripting language that is supported by BSF. Therefore on the one hand you can combine the advantages of scripting languages e.g. the ability to write code in a fast way or use special modules which are only available or superior in this certain language[1] and on the other side you can use the whole functionality offered by a Java environment.
2. Usage of Java components in Scripting Languages: Here you tackle BSF from the scripting language's point of view. Imagine you have for some reason the task to write a platform-independent application in OORexx. Unfortunately you have to use components which are written in Java. Normally there is no way to do that[2]. With BSF you have the ability to have your component registered by the BSF Manager which makes it available for your script code.

BSF is not difficult at all. Following code would produce a „Hello World“ output in Rhino:

```
import org.apache.bsf.*;

public class BSFExample {
      public static void main(String args[]) {
            BSFManager manager = new BSFManager();
```

1 For example image manipulation

2 I know that there are work-arounds but I want to show the principle.

```
            try {
                  manager.loadScriptingEngine(„javascript");
                  manager.eval(„javascript", „test", 0, 0,
„java.lang.System.out.println(\"Hello World\")");
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
      }
}
```

3.3.2 History

BSF began its life in 1999 at IBM as a research project of Sanjiva Weerawarana. After that it was moved to Alpha Works (also IBM) where it was passed on to the developersWorks especially to Matt Duftler and Sam Ruby where it was integrated into IBM's Application Server Websphere and also into Xalan from the Apache project. [Orli02]

The usage in Xalan was the first contact with Apache and it did not last long until BSF was integrated into Apache's subproject Jakarta in fall 2003. As a result today there exists two versions of BSF:

- IBM: java.com.ibm.bsf.Main
- Apache: org.apache.bsf.Main

3.3.3 Architecture

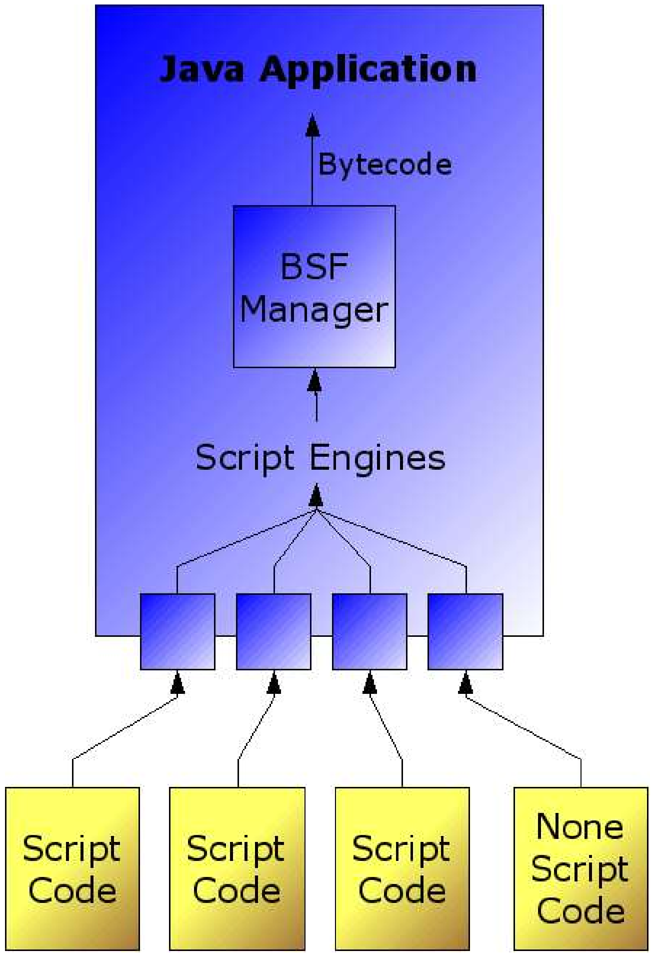


Figure 3 BSF, Rainer Hahnekamp

The image above shows the – as it is meant to be – architecture of an Java application using BSF. You can see that the Script code usually lies outside of the application. It is often a simple text file with source code that's content is passed to the appropriate Script Engine. In order to be a Script Engine the class has to implement a number of special methods – and that's all! The BSFManager class holds all Script Engine Objects and uses them to pass its translated Bytecode to the rest of the Java application.

Please notice that there is also the possibility to use Non-Scripting Code like JLog which is a reimplementation of Prolog in Java or even XSLT.

3.4 JSR-223

JSR stands for „Java Specification Request“ and is a part of the Java Community Program.

The JCP was invented in 1998 to allow the community to take part of future Java technologies, reference implementations or test suites. If you want to become a member I would suggest that you try to get a status of an individual since the other fees are $2,000 for educational, governmental or non-profit organizations and for commercial entities they are up to $5,000.

A JSR is created when some members submit a proposal to the Process Management Office of Sun to develop a new technology. If it is approved it gets the name JSR followed by an identity number. JSR-223 is such a Specification Request. [Sun04a]

A JSR consists usually of an expert group consisting of individuals and companies. In JSR-223 we have companies like Macromedia, Oracle, Zend or IBM and as one of the individual experts Prof. Flatscher.

Shortly JSR-223 has following goals:

- providing a common set of programming interfaces that can be used to execute scripts in scripting engines and bind application objects into the name spaces of the scripts
- allowing Java application developers to execute programs written in scripting languages in their Java applications
- Enabling scripting pages to be executed in any Java Web application or ServletContext in a Web container
- Standardizing how Web pages written in scripting languages are packaged into Java Web applications
- suggesting how Java objects can be exposed in programs written in scripting languages [Sun04b]

Doesn't this sound familiar to you? Yes, the goals are nearly the same as in BSF.

Considering the fact that BSF was initially written to be used within an Application Server one could assume that JSR-223 and BSF are the same.

The first results are already out. There exists a so-called Early Reference Implementation where you can already write servlets in PHP respectively write Java applications in script languages Rhino and Groovy. Unfortunately PHP is only able to run within a ServletContext so it was not integrated in OOBabylon. Rhino and Groovy can be run „freely“[3] and are therefore available in OOBabylon but it looks like that Rhino and Groovy are internally

3 without a ServletContext

running via BSF since the BSF package is included in the ERI. This leads to my personal assumption that in the final version the core technology will be BSF.
Summarized, the main difference between BSF and JSR-223 or the bridge between them is that JSR-223 will make BSF to an official part of upcoming Java versions.

3.5 Summary

After reading this chapter you should know what script languages are, what they can be used for and also some popular examples which are also available in OOBabylon.
Further more you should know that in OpenOffice.org the „Unified Network Object“ model builds the API and is designed to be used in every programming language. Its implementation is a client/server system. In OOBabylon the UNO is accessed via the Java bindings.
BSF is a bridge between Java and scripting languages that has the advantages that the time and therefore the costs are smaller since scripting languages allows you a RAD (=Rapid Application Development) and that you can use the whole functionality of Java environments like J2EE.
Finally JSR-223 is a process that aims to create – as in BSF – a bridge between scripting in Java. The result will be part of upcoming Java versions.

4 Course of Action

This chapter deals with the environment in which OOBabylon was implemented and with the different phases during the semester.

4.1 Environment

I am studying a six-semester bachelor's program in Information Systems at the Vienna University of Economics and Business Administration. So this document is not only a documentation about OOBabylon it is also my Bachelor's thesis that was made during the semester lasting from October 2004 until February 2005.
This course was held by Prof. Flatscher from the Department for Business Administration and IS who assigned the research question explained at the beginning of the document to us and supported us during our work.
My colleague was Walter Augustin who had finally the task to implement nutshells which are little coding examples. [Augu05]

4.2 Conceptual phase

4.2.1 Getting to know BSF

BSF was not a real problem, since I am working on a Debian system which offers already a BSF and rhino package. Therefore it was not really difficult to write a „Hello World“ script in rhino and execute it through the appropriate BSF-Engine.

The theoretical aspect was not a difficulty one since the BSF project page offers enough information.

4.2.2 Getting to know UNO

The access to OpenOffice.org was by far a bigger problem. We hoped that writing scripts in OpenOffice.org would be as easy as it is in MS Office with VBA. But to our surprise we found out, that OpenOffice.org API is a Client/Server system and is based on UNO. Our next step was to get to know UNO and to write an example in Java that opens a Writer Document and prints „Hello World“. We tried to get as much information as possible in the short time we had

(~1 month). Finally we ended up with a functional script and some basic knowledge what UNO is, how it is integrated in OpenOffice.org and what possibilities it offers.

4.2.3 Programming a Proof-of-Concept

Next step was to program a so-called Proof-of-Concept showing that it is possible to write macros with BSF. As example we used our “Hello World” script and reimplemented it in BSF. To our surprise there were no problems and all worked perfectly from the very beginning on.

4.2.4 Dividing up

After having a Proof-of-Concept and enough knowledge about BSF and UNO Walter and I decided to split our work. Walter concentrated more on UNO and its structure to create nutshells in Rhino and OORexx. My task was to go a little bit into BSF, its languages and to develop a concept how to simplify the programming process with BSF and UNO since we already knew that UNO is not an easy model.

4.2.5 Meta-API

After some time Walter came to the conclusion that it would be the best to program some script functions for different tasks like opening a document, saving it, getting the content of cells and so on. As you can imagine this functions should implement some of the most used actions in OpenOffice.org. His idea grew and grew and in the end we decided to develop a so-called Meta-API that would be a API over UNO offering some very powerful functions. Our goal was that a macro developer should use this Meta-API for common tasks and day-to-day programming. For more complicated things he can always access the UNO interface. I for myself doubted a little bit that we were able to implement such a powerful functionality in such a short time of 6 months with all our other studying work in the background.

Eventually it was Professor Flatscher's advice not to stick to a Meta-API since it could become a complex problem and in the case of a reorganisation of UNO it would be useless. On the other side we should concentrate on our main task namely to find out how we can help the developers to get access to this complicated API and facilitate their work. So we decided that Walter should go on with his nutshells and I for myself put me in the position of a macro developer and asked myself what I would wish if I had the task to program scripts for OpenOffice.org.

4.3 First sight of OOBabylon

After some thinking the first shapes of a GUI what would later become OOBabylon respectively Live JavaDoc came into being. OOBabylon should be an application with a pre-configured BSF respectively JSR-223 and a GUI that hides most of the complexity and offers some tools to facilitate programming. One of the main components was Live JavaDoc that uses the reflection mechanism of Java to analyse classes or objects during the runtime.[4]

4.4 Implementation phase

4.4.1 BSF packaging

After the GUI was programmed and fully functional I went on to create a „BSF package“ with a selection including the most popular scripting languages like Jython, Perl, Javascript, Jacl, Ruby or OORexx. Further more I added a specialized version of JLog (=Prolog in Java) from Ulf Dittmer which is in my opinion a very interesting language.

4.4.2 JSR-223 ERI packaging

Next on the To-Do list was the configuration of JSR-223, which is still – as its name says already – in development. Nevertheless the expert group around it is offering an Early Reference Implementation (=ERI) with a running PHP 5 – Release Candidate 2, Groovy and Javascript. Unfortunately PHP was only able to run in a ServletContext so I could not implement it in OOBabylon.

4.4.3 Live JavaDoc

After BSF and JSR-223 was linked with the GUI, OOBabylon was getting more and more functionally. The next task was the integration of Live JavaDoc into OOBabylon, which was developed at the same time. The LJD (=Live JavaDoc) component was used to identify often used classes of the OpenOffice.org API or classes chosen from the user himself.

For more info please read the chapters about OOBabylon and Live JavaDoc

4 It also available as an object in the BSF registry

4.4.4 The whole picture

At the end Walter's nutshells were integrated into the GUI to complete the whole picture of OOBabylon. Finally we had a prototype of a OOBabylon offering on the one side an easy access to the OpenOffice.org API by means of nutshells and other small tools and on the other side was able to execute code in a number of different scripting languages.

5 Live JavaDoc

This chapter deals with Live JavaDoc which analyses Java classes via the Java Reflection namespace and delivers an XML String as result.

5.1 Introduction

Live JavaDoc is one of the two main components that were developed with the purpose to facilitate the work of macro developers to write scripts for OpenOffice.org. LJD (=Live JavaDoc) allows you to analyse a class dynamically and during the runtime.
So if you encounter an object in your code that is not documented anywhere you can pass it to LJD which gives you a detailed output of its methods, fields, interfaces, constructors and superclasses.

5.2 Architecture

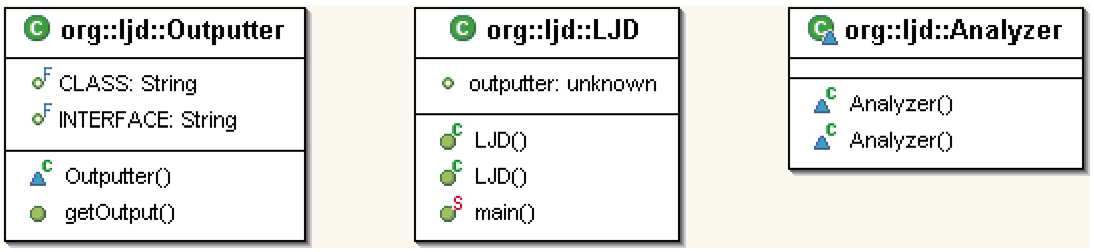


Figure 4 Live JavaDoc, Rainer Hahnekamp

The architecture of LJD is a very simple on. We have three classes. You start the analysing-process by instantiating the LJD class. As parameter you have to pass a java.lang.Class object or a String with the fully name of the class you want to analyse. It is then passed on to the Analyzer class which – as the name says – analyses the class. After the Analyzer is finished it passes his data to the Outputter class which is responsible for returning the XML String. As user you can access the Outputter through your initially instanced LJD object. Please notice that you do not have to trigger the analyse process explicitly. It is automatically started if you create an LJD object.

If you want to use LJD from the command line you just need to execute the LJD main method and will get in return the desired object analysed in an XML format.

5.3 Main Activity

The following UML activity diagram shows how LJD is used in OOBabylon. In the next chapter on OOBabylon this process is discussed in detail.

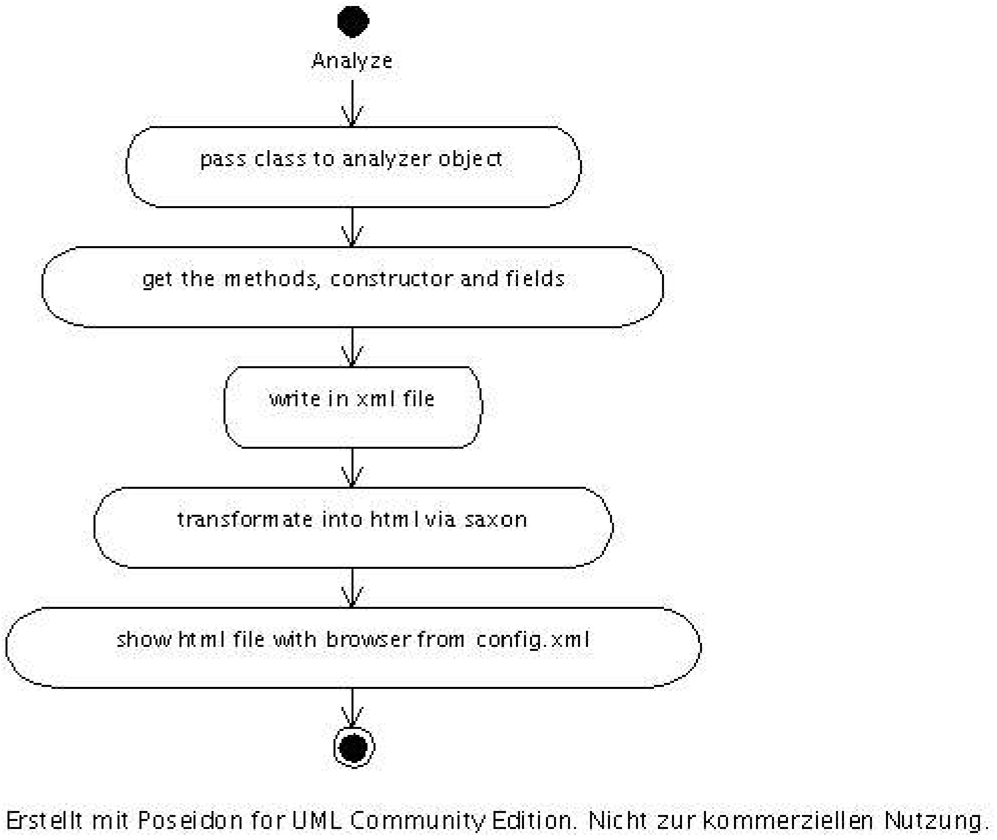

Analyze
pass class to analyzer object
get the methods, constructor and fields
write in xml file
transformate into html via saxon
show html file with browser from config.xml
Erstellt mit Poseidon for UML Community Edition. Nicht zur kommerziellen Nutzung.

Figure 5 Live JavaDoc Activity, Rainer Hahnekamp

5.4 XML output

As already mentioned above the output of LJD is an XML String. I chose XML because of its possibility to transform it easily in another format. You can integrate LJD in your application take the XML String and transform it in PDF or HTML as it is the case in OOBabylon. You could also build a TreeView in your application that shows the different methods.
An interesting usage could be the integration into a Servlet, where you transform the XML into HTML and then create a link to every class which is also used, e.g. as parameter for methods. The user can now click on this class and the Servlet is executed again but now with

the analysis of the clicked class and so on...
In this respect you can build a Servlet which analyses dynamically all classes it has in its Classpath.
For the creation of the XML String I am using the library JDOM which is available at www.jdom.org.

5.5 Tutorial

Let's try LJD for real! You just need the three classes mentioned the right directory structure so that there are no problems with the namespace (org.ljd.*). We are going to examine the java.util.HashSet class via the console mode.
You open your console go to the root directory of LJD and type in following command:

```
java org.ljd.LJD java.util.HashSet > HashSet.xml
```

Now LJD analyses the HashSet class and writes its output into the file HashSet.xml which would look like this:

```
<class name="java.lang.Integer">
      <modifiers />
      <superclasses>
            <superclass>class java.lang.Number</superclass>
            <superclass>class java.lang.Object</superclass>
      </superclasses>
      <interfaces>
            <interface>java.io.Serializable</interface>
            <interface>java.lang.Comparable</interface>
      </interfaces>
      <constructors>
            <constructor>
                  <exceptions>
                        <exception>java.lang.NumberFormatException</exception>
                  </exceptions>
                  <parameters>
                        <parameter>java.lang.String</parameter>
                  </parameters>
            </constructor>
      </constructors>
      <methods>
```

```
            <method>
                  <name>byteValue</name>
                  <return>byte</return>
            </method>
      ......
            <method>
                  <name>valueOf</name>
                  <return>java.lang.Integer</return>
                  <exceptions>
                        <exception>java.lang.NumberFormatException</exception>
                  </exceptions>
                  <parameters>
                        <parameter>java.lang.String</parameter>
                        <parameter>int</parameter>
                  </parameters>
            </method>
            <method>
                  <name>wait</name>
                  <return>void</return>
                  <exceptions>
                        <exception>java.lang.InterruptedException</exception>
                  </exceptions>
                  <parameters>
                        <parameter>long</parameter>
                        <parameter>int</parameter>
                  </parameters>
            </method>
      </methods>
      <fields>
            <field>
                  <name>MAX_VALUE</name>
                  <type>int</type>
            </field>
            <field>
                  <name>MIN_VALUE</name>
                  <type>int</type>
            </field>
            <field>
                  <name>TYPE</name>
                  <type>java.lang.Class</type>
            </field>
      </fields>
</class>
```

As you can see the XML file has 6 main nodes:

- Modifiers: if the class is static, etc.
- Superclasses: the parent classes of the specific class until to java.lang.Object
- Interfaces: these are the interfaces which have to be implemented
- Constructors: all constructors with their arguments
- Methods: the available methods with their arguments
- Fields: the fields if available

In the next version of LJD the XML output will get a DTD.

5.6 Summary

After reading this chapter you should be familiar with the functionality of LJD which can be - if used in the right way – a very powerful tool. Live JavaDoc is able to analyse every Java class and prints out an XML string with the similar to that of the original JavaDoc program.
You should also have an idea of the possibilities you can do if you have such an XML String. In OOBabylon for example it used for creating an HTML file which is automatically shown in a browser window.

6 OOBabylon

In this chapter I am going to show you OOBabylon. We start with the layer model and its different components along their functionality. After that we analyse the three different types OOBabylon can be used for. At the end we take a closer look at OOBabylon's architecture and how it can be configured.

6.1 Introduction

OOBabylon can be seen as the result or answer to the research question. At first glance it may look like a normal text editor but there is more. OOBabylon is divided up into a layer model which consists of a GUI, environmental tools and a core layer.
Besides the three layers it offers also three types of usage namely as GUI, console or library.

6.2 Layers

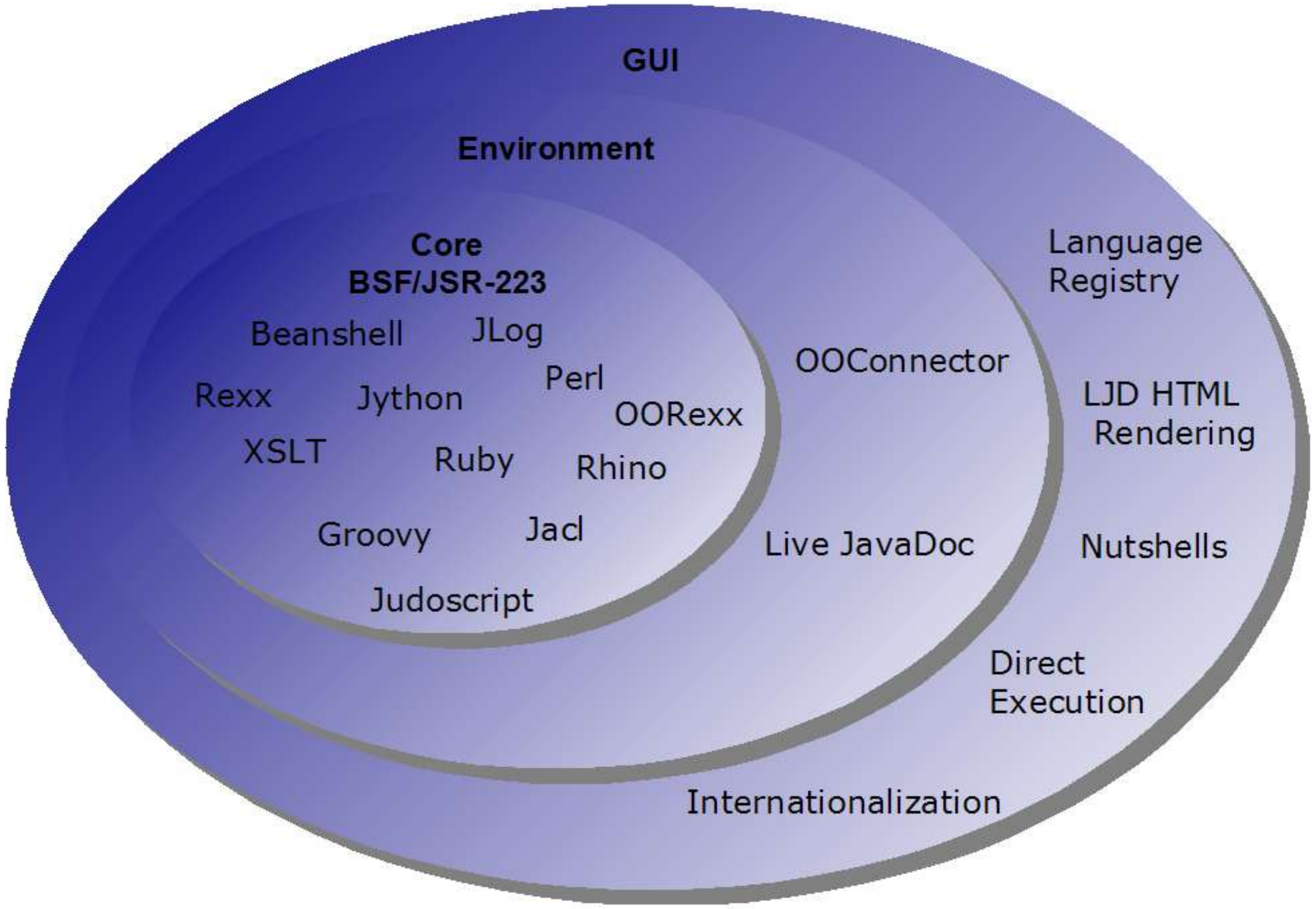


Figure 6 OOBabylon Layer Model, Rainer Hahnekamp

As you can see OOBabylon is divided into 3 Layers beginning with Layer 1 as core, Layer 2 is the environment and Layer 3 as the GUI.

6.3 Components

In this chapter we are going to discuss all 3 layers with their components.

6.3.1 Core

The core consists of BSF and JSR-223. As already mentioned in chapters above, JSR-223 is integrated with its Early Reference Implementation. So only Groovy and ECMAScript are supported. Since PHP needs a ServerContext in some way it had been dropped.
BSF is integrated in its latest version 2.3 with support and configured standard languages Beanshell, Jython, XSLT, Rhino, Jacl, Judoscript and Ruby.
JLog is an reimplementation of Prolog and as you are probably aware of is like XSLT not a scripting language. Prolog is the most popular representative of the 4GL (4th generation language) and is used for AI.
Another language which is not included in the standard distribution is Perl. If you want to use it, please make sure that you have a Perl interpreter installed which is in your path.
Finally OORexx is also a scripting language which is not supported by the standard distribution (NetRexx is, but OORexx replaces it in OOBabylon). If you want to use OORexx you have to do some manual configuration work. Therefore take a look at the documentation at http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/dist.20030611/readme.txt.

6.3.2 Environment

The Environment layer contains in this version of OOBabylon only two useful objects which are registered in the BSF registry and can therefore be used by all scripting languages.

Live JavaDoc

This is the LJD component as you know it from the chapter above. You can use it in your script code via the object name „oobLJD“. For further information please take a look at the nutshells in the OOBabylon GUI at „LJDUsage.js“.

oobComponentLoader

The oobComponentLoader is useful object for OpenOffice.org macros. It connects to the server and return an XcomponentLoader object which you can access via the the lookupBean() method via the name „oobComponentLoader“.

To connect to the server oobComponentLoader uses the url from the configuration file " configuration.xml". The Setup functionality of OOBabylon is explained in detail in an own sub chapter.
For your better understanding here the code how the oobComponentLoader connects to OpenOffice.org:

```
import oob.Setup;

import com.sun.star.bridge.XUnoUrlResolver;
import com.sun.star.uno.UnoRuntime;
import com.sun.star.uno.XComponentContext;
import com.sun.star.lang.XMultiComponentFactory;
import com.sun.star.beans.XPropertySet;
import com.sun.star.connection.NoConnectException;
import com.sun.star.frame.XComponentLoader;

....

            Setup setup = new Setup();
            try {
                  XComponentContext xLocalContext =
com.sun.star.comp.helper.Bootstrap
                              .createInitialComponentContext(null);
                  XMultiComponentFactory xLocalServiceManager = xLocalContext
                              .getServiceManager();
                  Object urlResolver =
xLocalServiceManager.createInstanceWithContext(
                              "com.sun.star.bridge.UnoUrlResolver",
xLocalContext);
                  XUnoUrlResolver xUnoUrlResolver = (XUnoUrlResolver)
UnoRuntime
                              .queryInterface(XUnoUrlResolver.class,
urlResolver);
                  Object initialObject = xUnoUrlResolver.resolve
(setup.getUnoUrl());
                  XPropertySet xPropertySet = (XPropertySet)
UnoRuntime.queryInterface(
                              XPropertySet.class, initialObject);
                  Object context = xPropertySet.getPropertyValue
("DefaultContext");
                  XComponentContext xRemoteContext = (XComponentContext)
UnoRuntime
                              .queryInterface(XComponentContext.class,
context);
                  XMultiComponentFactory xRemoteServiceManager =
xRemoteContext
                              .getServiceManager();
                  Object desktop =
xRemoteServiceManager.createInstanceWithContext(
                              "com.sun.star.frame.Desktop", xRemoteContext);
                  this.xComponentLoader = (XComponentLoader)
UnoRuntime.queryInterface(XComponentLoader.class, desktop);
            }
            catch (NoConnectException nce) {
                  System.err.println("No connection to OpenOffice.org at " +
setup.getUnoUrl());
            }
            catch (Exception e) {
                  e.printStackTrace();
            }
```

With oobComponentLoader you can shorten your code and can start programming right ahead within OpenOffice.org. Please take a look at following example that shows you a macro that opens a new Writer document and writes the text “Hello World” in it.

```
xComponentLoader = bsf.lookupBean("oobComponentLoader");
loadProps = new Array();
xWriterComponent =
      xComponentLoader.loadComponentFromURL("private:factory/swriter",
"_blank", 0, loadProps);
xTextDocument =
      Packages.com.sun.star.uno.UnoRuntime.queryInterface(
      java.lang.Class.forName("com.sun.star.text.XTextDocument"),
xWriterComponent);
xText = xTextDocument.getText();
xText.setString(„Hello World“);
```

6.3.3 GUI

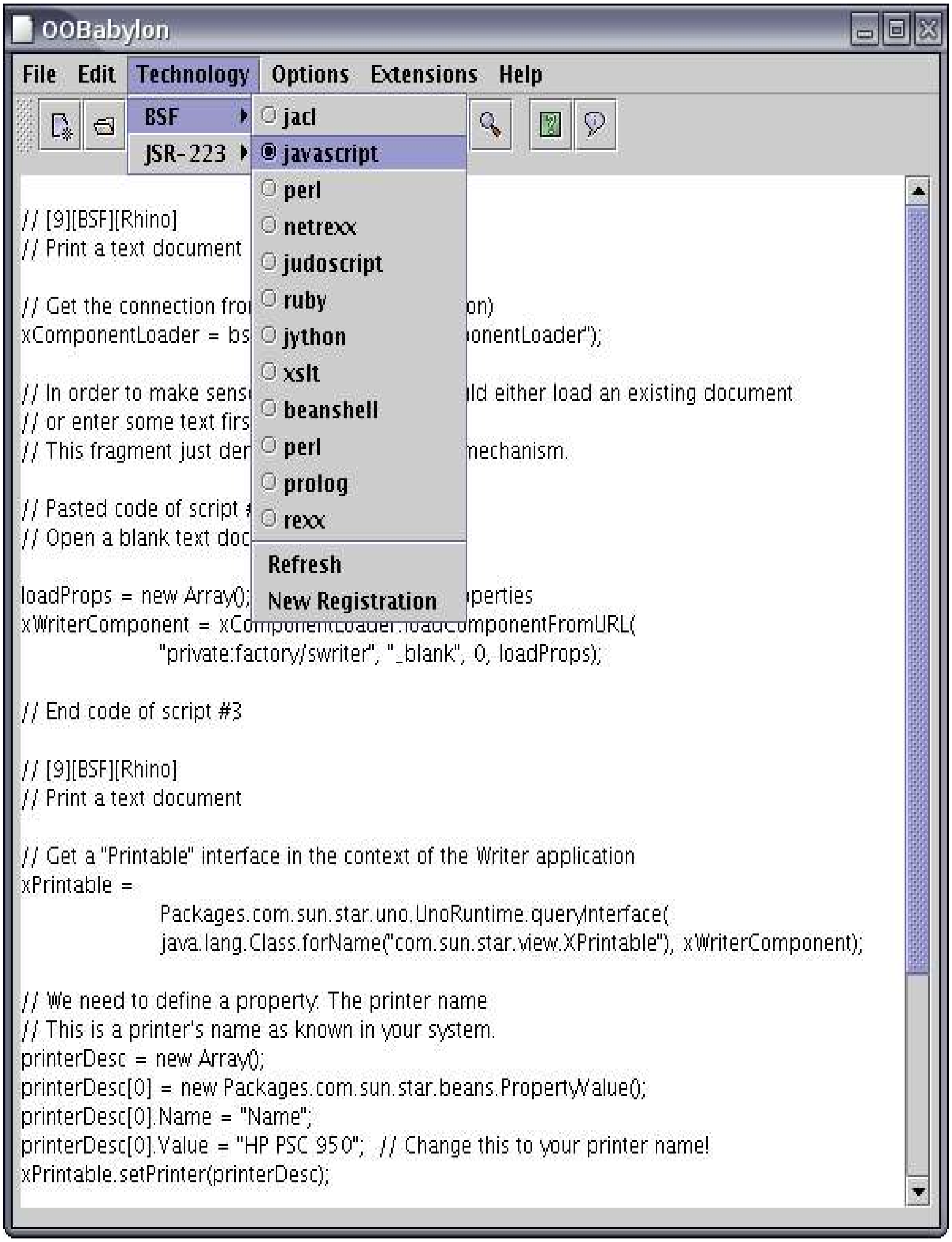
The GUI is on top of the first two layers and has a simple and intuitive design. For example if you open a script file OOBabylon automatically checks the scripting language according to the file's extension.

Internationalization

OOBabylon is available in three languages: German, English and French. You can select your favourite language in the “config.xml”. Since my knowledge of the French language is worse than in English you can imagine how happy I would be if someone could send me a correct translation.
For internationalization OOBabylon the build-in internationalization feature of the Java API. If you want to translate or add a new language to OOBabylon you only have to open the appropriate textfile called MessageBundles_COUNTRYCODE.properties and edit it. That's all!

Direct Execution

OK this is the feature you will probably need most of the time. You write or open a document into OOBabylon's writing area. As mentioned above, if you open a file OOBabylon tries to get the scripting language automatically from the extension of the file and selects it in the menu. You can choose between JSR-223 and BSF. Just open the Technology menu and you will find a submenu for both BSF and JSR-223 with their installed languages.
After writing your code simply click on the play button or type „F5“ to execute the code.
Very easy - very fast!



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

// Get the connection fro                                on)
xComponentLoader = bs                                    onentLoader");

// In order to make sens                                 ld either load an existing document
// or enter some text firs
// This fragment just der                                mechanism.

// Pasted code of script #
// Open a blank text doc

loadProps = new Array();                                 perties
xWriterComponent = xCo                                   mponentFromURL(
                    "private:factory/swriter", "_blank", 0, loadProps);

// End code of script #3

// [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);
```

Figure 7 OOBabylon GUI, Rainer Hahnekamp

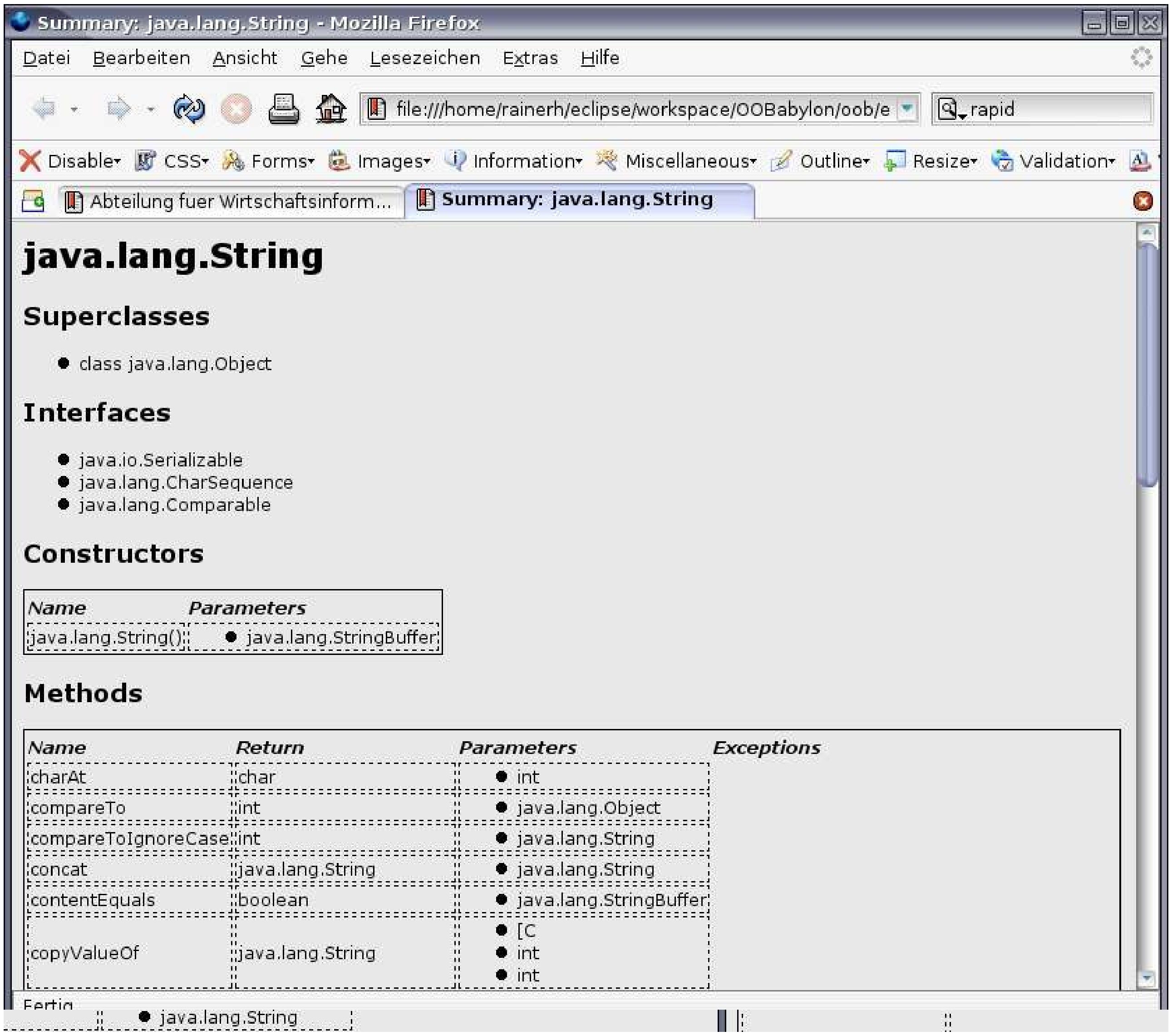
Nutshells

I do not want to discuss the Nutshells in detail since Walter programmed them in both OORexx and Rhino. For more information please read his document. [Augu05]

At the moment OOBabylon only offers Walter's nutshells but I am going to add as soon as possible

- a „Hello World“ script like the one above for every language which is supported
- a special JLog example to demonstrate the power of Prolog
- a demonstration of the Live JavaDoc

LJD HTML Rendering


Summary: java.lang.String - Mozilla Firefox
Datei Bearbeiten Ansicht Gehe Lesezeichen Extras Hilfe
file:///home/rainerh/eclipse/workspace/OOBabylon/oob/e
rapid
Disable CSS Forms Images Information Miscellaneous Outline Resize Validation
Abteilung fuer Wirtschaftsinform...
Summary: java.lang.String
Fertig

java.lang.String

Superclasses

- class java.lang.Object

Interfaces

- java.io.Serializable
- java.lang.CharSequence
- java.lang.Comparable

Constructors

Name	Parameters
java.lang.String()	● java.lang.StringBuffer

Methods

Name	Return	Parameters	Exceptions
charAt	char	● int	
compareTo	int	● java.lang.Object	
compareToIgnoreCase	int	● java.lang.String	
concat	java.lang.String	● java.lang.String	
contentEquals	boolean	● java.lang.StringBuffer	
copyValueOf	java.lang.String	● [C ● int ● int	
		● java.lang.String	

Figure 8 OOBabylon – LJD HTML Rendering, Rainer Hahnekamp

OOBabylon provides the user with the LJD in two types. First it is available as an object in the environment layer. Second it is integrated into the GUI in two ways:

- a preselection of the objects you often use in macros (according to Walter's advice)
- the possibility to analyse every class available which is in the CLASSPATH of OOBabylon via an input field

The result of JLD is translated into an HTML using the XSLT processor Saxon (http://saxon.-sourceforge.net) and shows it immediately in a browser window like the screenshot above. Please notice that you can select the browser in the “config.xml”.

For your better understanding here the XSL file which is used by Saxon to transform the XML file from JLD into a HTML file.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
xmlns="http://www.w3.org./TR/REC-html40">
  <xsl:template match="/class">
    <html>
      <head>
        <title>Summary: <xsl:value-of select="@name" /></title>
        <link href="style.css" rel="stylesheet" type="text/css" />
      </head>
      <body bgcolor="#FFFFFF">
        <h1>
          <xsl:value-of select="@name" />
        </h1>
        <xsl:for-each select="superclasses">
          <xsl:call-template name="superclasses" />
        </xsl:for-each>
        <xsl:for-each select="interfaces">
          <xsl:call-template name="interfaces" />
        </xsl:for-each>
        <xsl:for-each select="constructors">
          <xsl:call-template name="constructors" />
        </xsl:for-each>
        <xsl:for-each select="methods">
          <xsl:call-template name="methods" />
        </xsl:for-each>
        <xsl:for-each select="fields">
          <xsl:call-template name="fields" />
        </xsl:for-each>
        <p>
          Created with Live JavaDoc by <a
href="mailto:Rainer.Hahnekamp@aon.at">Rainer Hahnekamp</a>
```

```
      </p>
    </body>
  </html>
</xsl:template>

<xsl:template name="superclasses">
  <h2>Superclasses</h2>
  <ul>
    <xsl:for-each select="superclass">
      <li>
        <xsl:value-of select="." />
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>

<xsl:template name="interfaces">
  <h2>Interfaces</h2>
  <ul>
    <xsl:for-each select="interface">
      <li>
        <xsl:value-of select="." />
      </li>
    </xsl:for-each>
  </ul>
</xsl:template>

<xsl:template name="constructors">
  <h2>Constructors</h2>
  <table>
    <tr>
      <th>Name</th>
      <th>Parameters</th>
    </tr>
      <xsl:for-each select="constructor">
        <tr>
        <td>
          <xsl:value-of select="/class/@name" />()
        </td>
        <td>
        <xsl:for-each select="parameters">
          <ul>
            <xsl:for-each select="parameter">
              <li>
                <xsl:value-of select="." />
              </li>
            </xsl:for-each>
          </ul>
        </xsl:for-each>
      </td>
    </tr>
    </xsl:for-each>
  </table>
</xsl:template>

<xsl:template name="methods">
  <h2>Methods</h2>
  <table>
    <tr>
      <th>Name</th>
      <th>Return</th>
      <th>Parameters</th>
      <th>Exceptions</th>
    </tr>
    <xsl:for-each select="method">
      <tr>
        <td>
          <xsl:value-of select="name" />
        </td>
        <td>
          <xsl:value-of select="return" />
        </td>
        <td>
          <xsl:for-each select="parameters">
            <ul>
              <xsl:for-each select="parameter">
                <li>
                  <xsl:value-of select="." />
                </li>
              </xsl:for-each>
            </ul>
          </xsl:for-each>
        </td>
        <td>
          <xsl:for-each select="exceptions">
            <ul>
              <xsl:for-each select="exception">
                <li>
                  <xsl:value-of select="." />
                </li>
              </xsl:for-each>
            </ul>
          </xsl:for-each>
        </td>
      </tr>
    </xsl:for-each>
  </table>
</xsl:template>

<xsl:template name="fields">
  <h2>Fields</h2>
  <table>
    <tr>
      <th>Name</th>
      <th>Type</th>
    </tr>
    <xsl:for-each select="field">
      <tr>
        <td>
          <xsl:value-of select="name" />
        </td>
        <td>
          <xsl:value-of select="type" />
        </td>
    </tr>
  </xsl:for-each>
</table>
```

```
</xsl:template>

</xsl:stylesheet>
```

6.3.4 Language Registry

As BSF is in development one can except that more and more languages are becoming supported. Therefore you have the possibility to add a language yourself. All you have to do is to fill in three parameters:

- name of the language
- the complete classname of the BSFEngine
- the extensions

After the registration the new language is stored into the “config.xml” and is loaded every time you execute your code.

6.4 Levels

As macro developer you have the possibility to use OOBabylon in three different types. These are:

- GUI: The GUI has been discussed above.
- Command Line: You have the possibility to execute script code by just calling OOBabylon over the command line. For example you can write tcl-code with Emacs or another editor, change to the console and type in:

```
OOBabylon –help #lists all available parameters
OOBabylon -file openoffice.tcl
```

- Library: The third type of usage is to integrate OOBabylon in your own application using it as a library. In this case you do not have the possibility to use all features. To execute script code for example you have to manually instantiate the class oob.bsf.Runtime and trigger the execute method by yourself. The advantage is that you are more flexible and can for example program your own GUI or use OOBabylon in a Servlet.

6.5 Architecture

In this sub chapter you should get a little insight into the architecture of OOBabylon. I want to underline that this chapter is not intended that get full knowledge how OOBabylon is programmed. You should only get a glance of it. If you are interested take a look at the source code or contact me.

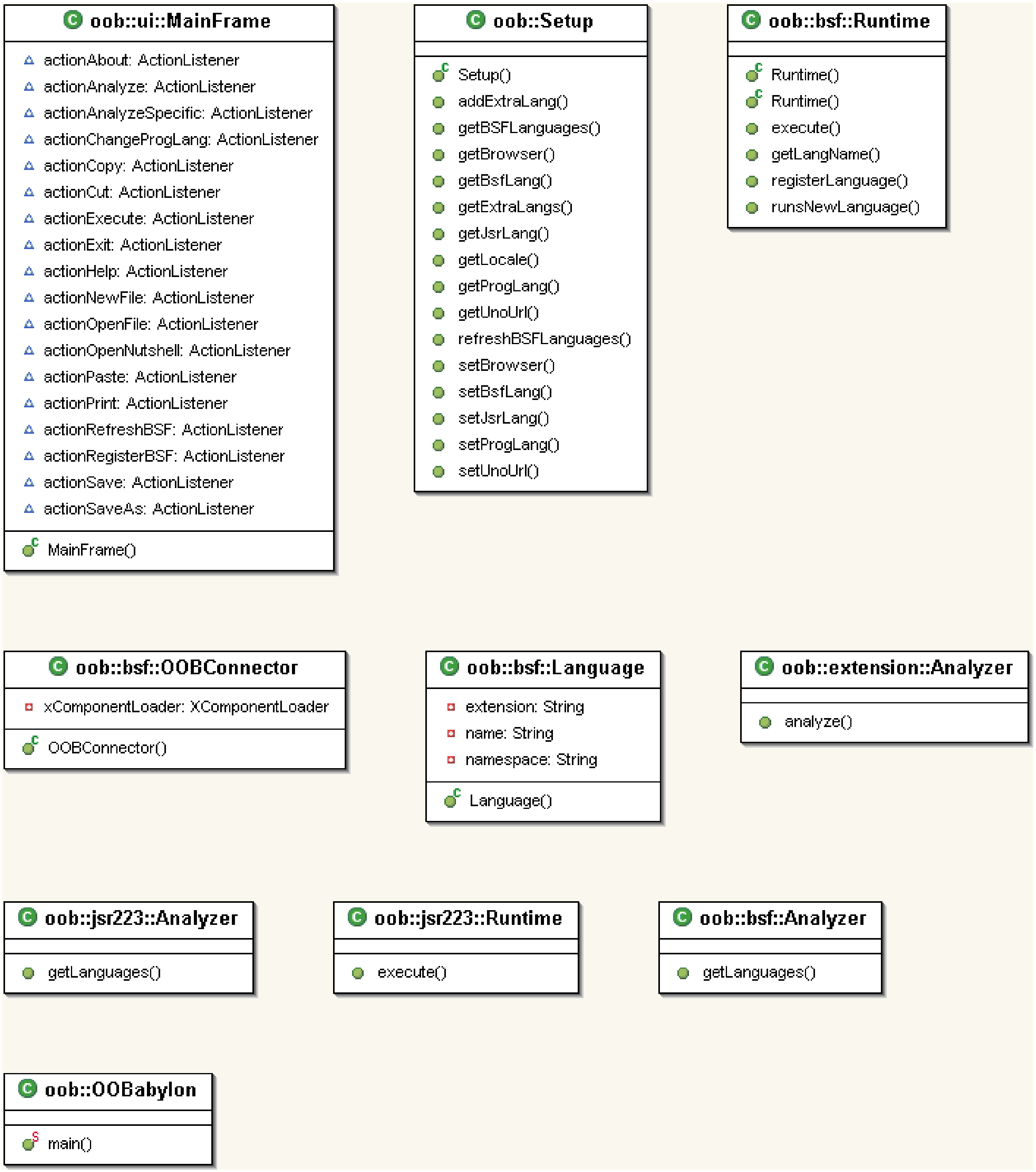


Figure 9 OOBabylon Architecture, Rainer Hahnekamp

OOBabylon is divided into following packages:

- oob: The root package contains the Setup and the OOBabylon class. Setup is used to get and set the information of the “config.xml” and OOBabylon for starting the application.
- oob.ui: This package holds the JFrame object and represents the main window with all its events and methods.
- oob.bsf: This packages includes the needed classes for BSF. Main class is the Runtime that initiates the BSFManager, registers all languages and the objects from the environment layer.
- oob.jsr223: This package is the same as the bsf package but runs JSR-223 without the functionalities like object registry of the bsf package.
- oob.extension: This package includes all extensions. In the latest version it is only the Analyzer that is the interface to JLD and the nutshells which are only available as text files.

6.6 Main Activity

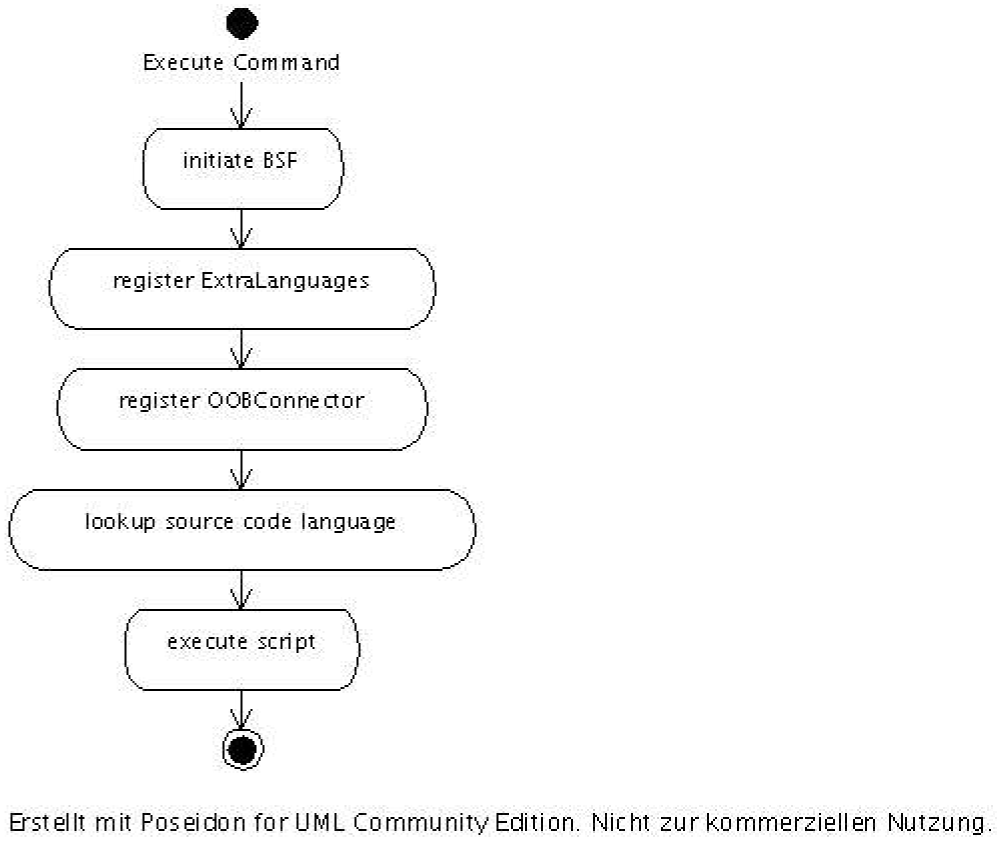

Execute Command
initiate BSF
register ExtraLanguages
register OOBConnector
lookup source code language
execute script
Erstellt mit Poseidon for UML Community Edition. Nicht zur kommerziellen Nutzung.

Figure 10 OOBabylon Activity, Rainer Hahnekamp

Here you can see the main activity of a typical usage of OOBabylon. You type in your code and press the execute button. Immediately the BSFManager is instantiated the languages and classes registered and finally – if all works - the script is executed.

6.7 Setup

The standard config.xml looks has following content:

```
<?xml version="1.0" encoding="UTF-8"?>
<config>
     <progLang>English</progLang>

<bsfAvailableLang><name>jacl<extension>jacl</extension></name><name>javascrip
t<extension>js</extension></name><name>netrexx<extension>nrx</extension></nam
e><name>judoscript<extension>judo</extension><extension>jud</extension></name
```

```
><name>ruby<extension>rb</extension></name><name>jython<extension<py</extensi
on></name><name>xslt<extension>xslt</extension></name><name>beanshell<extensi
on>bsh</extension></name></bsfAvailableLang>
      <bsfExtraLangs>
<lang><name>perl</name><namespace>net.sourceforge.bsfperl.PerlEngineImpl</nam
espace><extension>pl</extension></lang><lang><name>prolog</name><namespace>ub
c.cs.JLog.API.bsf.JLogBSFEngine</namespace><extension>plog</extension></lang>
<lang><name>rexx</name><namespace>org.apache.bsf.engines.rexx.RexxEngine</nam
espace><extension>rexx</extension>
      </bsfExtraLangs>
      <browser>mozilla-firefox</browser>

<unoUrl>uno:socket,host=localhost,port=8100;urp;StarOffice.ServiceManager</un
oUrl>
</config>
```

- progLang: This entity contains the language of OOBabylon. You can choose between „Deutsch“, „English“ or „French“
- bsfAvailableLang: This entity contains the languages which are registered in the “language.properties” file of the bsf.jar and are also working. OOBabylon is able to check if languages are functioning properly if you press the Refresh button in the BSF-Menu.
- bsfExtraLangs: These are languages that are not registered in the “language.properties” file but were added manually.
- browser: The command which is executed when OOBabylon displays the result of LJD, should logically be a browser.
- unoUrl: This is the url the oobComponentLoader connects at the beginning.

6.8 Summary

This chapter introduced you to OOBabylon that was developed to extend the scripting abilities of OpenOffice.org with BSF/JSR-223 in a way that hides all its complexity from the macro developer.
You should know that OOBabylon consists of 3 layers with many useful components. Further more OOBabylon can be used in different ways: In its main mode as a GUI, as a library or as a console application.

Also have in mind that one of the main components is the Live JavaDoc which allows you to analyse all Java classes you come across. You can use it in your scripts especially to get more information on some classes from UNO.
OOBabylon does not have an own setup dialog but has a “config.xml” where all setup parameters are stored and can be edited.

7 Outlook

As you now know what OOBabylon and LJD are and how you can use them I want to give you some information about future plans.

OOBabylon and LJD will be tested some parts of code redesigned and after that I am going to submit them to www.sourceforge.net where everybody can take part in the development.

OOBabylon will further be improved with nutshells in different languages. I will try to concentrate more on Prolog examples so that you can see how different problems can be solved with logical programming languages.

As final step I am going to contact the OpenOffice.org community and offer them OOBabylon for further usage. Licence will be LGPL.

8 References

[Augu05] Augustin, Walter: Examples for Open Office Automation with Scripting Languages. http://wi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/bsf.ooffice/Augustin%20SE-Arbeit%209250416.pdf, 2005, retrieved on 2005-02-01

[Bsf02] Apache Jakarta: Jakarta BSF - Frequently Asked Questions. http://jakarta.apache.org/bsf/faq.html, 2002, retrieved on 2005-02-01

[Flat03] Flatscher, Rony G.: The Augsburg Version of BSF4Rexx. http://wi.wu-wien.ac.at/rgf/rexx/orx14/orx14_bsf4rexx-av.pdf, 2003, retrieved on 2005-02-01

[Flat04a] Flatscher, Rony G.: BSF, Part 1. http://wi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/folien/AutoJava_07.pdf, 2004, retrieved on 2005-02-01

[Flat04b] Flatscher, Rony G.: Camouflaging Java as Object Rexx. http://wi.wu-wien.ac.at/rgf/rexx/orx15/2004_orx15_BSF-ORX-layer-2.pdf, 2004, retrieved on 2005-02-01

[HaNe01] Hans-Robert Hansen, Gustaf Neumann: Wirtschaftsinformatik I. Lucius & Lucius, Stuttgart 2001

[Kale01] Kalender Peter: A Concept for and an Implementation of the Bean Scripting Framework for Rexx. http://nestroy.wi-inf.uni-essen.de/Lv/seminare/ws0001/PKalender/Seminararbeit.pdf, 2001, retrieved on 2005-02-01

[NeZd05] Neumann, Gustaf. Zdun, Uwe: XOTcl - Tutorial. http://media.wu-wien.ac.at/doc/tutorial.html, 2005, retrieved on 2005-02-01

[Open03] OpenOffice.org: OpenOffice.org Developer's Guide. http://api.openoffice.org/docs/DevelopersGuide, 2003, retrieved on 2005-02-01

[Orli02] Victor J. Orlikowski: An Introduction to the Bean Scripting Framework. http://www.dulug.duke.edu/~vjo/papers/ApacheCon_US_2002/intro_to_bsf.pdf, 2002, retrieved on 2005-02-01

[Sun04a] Sun Microsystems: The Java Community(SM) Program - Introduction - FAQ. http://www.jcp.org/en/introduction/faq, 2004, retrieved on 2005-02-01

[Sun04b] Sun Microsystems: JSR 223: Scripting Pages in Java Web Applications - Installation and Getting Started Guide (Early Access). http://javashoplm.sun.com/ECom/docs/Welcome.jsp?StoreId=22&PartDetailId=php-1_0-ea-doc-oth-JPR&SiteId=JCP&TransactionId=noreg, 2004, retrieved on 2005-02-01

[Udkp05] OpenOffice.org: UNO Development Kit Project. http://udk.openoffice.org/, 2005, retrieved on 2005-02-01

[Wall00] Wall, Larry et al: Programming Perl. O'Reilly, Sepastopol 2000

[Wiki04] Wikipedia: REXX. http://en.wikipedia.org/wiki/Rexx, 2004, retrieved on 2005-02-01

[Wiki05a] Wikipedia: Skriptsprache. http://de.wikipedia.org/wiki/Skriptsprache, 2005, retrieved on 2005-02-01

[Wiki05b] Wikipedia: Python programming language. http://en.wikipedia.org/wiki/Python_programming_language, 2005, retrieved on 2005-02-01

[Wiki05c] Wikipedia: OpenOffice.org. http://de.wikipedia.org/wiki/OpenOffice, 2005, retrieved on 2005-02-01

9 Appendix: Link List

General

http://en.wikipedia.org/wiki/Main_Page	Wikipedia, the number one online encyclopaedia
http://www.heise.de/	Heise online – one of the best resources about IT in German language

BSF/JSR-223

http://jakarta.apache.org/bsf/	Official Site of the Bean Scripting Framework
http://rexxla.org/oorexx/	Official Site of OORexx
http://wi.wu-wien.ac.at/rgf/rexx/	
http://wi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/folien/AutoJava_07.pdf	Prof. Flatscher's resources about Rexx in general and BSF4Rexx
http://wi.wu-wien.ac.at/Studium/LVA-Unterlagen/rgf/autojava/folien/AutoJava_08.pdf	
http://www.mozilla.org/rhino/	Official Site of Rino
http://www.tcl.tk/software/java/	Official Site of Jacl
http://www.jython.org/	Official Site of Jython
http://www.perl.com/	The source for Perl – Perl development, perl conferences
http://www.perl.org/	Perl Directory
http://cpan.org/	Comprehensive Perl Archive (Modules)
http://www.judoscript.com/index.html	Official Site of Judoscript
http://www.beanshell.org/	Official Site of Beanshell
http://groovy.codehaus.org/	Official Site of Groovy
http://jruby.sourceforge.net/	Official Site of JRuby
http://xml.apache.org/xalan-j/	Java version of Xalan (XSLT Processor)
http://jlogic.sourceforge.net/	Official Site of the original JLog
http://www.ulfdittmer.com/jlog/index.html	Official Site of Ulf Dittmer's JLog which is used in BSF
http://www.jcp.org/en/jsr/detail?id=223	Official Site of JSR-223
http://www.jcp.org/	Official Site of Java Community Process

Java

http://java.sun.com/docs/books/tutorial/i18n/	Internationalization tutorial
http://java.sun.com/docs/books/tutorial/reflect/	Reflection tutorial
http://java.sun.com/docs/books/tutorial/uiswing/	Swing tutorial
http://www.mindview.net/Books/TIJ/	**Bruce Eckel's book Thinking in Java**
http://java.sun.com/reference/api/index.html	Java API

Open-Source & Linux

http://www.opensource.org/	The official site of the Open Source Initiative with all licenses

http://www.dbus.de/eip/inhalt.html	**Open Source Tutorial – good introduction into the world of Open-Source (German)**
http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/	**A must-have for everybody who is interested in OS in general**
http://www.catb.org/~esr/writings/cathedral-bazaar/magic-cauldron/	Another book of E. Raymond about business models in OS
http://www.debian.org/	In my opinion the best Linux distribution
http://linuxdocs.org	Collection of How-Tos
http://www.linuxquestions.org/	Web Forum for all areas about Linux
http://www.gnu.org/	Official Site of the GNU project
http://www.apache.org/	Official Site of the Apache Software Foundation
OpenOffice.org	
http://www.openoffice.org/	Official Site of OpenOffice.org
http://download.openoffice.org/680/	Beta versions of the upcoming OpenOffice.org
http://www.ooomacros.org/	OpenOffice.org Macros
http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html	OpenOffice.org developer's guide
http://ooextras.sourceforge.net/	OpenOffice.org Extras
Web Server Technologies	
http://java.sun.com/j2ee/1.4/docs/tutorial/doc/index.html	**J2EE 1.4 Tutorial**
http://www.jboss.org/	Official Site of JBoss (Open-Source Application Server)
http://www.theserverside.com/	Enterprise Java Community with free books
http://msdn.microsoft.com/	Microsoft Developer Network – Number 1 for Microsoft programmers
http://www.theserverside.net/	Counterpart of the theserverside.com for the .NET Framework
http://www.php.net/	Official Site of PHP
http://pear.php.net/	PHP Extension and Application Repository
http://de.selfhtml.org/	**SelfHTML a popular german introduction to web development around HTML**
XML and its technologies	
http://w3c.org/	The World Wide Web Consortium – the central point of the Internet
http://www.w3schools.com/	**Tutorials for nearly all technologies in a short but sufficient way**