Projektseminar aus Wirtschaftsinformatik

Creation and Installation of BSF4ooRexx on MacOSX

Seminararbeit

26.01.2011

Jürgen Hesse

Manuel Paar

Table of contents

1 Introduction

The following chapters will provide an insight into the world of ooRexx, BSF4ooRexx and MacOSX.

1.1 ooRexx

The programming language Open Object Rexx (ooRexx) is an object-oriented scripting language that was developed by IBM for its operating system OS/2.
It is an enhancement of the classic programming language Rexx and was initially developed for the operating system VM/SP, and later ported to the platforms MVS, OS/2 and PC-DOS.
In October 2004, IBM gave the ooRexx source code for a couple of operating systems, including Microsoft Windows, Sun Solaris and Linux, to the Rexx Language Association (RexxLA), which released it under the Common Public License (CPL) [**1**].
The clear advantage of ooRexx is, that its syntax has large similarities to the English language and therefore it is very easy to learn and to understand. Besides its simplicity, it is a very mighty programming language and fully supports object oriented programming. Additionally, ooRexx offers full downward compatibility to the classic REXX language [**2**].

1.2 BSF4ooRexx

The “Bean Scripting Framework for Open Object Rexx” (in short BSF4ooRexx) is a project that enables a developer to access ooRexx from Java and vice versa.
What started as proof-of-concept-project in the context of a seminar at the University of Essen, Germany, in winter semester 2000/2001, rapidly developed into a highly interesting project with a large number of followers. In 2003, the “Augsburg” version of BSF4ooRexx was released, which made it possible to load Java from ooRexx and to interact with Java objects. The access itself was quite complex at this time and constantly required the developer to explicitly define the Java data types when calling a Java Method.
Because ooRexx, and therefore also BSF4ooRexx, are built upon the principle of simplicity, this problem had to be solved. Only three years later, the “Vienna” version was released which not only improved the access to Java drastically, but also implemented support for OpenOffice [**3**].

BSF4ooRexx makes it, for example, possible to create graphical user interfaces (in short GUIs) or to automate Java applications from within ooRexx scripts. Especially the last point becomes very important in connection with the automation of OpenOffice and lets one sense the high value of this project.

1.3 Apple MacOSX

MacOSX from Apple Inc. is the most successful UNIX-based operating system at the moment [**4**]. The current version 10.6 (as of January 2011) with the code name "Snow Leopard" has a largely improved support for 64 bit applications. The most preinstalled software programs run in 64 bit mode, but can also be started in 32 bit mode [**5**]. Initially, MacOSX 10.6 was released on 28th August, 2009.

MacOSX consists of two components [**6**]:

- The UNIX-Kernel, called „Darwin" and
- the graphical user interface, called „Aqua", which delivers the typical Mac „Look and Feel"

1.4 Apple's Java

Along with its operating system, Apple delivers its own version of the Java programming language, which contains the same components as the original version from Oracle (formerly by Sun). To enhance compatibility, Apple's Java can be run in 32 and 64 bit mode and is implemented as a framework[1].

In October 2010, Apple released the latest update for its Java version and announced, that there will be no Java support in future MacOSX versions anymore. By its own account, Apple instead wants to work together with Oracle on an Open Source version of Java called OpenJDK, in order to create a version compatible with MacOSX. This is a logical step if one keeps in mind that the Java versions from Apple always lag behind of those from Oracle [**7**], [**8**], [**9**].

[1] The exact distinction between "bundles" and "frameworks" will be discussed later in chapter 3 "Usage of BSF4ooRexx on MacOSX".

2 BSF4ooRexx on MacOSX

The following chapters will take a closer look at the main goal of this work, namely the creation of BSF4ooRexx on MacOSX.

2.1 The Problem

The main problem, upon which this paper has been created, was, that there are BSF4ooRexx versions for Microsoft Windows and several Linux distributions, but none for MacOSX.

Especially because of the constantly growing popularity of Apple computers and the permanently growing number of MacOSX users, it was only a matter of time until the growing pressure from the user-community led to the necessity of porting BSF4ooRexx to this platform.

Although runnable versions of BSF4Rexx[2] for earlier versions of MacOS existed, it was not possible to create a BSF4ooRexx version for MacOSX until now. The reasons for this were serious changes Apple made to its operating system while jumping from version 9 to 10.

In order to solve this problem and to create a suitable version for MacOSX, among other things the following challenges have to be met:

1. Creation of a suitable Makefile for the compilation of BSF4ooRexx's dynamic library (in short "dylib") using the `g++` compiler,
2. adaption of BSF4ooRexx's C++ source code to MacOSX,
3. compilation of the source code and creation of 32 and 64 bit compatible dynamic libraries,
4. setup of BSF4ooRexx for a given system[3].

The preconditions for successfully carrying out the steps above, are as follows:

1. ooRexx is installed in the same bitness as the BSF4ooRexx dynamic library that will be compiled and
2. the XCode Development Environment is installed. It contains the programs that are required for compiling the C++ source code. Although it comes with every Apple computer, XCode must be installed separately.

[2] BSF4ooRexx's predecessor for earlier versions of classic Rexx.
[3] This step will be discussed in further detail in chapter 3 "Usage of BSF4ooRexx on MacOSX".

2.2 Compiling BSF4ooRexx on MacOSX

In the following parts of this document, the steps mentioned above will be described in more detail.

2.2.1 Creating the Makefile

For easier compilation of the dynamic libraries, the first step was to create a Makefile that handles all necessary actions. Makefiles are, basically, simple text files that tell the compiler in which way the given source code has to be compiled.
In our case, for example, the path settings for the Java Virtual Machine (JVM) and the ooRexx include files are defined through variables as shown in listing 1, which are later used during the compilation process.

```
INC_PATH = -I. -I/System/Library/Frameworks/JavaVM.framework/headers

INC_PATH_ORX = -I/opt/ooRexx/include
```

Listing 1: Makefile variable.

In order to create a 32 bit version of BSF4ooRexx, the `g++` commands of listing 2 have to be executed[4].

```
g++ -c -fPIC $(INC_PATH) -I$(INC_PATH_ORX) -m32
 ↪ -arch i386 -DUSE_OREXX -DUNIX -DBSF4REXX_32_BIT
 ↪ -oBSF4ooRexx-mac-i386.o BSF4ooRexx.cc

g++ -dynamiclib -shared -o libBSF4ooRexx-i386.dylib
 ↪ BSF4ooRexx-mac-i386.o /usr/lib/librexx.dylib
 ↪ /usr/lib/librexxapi.dylib -framework JavaVM -arch i386
```

Listing 2: Compilation commands (32 bit).

On the other hand, if a 64 bit version has to be created, slightly different commands are necessary, as depicted in listing 3.

```
g++ -c -fPIC $(INC_PATH) -I$(INC_PATH_ORX) -m64
 ↪ -arch x86_64 -DUSE_OREXX -DUNIX -DBSF4REXX_64_BIT
 ↪ -oBSF4ooRexx-mac-x86_64.o BSF4ooRexx.cc

g++ -dynamiclib -shared -o libBSF4ooRexx-x86_64.dylib
 ↪ BSF4ooRexx-mac-x86_64.o /usr/lib/librexx.dylib
 ↪ /usr/lib/librexxapi.dylib -framework JavaVM -arch x86_64
```

Listing 3: Compilation commands (64 bit).

[4] The arrow symbol (↪) indicates a word wrap and not a printable character.

After successfully creating a Makefile, the compilation can be started using the `make` command as given in listing 4.

```
make --makefile=Makefile
```

Listing 4: Compilation.

A Makefile capable of creating either a 32 or 64 bit dynamic library is depicted in chapter 5.2 in the Appendix.

In our case, several errors occurred during the first run. These errors made it necessary to adapt the BSF4ooRexx's source code. The next section will take a closer look at these changes.

2.2.2 Adapting the C++ Source Code

Because of different compiler-side requirements (in our case from `g++`), it was necessary to add a series of explicit casts to method calls in the BSF4ooRexx source code. This circumstance was especially surprising, because the same source code compiled without any problems on Windows and Linux.

After fixing these problems, it was possible to create the first runnable dynamic library, which made it possible to run ooRexx from Java using the program `rexxj2.sh`.

In order to cover the second use case, namely to run and access Java from within an ooRexx script (by using the program `rexx`), further changes to the source code had to be made. The main problem origins from the fundamental design of MacOSX's graphical user interface (short GUI), called Aqua.

In Java on MacOSX, the JVM's (Java Virtual Machine) main thread is used to process the events from the GUI. To do so, it has a so-called `CFRunLoop` object. *"A `CFRunLoop` object monitors sources of input to a task and dispatches control when they become ready for processing. Examples of input sources might include user input devices, network connections, periodic or time-delayed events, and asynchronous callbacks."* [**10**]

Access to the GUI is required when Java's Abstract Window Toolkit (short AWT) is used. If AWT is not required, it is not necessary that the main thread has a `CFRunLoop` object.

ooRexx, on the other hand, does not have a `CFRunLoop` object at the time of writing this document. If it is started first and tries to use Java's AWT via BSF4ooRexx, the

program hangs because AWT has to be called from the main thread that handles `CFRunLoop` in order to gain access to the GUI.
Therefore, if a script has to be executed that uses AWT, the program `rexxj2.sh` has to be used. If AWT is not required, the script can be executed with either `rexx` or `rexxj2.sh`.

2.2.3 Compilation

Using the Makefile created before, together with the adapted C++ source code, it is now possible to create fully functional dynamic libraries for BSF4ooRexx on MacOSX in a 32 and 64 bit version.

3 Usage of BSF4ooRexx on MacOSX

The following chapters will take a closer look on the possibilities for using BSF4ooRexx on MacOSX and will make a recommendation for the best solution to integrate BSF4ooRexx in MacOSX.

3.1 Types of Bundles

In MacOSX, software is mostly organized in Bundles. A Bundle is a directory, which contains an executable software program and all its dependent components (e.g. icons, documentation, etc.). This architecture makes it possible to gather all required resources for running the program in one single place. As a result, installation and deinstallation becomes very easy. In the case of installation, the resources are stored in a single directory and don't have to be spread across the operating system. On the other hand, deinstallation is very simple because the only thing the user has to do in order to cleanly remove all the dependent resources, is to drop the bundle directory into MacOSX's recycle bin [**11**].

Basically, it is possible to differentiate between three different kinds of bundles [**11**]:

- Application bundle,
- loadable bundle and
- framework bundle.

3.1.1 Application Bundle

An application bundle (e.g. `Editor.app`) contains all necessary resources for running a specific application (in this example an editor). Although a bundle basically is an ordinary directory with the postfix `.app` and may contain several subdirectories, MacOSX interprets it as a single executable application. Such applications are stored in the directory `/Applications` under MacOSX.

The bundle's content is normally hidden from the user. Nevertheless, it can be displayed by right-clicking the Bundle icon and selecting "Show package content", as shown in figure 1.

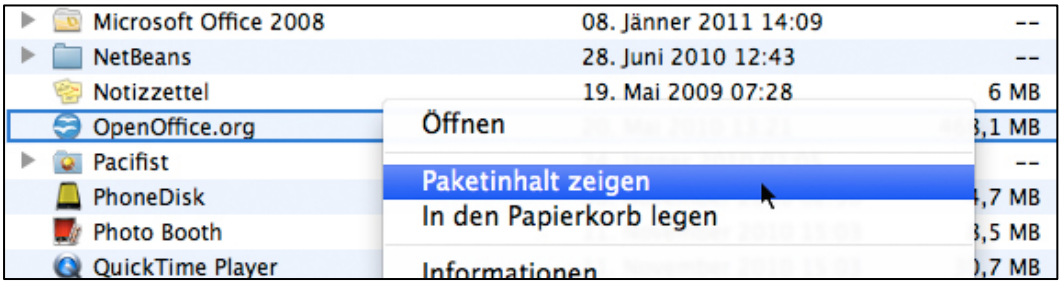

Figure 1: How to open a package content.

For example, a bundle directory could be organized like that depicted in figure 2.

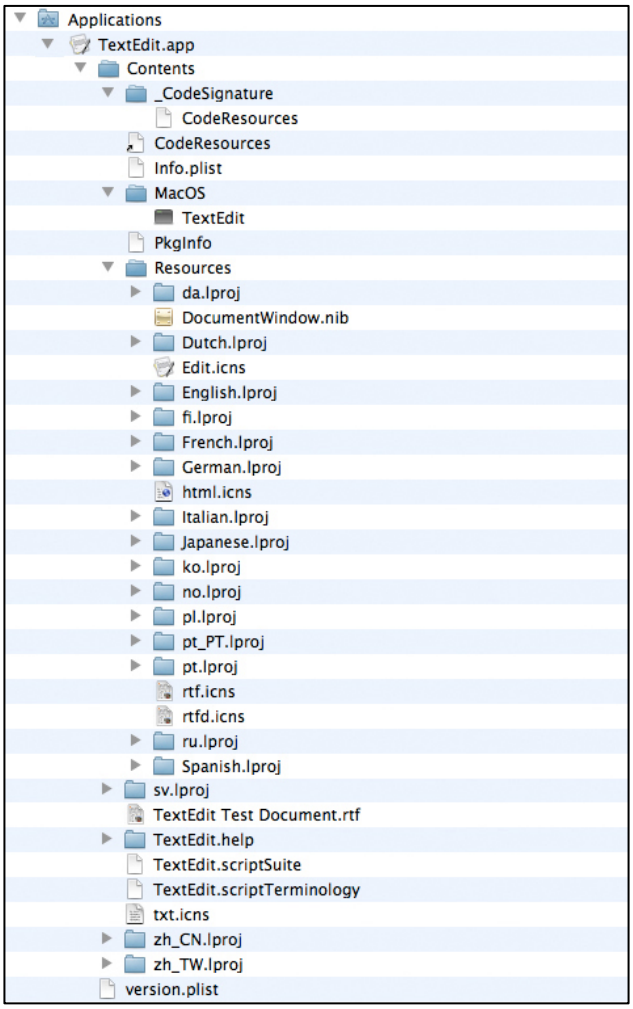

Figure 2: Directory structure of the TextEdit application bundle.

The file `Info.plist` in the `Contents` directory is a so-called property list, which contains meta data about the bundle. Property lists are written in XML format and include information like the name of the bundle, the bundle's icon, location of the documentation and so on.

The code snippet in listing 5 shows an example for such a file.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
    "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
 <dict>
       <key>CFBundleDevelopmentRegion</key>
       <string>English</string>
       <key>CFBundleDisplayName</key>
       <string>TextEdit</string>
       <key>CFBundleDocumentTypes</key>
       <array>
             <dict>
                   <key>CFBundleTypeIconFile</key>
                   <string>rtf.icns</string>
                   <key>CFBundleTypeName</key>
                   <string>NSRTFPboardType</string>
                   <key>CFBundleTypeRole</key>
                   <string>Editor</string>
                   ...
```

Listing 5: Info.plist example.

3.1.2 Loadable Bundle

A loadable bundle offers the possibility to extend the kernel of the MacOSX operating system. Such a bundle has normally a file extension like `.kext` and is stored in the `/System/Library/Extensions` folder [**11**].

We quote it here for completeness only, because we have no use for such a bundle in our term paper.

3.1.3 Framework Bundle

Special forms of bundles on a MacOSX operating system are frameworks. They are stored in the `/System/Library/Frameworks` folder and contain resources like header files, documentation and so on. Executable files may also be included.

The difference to conventional bundles is that one framework can contain different versions of this framework. Other bundles only contain one version.

This has the advantage that version conflicts can be avoided and the compatibility with older programs can be guaranteed [**11**], [**12**].

Furthermore a framework can contain “shared resources”. On MacOSX, libraries that act as such shared resources are called “dynamic shared libraries” (dylib) and can be used by many different applications simultaneously. Shared resources will only be loaded by the system when they are required.

For this purpose, a write-protected copy of the framework will be loaded into the memory. From now on, applications will use this cached copy only. As a result, the

performance of the whole system is greatly improved, because data in the memory can be accessed much faster than data on the hard disk.
A special form of a framework is a so-called “umbrella framework”.
Such a framework includes one or more sub-frameworks [**12**].

For example, a framework directory could be organized like that depicted in figure 3.

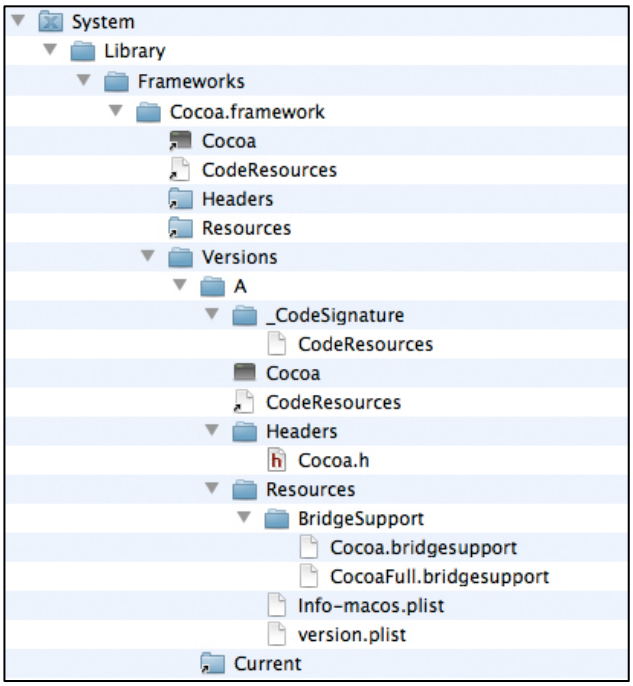

Figure 3: Directory structure of the Cocoa framework.

Thanks to this example, the versioning is clearly visible: In the root directory (`Cocoa.framework`) are many different alias-resources (they are represented with a small arrow on the icon). They are all linked to resources, which are located in the `Current` folder, which is located in the `Versions` directory.
The `Current` folder refers to the current framework version. The different versions of the framework are stored in separate directories in the `Versions` folder. In this example, there is only one version, which is stored in the `A` directory.

3.2 Recommendation to Integrate BSF4ooRexx in MacOSX

In the following chapters we will explain which deployment method is the best for BSF4ooRexx and how it is carried out.

3.2.1 Choosing a Bundle Type

As already mentioned above, on MacOSX, there are three different possibilities to create a bundle.

BSF4ooRexx is not a kernel extension, so a loadable bundle is not a suitable choice.

Also application bundles are not appropriate in this case, because such a kind of bundle is intended for executable applications and their resources (e.g. samples and utilities).

BSF4ooRexx is a “dynamic library”, which is used by many different applications simultaneously, like Java, OpenOffice and ooRexx. Therefore, the third option, a bundle in form of a framework, is the best choice.

3.2.2 Creating the BSF4ooRexx Framework

To implement BSF4ooRexx as a framework on MacOSX, a directory structure as shown in figure 4 could be used.

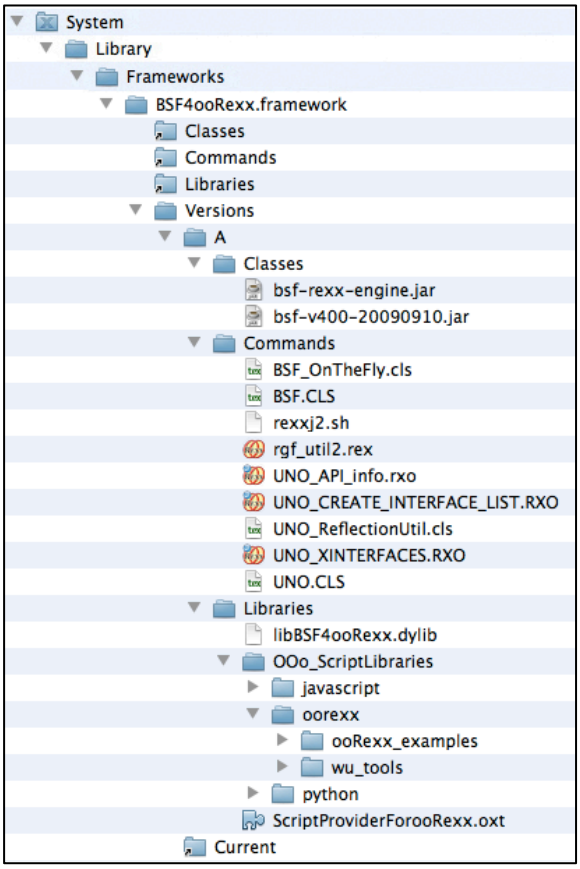

Figure 4: Directory structure of the BSF4ooRexx framework.

For a better overview, we put the needed resources in different directories (`Classes`, `Commands`, and `Libraries`).

`Classes` contains the `.jar` files and the `Commands` folder the executable files. The `Libraries` directory includes the dynamic library, which was compiled on MacOSX, the OpenOffice extension, `ScriptProviderForooRexx.oxt`, and the OpenOffice script libraries for all platforms. The correct setup of the `.oxt` file is described later in chapter 3.2.5.2 “Run “unopkg” to Add the ooRexx Extension to OpenOffice”.

Like in the preceding example, in the root directory (`BSF4ooRexx.framework`) there are three alias-resources. They all are linked to resources, which are located in the `Current` folder, which is located in the `Versions` directory.

The `Current` folder refers to the current framework version, which is stored in the `A` directory. After creating this directory structure, the framework has to be set up for a correct functionality. The necessary steps, to set up the framework bundle, are described in the following chapter.

3.2.3 Automating the Framework Creation Using a Makefile

To make the framework creation quicker and easier, the Makefile shown in listing 6 has been created. A Makefile allows defining so-called “targets”, each of which contains a set of commands. By doing so, commands for different tasks can be bundled together. To execute the targets in a Makefile the UNIX program `make` is required. To use `make` on MacOSX, it is necessary to install the Apple Development Environment “XCode”. [**13**]

In our case, the Makefile consists of a series of variable definitions and commands, which build the desired framework. The most important variables are `SRC_BSF`, `SRC_DYLIB` and `DST`.

`SRC_BSF` defines where the Makefile should look for the BSF4ooRexx install files. These files can be easily downloaded from the BSF4ooRexx homepage [**14**].

`SRC_DYLIB`, on the other hand, defines, where the dynamic library (as compiled in chapter 2.2 “Compiling BSF4ooRexx on MacOSX”), that has to be used for the framework, is stored.

The last variable to be discussed in more detail is `DST`. This variable defines were the output, namely the finally assembled framework itself, will be stored. By default, the framework will be created in the same directory where the Makefile is located.

```
# Define source directories
SRC_BSF = BSF4ooRexx_install
SRC_DYLIB = BSF4ooRexx
SRC_REXXJ2 = BSF4ooRexx


# Define destination directories
DST = BSF4ooRexx.framework
VERSION_A = $(DST)/Versions/A
CURRENT = $(DST)/Versions/Current
CLASSES = $(VERSION_A)/Classes
COMMANDS = $(VERSION_A)/Commands
LIBRARIES = $(VERSION_A)/Libraries

default:
        @echo
        @echo "Usage:"
        @echo "make {clean|build32|build64}"
        @echo
        @echo "clean:"
        @echo "   Cleans the destination directory ($(DST))."
        @echo "build32:"
        @echo "   Builds the framework with 32 bit libraries."
        @echo "build64:"
        @echo "   Builds the framework with 64 bit libraries."
        @echo
        @echo

clean:
        @echo "-----------------------------------"
        @echo "Cleaning..."
        @echo "-----------------------------------"
        @echo
        @echo "Removing old directory structure..."
        @echo
        rm -rf $(DST)
        @echo
        @echo "Done."
        @echo

build:
        @echo "-----------------------------------"
        @echo "Building..."
        @echo "-----------------------------------"
        @echo
        @echo "Creating directory structure..."
        @echo

        mkdir -p $(CLASSES)
        mkdir -p $(COMMANDS)
        mkdir -p $(LIBRARIES)

        @echo
        @echo "Copying classes (jars)..."
        @echo

        cp $(SRC_BSF)/bsf4oorexx/bsf-rexx-engine.jar   $(CLASSES)/
        cp $(SRC_BSF)/bsf4oorexx/bsf-v400-20090910.jar $(CLASSES)/

        @echo
        @echo "Copying executable files..."
        @echo
```

```
        cp $(SRC_BSF)/bsf4oorexx/BSF.CLS $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/BSF_OnTheFly.cls $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/rgf_util2.rex $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO.CLS $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_API_info.rxo $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_CREATE_INTERFACE_LIST.RXO
         ↳ $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_ReflectionUtil.cls $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_XINTERFACES.RXO $(COMMANDS)/

        @echo
        @echo "Copying OpenOffice libraries..."
        @echo

        cp $(SRC_BSF)/bsf4oorexx/install/ScriptProviderForooRexx.oxt
         ↳ $(LIBRARIES)/
        cp $(SRC_BSF)/bsf4oorexx/install/OOo_ScriptLibraries
         ↳ $(LIBRARIES)/

        @echo
        @echo "Creating symbolic links..."
        @echo

        cd $(DST)/Versions
        ln -s A $(DST)/Versions/Current
        cd ..
        ln -s Versions/Current/Classes   $(DST)/
        ln -s Versions/Current/Commands  $(DST)/
        ln -s Versions/Current/Libraries $(DST)/

build32: build

        @echo
        @echo "Copying 32 bit files..."
        @echo

        cp $(SRC_DYLIB)/libBSF4ooRexx_32.dylib $(LIBRARIES)/
         ↳ libBSF4ooRexx.dylib
        cp $(SRC_REXXJ2)/rexxj2_32.sh $(COMMANDS)/rexxj2.sh

        @echo
        @echo "Done."
        @echo


build64: build

        @echo
        @echo "Copying 64 bit files..."
        @echo

        cp $(SRC_DYLIB)/libBSF4ooRexx_64.dylib $(LIBRARIES)/
         ↳ libBSF4ooRexx.dylib
        cp $(SRC_REXXJ2)/rexxj2_64.sh $(COMMANDS)/rexxj2.sh

        @echo
        @echo "Done."
        @echo
```

Listing 6: The complete framework Makefile.

The Makefile has five different targets that can be invoked, namely `default`, `clean`, `build`, `build32` and `build64`.
The `clean` target simply erases the old version of the framework directory from the destination folder. `build` does the most work. It prepares the framework's directory structure, copies the required files to their places and adds the corresponding symbolic links for the current version. Even more important than `build` are `build32` and `build64`. These two targets copy the 32 or 64 bit versions of the BSF4ooRexx dynamic library and `rexxj2.sh` to the correct location. They depend on the `build` target, which means that `build` is executed before `build32` or `build64`. This creates the framework's complete structure as already shown in chapter 3.2.2 "Creating the BSF4ooRexx Framework".

By default, the target `default` is executed automatically, even if no explicit target for invocation has been defined. This is the default behaviour of `make`. If no target is given when calling `make`, the first target defined in the Makefile if used. It will display a short information text on how to use the Makefile.

Even if each target can be called individually, the targets shown in listing 7 and listing 8 are sufficient to build the framework in 32 or 64 bit mode on a MacOSX system.

```
make build32
```

Listing 7: Calling the Makefile's build32 target using make.

```
make build64
```

Listing 8: Calling the Makefile's build64 target using make.

Now that the framework has been created, it has to be configured. The necessary steps are described in the following chapters.

3.2.4 Setting Up the Framework

To set up the framework, a series of symbolic links (symlinks) has to be created and a new entry has to be added to the system's `CLASSPATH`. This is necessary for establishing the required dependencies and successfully executing ooRexx scripts using BSF4ooRexx.

The commands given in listing 9 will create the symbolic links to the required resources.

```
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Commands/BSF.CLS
 ↪ /usr/bin/
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Commands/
 ↪ BSF_OnTheFly.cls /usr/bin/
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Commands/
 ↪ rexxj2.sh /usr/bin/
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Commands/
 ↪ rgf_util2.rex /usr/bin/
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Libraries/
 ↪ libBSF4ooRexx.dylib /usr/lib/java/
ln -s /System/Library/Frameworks/BSF4ooRexx.framework/Libraries/
 ↪ libBSF4ooRexx.dylib /usr/lib/
```

Listing 9: Creating the symlinks.

Last but not least, the `.jar` files have to be added to the `CLASSPATH` environment variable. This setting will be taken in the `profile` file, which is stored in the `/etc` directory.
Therefore, the necessary lines shown in listing 10 have to be added to the `/etc/profile` file, so all users are able to use it.

```
export CLASSPATH=/System/Library/Frameworks/BSF4ooRexx.framework
 ↪ /Classes/bsf-rexx-engine.jar:/System/Library/Frameworks/
 ↪ BSF4ooRexx.framework/Classes/bsf-v400-20090910.jar:$CLASSPATH
```

Listing 10: Extending the CLASSPATH.

After these steps have been completed successfully, a BSF4ooRexx program can be executed as shown in listing 11.

```
rexxj2.sh HelloWorld.rxj
```

Listing 11: Starting a simple Hello World program using rexxj2.sh.

The shell script `rexxj2.sh` starts the so-called `RexxDispatcher`[5], which is a Java program that, on the other hand, starts ooRexx. ooRexx receives the script and executes it.

For the sake of completeness, it has to be mentioned, that it is also possible to use `rexx` instead of `rexxj2.sh` to run this script. But in the current version of BSF4ooRexx[6] the command in listing 12 does not work yet, as explained in chapter 2.2.2 ”Adapting the C++ Source Code”.

[5] The RexxDispatcher comes with BSF4ooRexx.
[6] As of February 16th, 2011.

```
rexx HelloWorld.rxj
```

Listing 12: Starting a simple Hello World program using rexx.

3.2.5 Integration in OpenOffice

One of the major advantages of BSF4ooRexx is its ability to extend OpenOffice to use ooRexx as a scripting language and for automation. To integrate ooRexx in OpenOffice the following steps are necessary:

1. Create necessary symbolic links,
2. create and adapt environment variables,
3. run `unopgk` to add the ooRexx extension to OpenOffice and
4. add ooRexx macros to OpenOffice.

At the time of writing this document, only a 32 bit version of OpenOffice for MacOSX existed. To run all of these steps successfully, ooRexx and Java also need to run in 32 bit mode. To change the Java version MacOSX uses, the program “Java Preferences” from the Utilities directory in the Applications folder has to be started. It allows to change to a specific Java version by drag-and-drop as shown in figure 5.

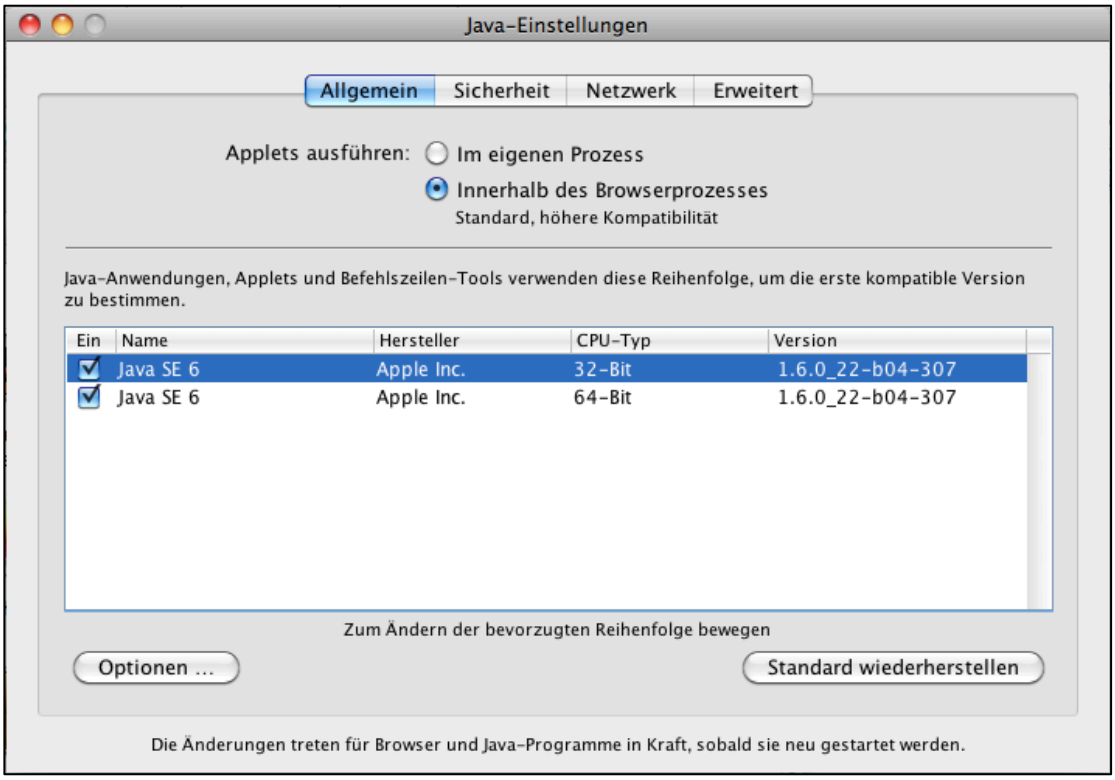

Figure 5: Changing the Java version.

Alternatively, the Java parameter `-d32` can be added to the `rexxh2.sh` shell script. Therefore, the according lines shown in listing 13 have to be changed to match the content of listing 14.

```
java org.rexxla.bsf.RexxDispatcher $*
...
java $BSF4Rexx_JavaStartupOptions org.rexxla.bsf.RexxDispatcher $*
```

Listing 13: The original rexxj2.sh.

```
java -d32 org.rexxla.bsf.RexxDispatcher $*
...
java -d32 $BSF4Rexx_JavaStartupOptions org.rexxla.bsf.RexxDispatcher
 ↪ $*
```

Listing 14: The changed rexxj2.sh that always runs Java in 32 bit mode.

3.2.5.1 Create and Adapt Environment Variables

To successfully run subsequent setup scripts it is necessary to set a couple of environment variables. This is done by adding the lines from listing 15 to `/etc/profile` as described in chapter 3.2.4 „Setting Up the Framework“. This will allow BSF4ooRexx to access OpenOffice.

```
export PATH=$PATH:/System/Library/Frameworks/BSF4ooRexx.framework/
↪ Commands
UNO_PATH=/Applications/OpenOffice.org.app/Contents/program
OOOJAVA=/Applications/OpenOffice.org.app/Contents/basis-link/
↪ ure-link/share/java
export CLASSPATH=$CLASSPATH:$UNO_PATH:$OOOJAVA/juh.jar:$OOOJAVA/
↪ jurt.jar:$OOOJAVA/ridl.jar
export CLASSPATH=$CLASSPATH:/Applications/OpenOffice.org.app/
↪ Contents/basis-link/program/classes/unoil.jar
```

Listing 15: Extending the CLASSPATH for OpenOffice.

3.2.5.2 Run “unopkg” to Add the ooRexx Extension to OpenOffice

The program `unopkg` allows a user to add extensions to OpenOffice from the command line [**15**]. It comes with OpenOffice and is normally located in `/Applications/OpenOffice.org/Contents/program`. Alternatively it would also be possible to add such an extension by using the built-in extension manager of OpenOffice which can be found in the “Extras” menu, as shown in figure 6.

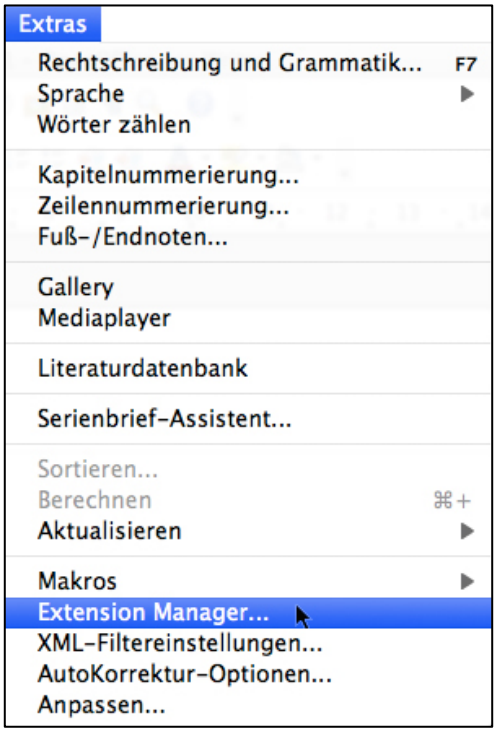

Figure 6: Open the OpenOffice extension manager.

Inside the extension manager, a new extension can be added by clicking the “Add extension” button and selecting the corresponding extension file, in our case a file called `ScriptProviderForooRexx.oxt`. When the extension has been added successfully, it is shown in the Extension Manager as depicted in figure 7.

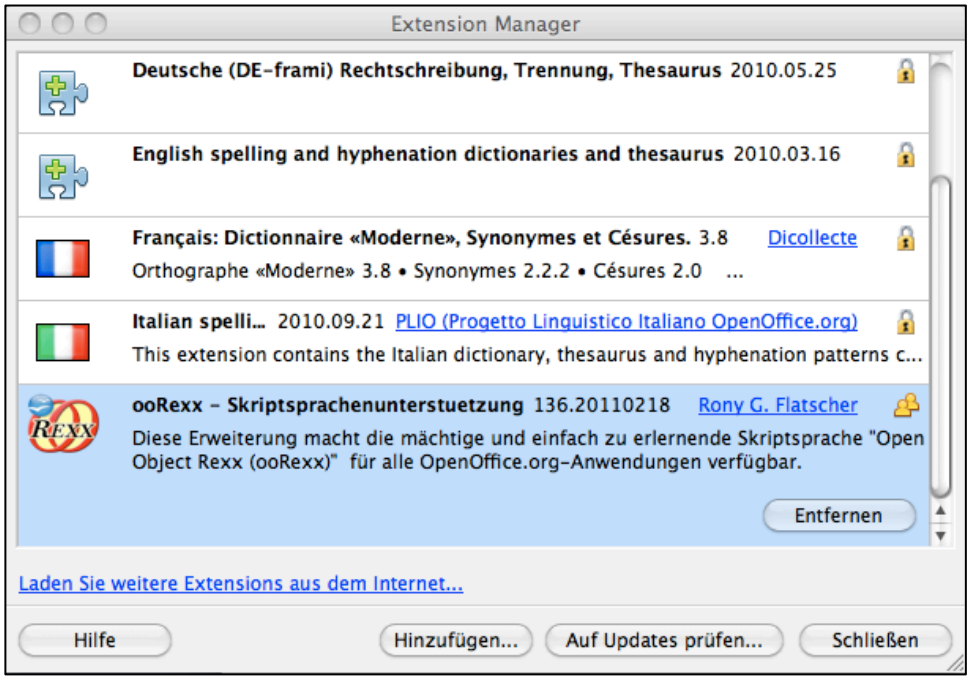

Figure 7: Add an extension to OpenOffice.

To add this extension by using the command line, the command shown in listing 16 has to be executed.

```
sudo unopkg add --shared ScriptProviderForooRexx.oxt
```

Listing 16: Adding the ooRexx extension to OpenOffice using unopkg.

3.2.5.3 Add ooRexx Macros to OpenOffice

The last step for setting up the OpenOffice integration is to add the ooRexx macros to OpenOffice.

To add the macros, the commands given in listing 17 have to be executed.

```
sudo cp -r /System/Library/Frameworks/BSF4ooRexx.framework/Libraries/
 ↳ OOo_ScriptLibraries/javascript/*
 ↳ /Applications/OpenOffice.org.app/Contents/basis-link/share/Scripts/
 ↳ javascript
sudo cp -r /System/Library/Frameworks/BSF4ooRexx.framework/Libraries/
 ↳ OOo_ScriptLibraries/python/*
 ↳ /Applications/OpenOffice.org.app/Contents/basis-link/share/Scripts/
 ↳ python
sudo cp -r /System/Library/Frameworks/BSF4ooRexx.framework/Libraries/
 ↳ OOo_ScriptLibraries/oorexx
 ↳ /Applications/OpenOffice.org.app/Contents/basis-link/share/Scripts/
sudo cp /System/Library/Frameworks/BSF4ooRexx.framework/Commands/
 ↳ UNO_API_info.rxo /Applications/OpenOffice.org.app/Contents/
 ↳ basis-link/share/Scripts/oorexx/wu_tools/
```

Listing 17: Adding ooRexx macros to OpenOffice.

After executing these commands and restarting OpenOffice the macros are available in the ooRexx macro window as depicted in figure 9.

After these steps have been completed successfully, a BSF4ooRexx program for OpenOffice can be executed, as shown in listing 18 and listing 19.

```
rexxj2.sh getVersion.rxo
```

Listing 18: Testing the OOo integration with rexxj2.sh.

For the sake of completeness, it has to be mentioned, that it is also possible to use `rexx` instead of `rexxj2.sh` to run this script. But in the current version of BSF4ooRexx[7] the command in listing 19 does not work yet, as explained in chapter 2.2.2 ”Adapting the C++ Source Code”.

```
rexx getVersion.rxo
```

Listing 19: Testing the OOo integration with rexx.

[7] As of February 16th, 2011.

Additionally, it is possible to run ooRexx scripts as macros from within OpenOffice. To do so, the user has to open the ooRexx macro window as shown in figure 8. In the Macro window the user can create, edit and delete the document's and the system wide macros, as show in figure 9.

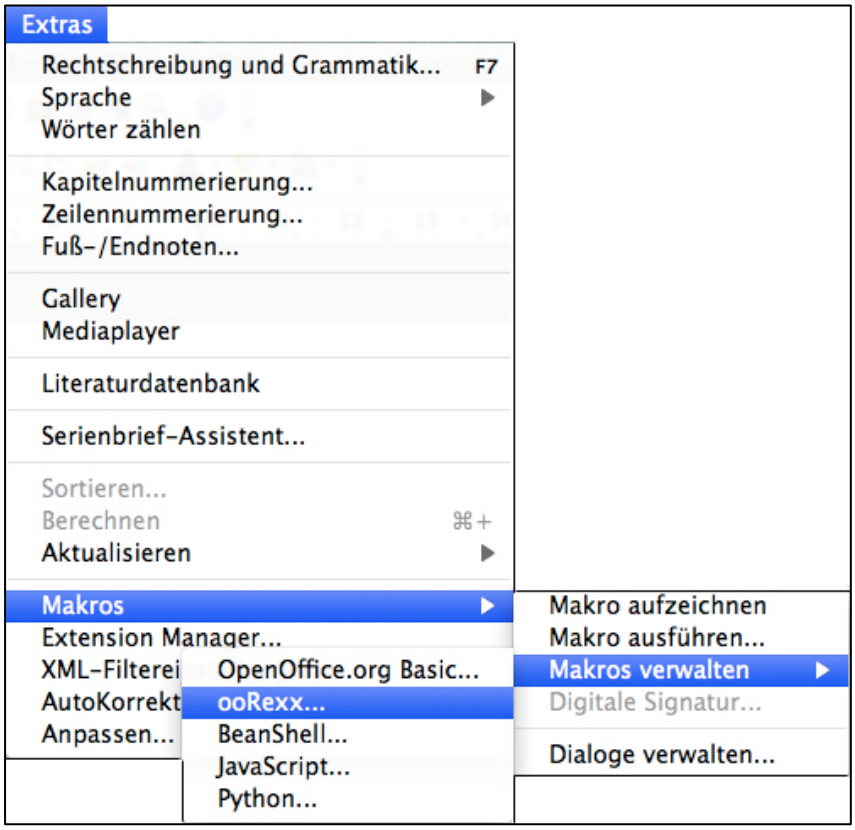

Figure 8: Open the ooRexx Macro window.

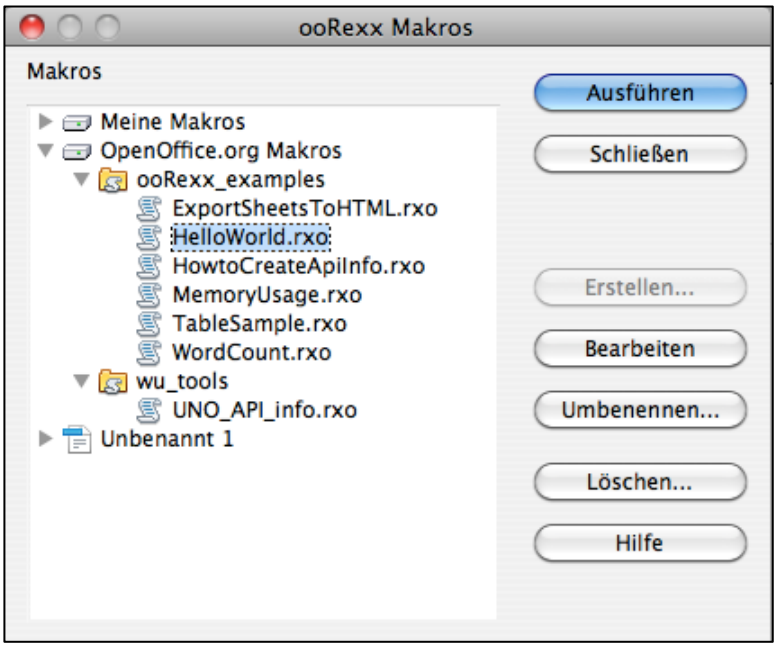

Figure 9: ooRexx Macro window.

3.3 Improving User Convenience

In modern operating systems, users are accustomed to simply execute scripts by double-clicking and to access their resources through attractive menu bars. To add this features to BSF4ooRexx on MacOSX the creation of an Application Bundle is necessary. This bundle will

- hold the necessary file associations for ooRexx and BSF4ooRexx scripts,
- handle the correct execution of such script files and will
- deliver an attractive menu bar to the user for easy access to (BSF4)ooRexx resources.

3.3.1 Creating the BSF4ooRexx Application Bundle

To achieve the improvements stated above, an Application Bundle[8] had to be created. The easiest way to create such a bundle is by using the program Automator. It is installed by default on every MacOSX system and is located in the Applications folder. As the name implies, it can be used to automate workflows on MacOSX computers. Automator comes with a number of templates for creating such workflows. In our case, we used the template “Program” to create a basic Application Bundle, as shown in figure 10.

Figure 10: Choosing the Automator template.

[8] Purpose and structure of such bundles have already been described in chapter 3.1.1 “Application Bundle”.

Actions are added to the program by dragging them from the sidebar on the left onto the application's main window. The action we need is called "Execute AppleScript". In the appearing editor window, the AppleScript, shown in listing 20 [**16**] [**17**], has been added. The result is shown in figure 11.

```
on run {input}
  set script_path to POSIX path of (path to me)
  if input is not {} then
    set the_path to POSIX path of input
    set cmd to "rexxj2.sh " & quoted form of the_path
    tell application "System Events" to set terminalIsRunning
    ↪ to exists application process "Terminal"
    tell application "Terminal"
      activate
      if terminalIsRunning is true then
        do script with command cmd
      else
        do script with command cmd in window 1
      end if
    end tell
  else
    tell application "System Events" to set terminalIsRunning to
    ↪ exists application process "Terminal"
    tell application "Terminal"
      activate
      if terminalIsRunning is true then
        do script with command "java -cp \"" & script_path
        ↪ & "Contents/MacOS\" BSF4ooRexx & exit;"
      else
        do script with command "java -cp \"" & script_path &
        ↪ "Contents/MacOS\" BSF4ooRexx & exit;" in window 1
      end if
    end tell
  end if
end run
```

Listing 20: AppleScript executed by the Application Stub.

The AppleScript will pass the double-clicked scripts to the ooRexx interpreter using `rexxj2.sh`. If no script file has been clicked and the Application Bundle has been invoked on its own, a menu bar, as shown in figure 13, will be loaded. Its complete source code is shown in chapter 5.2.5 in the Appendix.

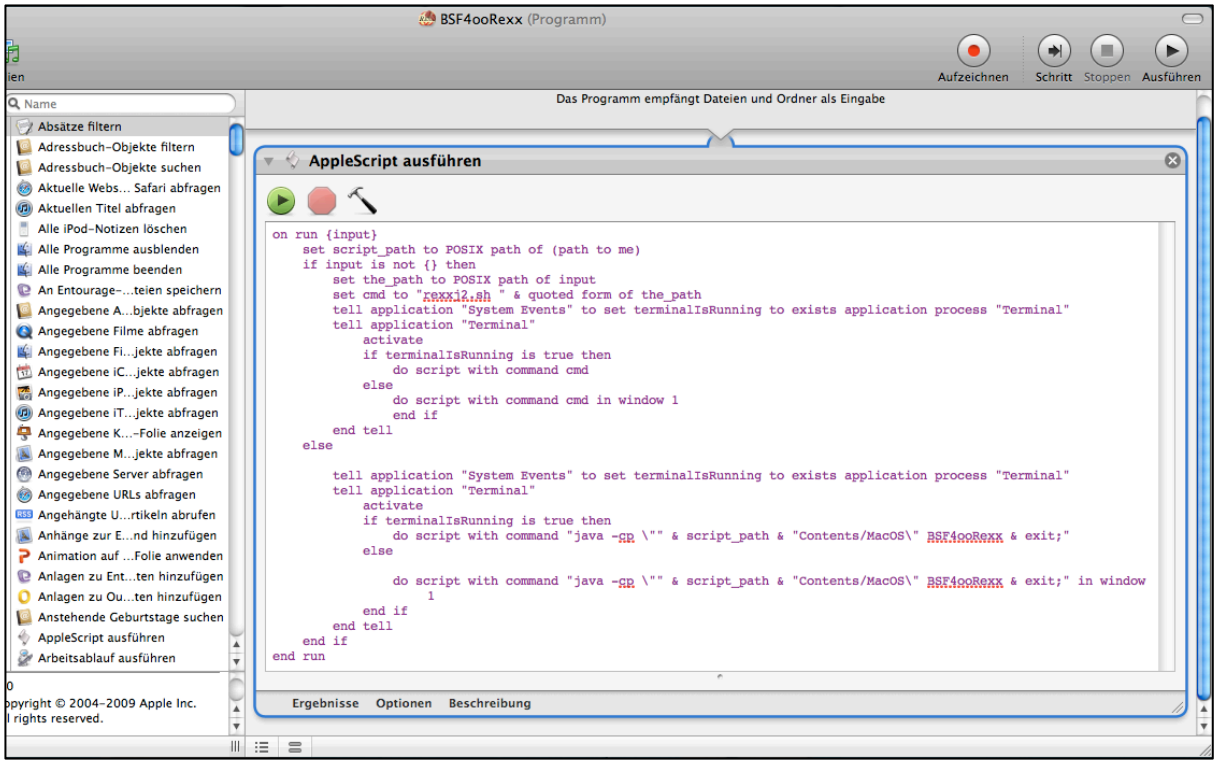

Figure 11: The Automator.

Finally, to create an Application Bundle, the user has to select “Save as...” from Automator’s menu bar and choose “Program” as the data format. After clicking on the OK button the Application Bundle is stored on the user’s hard disc.

The structure of the resulting Application Bundle, which was created, is shown in figure 12.

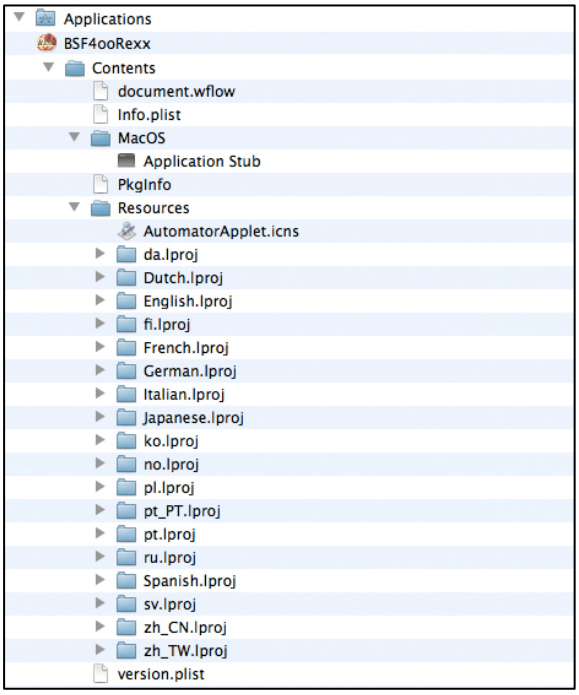

Figure 12: The Application Bundles generated by Automator.

The next step is to customize the newly created Application Bundle to fit our needs. Therefore, we changed the `Info.plist` file to match the contents of listing 21. It handles file associations and the execution of the BSF4ooRexx scripts. The code in the listing depicts a selection of the most important parts of the `Info.plist` file. The complete source code with all file type associations is shown in chapter 5.2.4 “Info.plist” in the Appendix.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>AMIsApplet</key>
        <true/>
        <key>AMStayOpen</key>
        <false/>
        <key>CFBundleDevelopmentRegion</key>
        <string>English</string>
        <key>CFBundleDocumentTypes</key>
        <array>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>rxj</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>
                                        text/x-rexx-java
                                </string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>RexxJavaScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>rex</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>text/x-rexx</string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>REXXScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>

                ...
```

```
        </array>
        <key>CFBundleExecutable</key>
        <string>Application Stub</string>
        <key>CFBundleIconFile</key>
        <string>bsf4oorexx.icns</string>
        <key>CFBundleIdentifier</key>
        <string>com.apple.automator.BSF4ooRexx</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundleName</key>
        <string>BSF4ooRexx</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>1.1</string>
        <key>CFBundleSignature</key>
        <string>????</string>
        <key>CFBundleVersion</key>
        <string>247.1</string>
        <key>LSMinimumSystemVersion</key>
        <string>10.5</string>
        <key>LSMinimumSystemVersionByArchitecture</key>
        <dict>
                <key>x86_64</key>
                <string>10.6</string>
                <key>i386</key>
                <string>10.4</string>
                <key>ppc</key>
                <string>10.4</string>
        </dict>
        <key>LSUIElement</key>
        <false/>
        <key>NSAppleScriptEnabled</key>
        <string>YES</string>
        <key>NSMainNibFile</key>
        <string>ApplicationStub</string>
        <key>NSPrincipalClass</key>
        <string>NSApplication</string>
</dict>
</plist>
```

Listing 21: Info.plist of the BSF4ooRexx Application Bundle.

The `Info.plist` must contain a tag called `CFBundleExecutable`. This tag defines the application to be started. In our case we don't use a full-featured executable, but an Application Stub, automatically created by Automator. This acts as a kind of placeholder and will pass the opened script files (if any present) to an AppleScript. If the user double-clicked a BSF4ooRexx script file, the AppleScript shown in listing 20 will pass the file to the ooRexx interpreter. On the other hand, if the user double-clicked the BSF4ooRexx Application Bundle itself, the built-in menu bar, as shown in figure 13, will be loaded.

Its complete source code is shown in chapter 5.2.5 in the Appendix.

Figure 13: The BSF4ooRexx menu bar.

The tag `CFBundleIconFile` contains the icon that is used for the Application Bundle itself [9], and `CFBundleName`, that defines the Bundle's name as displayed in the Finder. The latter is shown in figure 14.

Figure 14: BSF4ooRexx in the Application folder.

The XML tag `CFBundleDocumentTypes` is an array of different `CFBundleTypeExtensions`-Entries, which define the look and feel of multiple file type extensions handled by this application bundle. In our case, these extensions are `rxj`, `rxo`, `rex`, etc. The tag `CFBundleTypeName` defines the name of the file type.

Another tag, called `CFBundleTypeIconFile`, defines the file type icon. Files having the extensions defined above will get the assigned icon. To enable the application bundle to find the icon file, it must be placed in the bundle's `Resources` directory. MacOSX uses a special file format for its icons, called `.icns`. Such icons can be created using the program "Icon Composer" which comes with the XCode Development Environment[10] [**18**], [**19**].

It allows to convert a large number of existing picture formats into Apple's icon format. The conversion of the BSF4ooRexx logo into an `.icns` icon is shown in figure 15.

[9] This icon also has to be placed in the `Resources` directory.
[10] XCode comes with every Apple computer, but must be installed separately.

Figure 15: Converting an existing .png file into an .icns icon.

To complete the Application Bundle, the `samples` and the `utilities` folders have to be placed in the Contents directory. Last but not least, the compiled byte code of the menu bar (`BSF4ooRexx.class`) has to be put in the `MacOS` directory.

The final structure of the resulting Application Bundle is shown in figure 16.

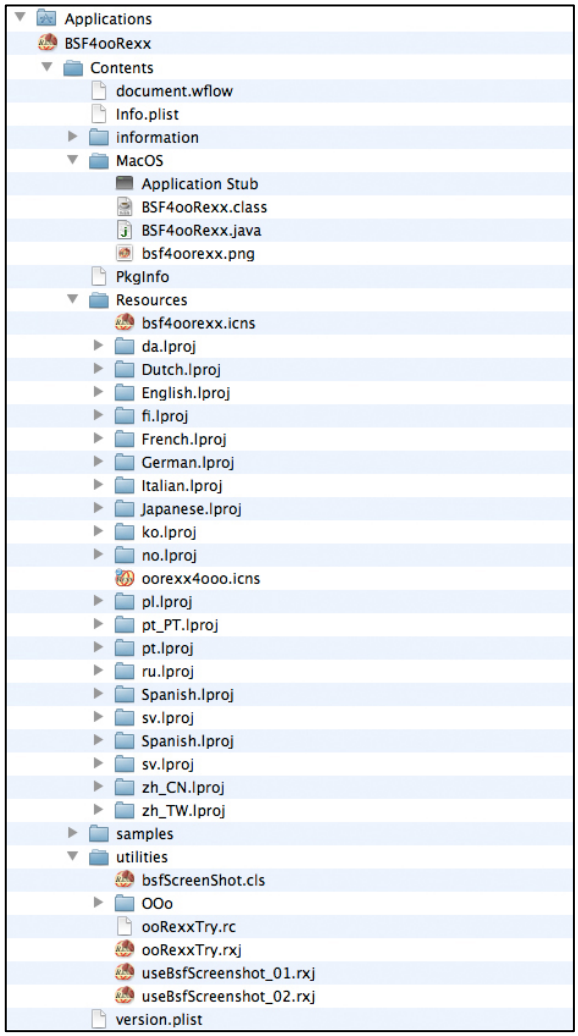

Figure 16: The BSF4ooRexx Application Bundle.

Once the Bundle has been placed in the Applications directory and it has been executed the first time, the file type associations will be set system wide for all users. Another way to force MacOSX to refresh all file association settings is to use the command shown in listing 22 [**20**].

```
/System/Library/Frameworks/CoreServices.framework/Versions/Current
↪ /Frameworks/LaunchServices.framework/Versions/Current/Support/
↪ lsregister -kill -r -domain system -domain local -domain
```

Listing 22: Refreshing the file associations by using lsregister.

Once the associations are active, the file type icons will be displayed appropriately in the Finder, as shown in figure 17.

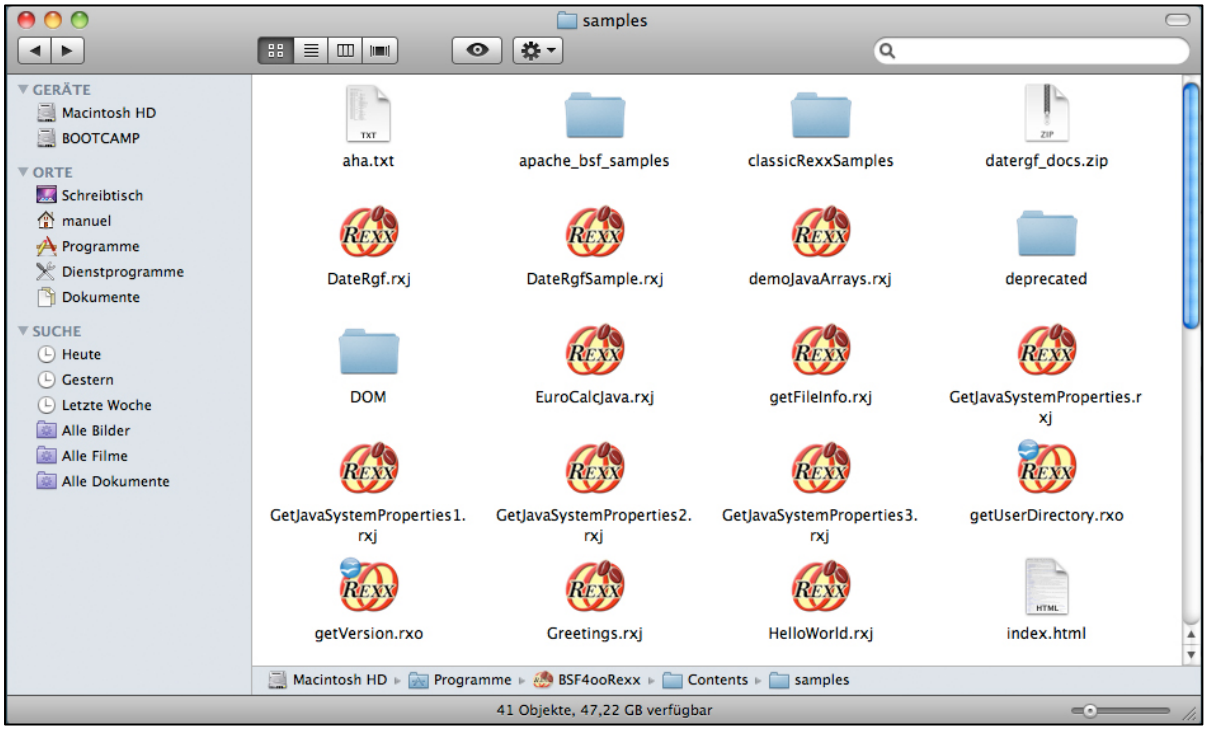

Figure 17: File type association.

For enhanced user convenience, the commands shown in chapters 3.2.4 “Setting Up the Framework” and 3.2.5 “Integration in OpenOffice” can be executed automatically by invoking the setup script shown in chapter 5.2.7 ”Setup Script for the BSF4ooRexx Application Bundle” in the Appendix.

3.3.2 Installing the BSF4ooRexx Application Bundle

The BSF4ooRexx Application Bundle follows the conventions of MacOSX. Therefore it can be installed by simply dragging it into the Application folder. Once it is put there, and executed the first time or the command in listing 22 is executed, the file associations will be set system wide.

The Application Bundle seamlessly integrates into the operating system and therefore it can be placed in the MacOSX Dock amongst other applications, as shown in figure 18.

Figure 18: BSF4ooRexx in the MacOSX Dock.

4 Roundup and Outlook

In summary it can be said that, besides some minor issues[11], this work was able to clear the way for finally using BSF4ooRexx on MacOSX.

Nevertheless, there are further improvements necessary on the sector of user convenience. Despite its high complexity, the user must be able to simply add and remove the BSF4ooRexx support, just like he is used to with other applications in MacOSX. Therefore, other projects will be necessary to create dedicated installer programs that handle the install, uninstall and reinstall processes. Especially the uninstall process is very important because all files, symbolic links and entries in the environment variables have to be cleanly removed, without negatively influencing other installed applications.

[11] As described in chapter 2.2.2 ”Adapting the C++ Source Code”.

5 Appendix

In the following chapters we have collected the source codes, which are necessary for a successful integration of BSF4ooRexx in MacOSX.

5.1 Used programs

To carry out the tasks described in this work, the programs explained in the following chapters have been used.

5.1.1 MacVim

MacVim [**21**] is a free, open source editor program with a large number of supported programming languages. During this work it is used for creating and editing Makefiles, the Info.plist, the Java source code and the shell scripts. Especially its feature for syntax highlighting makes it very useful.

5.1.2 XCode

Apple's XCode Development Environment [**13**] comes with everything necessary for compiling C++ source code. This ability makes it essential for the tasks described in chapter 2.2.1 and 2.2.3.

5.1.3 Automator

Automator [**22**] is installed by default on every MacOSX system. It can be used for automating workflows on Apple computers. In our case, it is used for creating a basic Application Bundle for BSF4ooRexx, as described in chapter 3.3.1 ”Creating the BSF4ooRexx Application Bundle”.

5.1.4 Icon Composer

The Icon Composer comes with XCode and can be used to create and convert MacOSX compatible icon files. Its usage is described in chapter 3.3.1 ”Creating the BSF4ooRexx Application Bundle”.

5.1.5 OpenOffice

OpenOffice [**23**] is free productivity suite for several operating systems, including Windows, Linux and MacOSX. Its functions can be automated using BSF4ooRexx.

5.2 Source Codes

The following chapters contain the source code of the files created in this work.

5.2.1 Makefile for BSF4ooRexx.dylib

The following Makefile is capable of creating either a 32 or 64 bit dynamic library for BSF4ooRexx.

```
# Name: Makefile
# Purpose: Makefile for BSF4ooRexx dynamic library (dylib)
# Author: Manuel Paar, Juergen Hesse
# Version: 1.0
#
# ------------------- Apache Version 2.0 license --------------------
#    Copyright (C) 2011 Manuel Paar, Juergen Hesse
#
#    Licensed under the Apache License, Version 2.0 (the "License");
#    you may not use this file except in compliance with the License.
#    You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing,
#    software distributed under the License is distributed on an "AS
#    IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
#    express or implied. See the License for the specific language
#    governing permissions and limitations under the License.
# -------------------------------------------------------------------

INC_PATH = -I. -I/System/Library/Frameworks/JavaVM.framework/headers

INC_PATH_ORX = -I/opt/ooRexx/include

i386: BSF4ooRexx.cc
    g++ -c -fPIC $(INC_PATH) -I$(INC_PATH_ORX) -m32
     ↪ -arch i386 -DUSE_OREXX -DUNIX -DBSF4REXX_32_BIT
     ↪ -oBSF4ooRexx-mac-i386.o BSF4ooRexx.cc

    g++ -dynamiclib -shared -o libBSF4ooRexx-i386.dylib
     ↪ BSF4ooRexx-mac-i386.o /usr/lib/librexx.dylib
     ↪ /usr/lib/librexxapi.dylib -framework JavaVM -arch i386

x86_64: BSF4ooRexx.cc
    g++ -c -fPIC $(INC_PATH) -I$(INC_PATH_ORX) -m64
     ↪ -arch x86_64 -DUSE_OREXX -DUNIX -DBSF4REXX_64_BIT
     ↪ -oBSF4ooRexx-mac-x86_64.o BSF4ooRexx.cc

    g++ -dynamiclib -shared -o libBSF4ooRexx-x86_64.dylib
     ↪ BSF4ooRexx-mac-x86_64.o /usr/lib/librexx.dylib
     ↪ /usr/lib/librexxapi.dylib -framework JavaVM -arch x86_64
```

To create a 32 bit dylib the following command has to be executed:

```
make --makefile=Makefile i386
```

To create a 64 bit library, on the other hand, to following command hast o be used:

```
make --makefile=Makefile x86_64
```

The usage of this Makefile is shown in chapter 2.2.1 ”Creating the Makefile”.

5.2.2 Makefile for BSF4ooRexx Framework

```
# Name: Makefile
# Purpose: Makefile for the BSF4ooRexx framework
# Author: Juergen Hesse, Manuel Paar
# Version: 1.0
#
# ------------------- Apache Version 2.0 license --------------------
#    Copyright (C) 2011 Manuel Paar, Juergen Hesse
#
#    Licensed under the Apache License, Version 2.0 (the "License");
#    you may not use this file except in compliance with the License.
#    You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing,
#    software distributed under the License is distributed on an "AS
#    IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
#    express or implied. See the License for the specific language
#    governing permissions and limitations under the License.
# -------------------------------------------------------------------

# Define source directories
SRC_BSF = BSF4ooRexx_install
SRC_DYLIB = BSF4ooRexx
SRC_REXXJ2 = BSF4ooRexx


# Define destination directories
DST = BSF4ooRexx.framework
VERSION_A = $(DST)/Versions/A
CURRENT = $(DST)/Versions/Current
CLASSES = $(VERSION_A)/Classes
COMMANDS = $(VERSION_A)/Commands
LIBRARIES = $(VERSION_A)/Libraries

default:
        @echo
        @echo "Usage:"
        @echo "make {clean|build32|build64}"
        @echo
        @echo "clean:"
        @echo "   Cleans the destination directory ($(DST))."
        @echo "build32:"
        @echo "   Builds the framework with 32 bit libraries."
        @echo "build64:"
        @echo "   Builds the framework with 64 bit libraries."
        @echo
        @echo

clean:
        @echo "-----------------------------------"
        @echo "Cleaning..."
        @echo "-----------------------------------"
        @echo
        @echo "Removing old directory structure..."
        @echo
        rm -rf $(DST)
        @echo
        @echo "Done."
        @echo
```

```
build:
        @echo "-----------------------------------"
        @echo "Building..."
        @echo "-----------------------------------"
        @echo
        @echo "Creating directory structure..."
        @echo

        mkdir -p $(CLASSES)
        mkdir -p $(COMMANDS)
        mkdir -p $(LIBRARIES)

        @echo
        @echo "Copying classes (jars)..."
        @echo

        cp $(SRC_BSF)/bsf4oorexx/bsf-rexx-engine.jar   $(CLASSES)/
        cp $(SRC_BSF)/bsf4oorexx/bsf-v400-20090910.jar $(CLASSES)/

        @echo
        @echo "Copying executable files..."
        @echo

        cp $(SRC_BSF)/bsf4oorexx/BSF.CLS $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/BSF_OnTheFly.cls $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/rgf_util2.rex $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO.CLS $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_API_info.rxo $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_CREATE_INTERFACE_LIST.RXO
         ↳ $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_ReflectionUtil.cls $(COMMANDS)/
        cp $(SRC_BSF)/bsf4oorexx/UNO_XINTERFACES.RXO $(COMMANDS)/

        @echo
        @echo "Copying OpenOffice libraries..."
        @echo

        cp $(SRC_BSF)/bsf4oorexx/install/ScriptProviderForooRexx.oxt
         ↳ $(LIBRARIES)/
        cp $(SRC_BSF)/bsf4oorexx/install/OOo_ScriptLibraries
         ↳ $(LIBRARIES)/

        @echo
        @echo "Creating symbolic links..."
        @echo

        cd $(DST)/Versions
        ln -s A $(DST)/Versions/Current
        cd ..
        ln -s Versions/Current/Classes   $(DST)/
        ln -s Versions/Current/Commands  $(DST)/
        ln -s Versions/Current/Libraries $(DST)/

build32: build

        @echo
        @echo "Copying 32 bit files..."
        @echo

        cp $(SRC_DYLIB)/libBSF4ooRexx_32.dylib $(LIBRARIES)/
         ↳ libBSF4ooRexx.dylib
```

```
        cp $(SRC_REXXJ2)/rexxj2_32.sh $(COMMANDS)/rexxj2.sh

        @echo
        @echo "Done."
        @echo


build64: build

        @echo
        @echo "Copying 64 bit files..."
        @echo

        cp $(SRC_DYLIB)/libBSF4ooRexx_64.dylib $(LIBRARIES)/
         ↳ libBSF4ooRexx.dylib
        cp $(SRC_REXXJ2)/rexxj2_64.sh $(COMMANDS)/rexxj2.sh

        @echo
        @echo "Done."
        @echo
```

This Makefile is described in further detail in chapter 3.2.3 “Automating the Framework Creation Using a Makefile”.

For further convenience, adapted versions of `rexxj2.sh` for 32 and 64 bit, as described in chapter 3.2.5 “Integration in OpenOffice”, have been used.

5.2.3 AppleScript

```
on run {input}
  set script_path to POSIX path of (path to me)
  if input is not {} then
    set the_path to POSIX path of input
    set cmd to "rexxj2.sh " & quoted form of the_path
    tell application "System Events" to set terminalIsRunning
    ↪ to exists application process "Terminal"
    tell application "Terminal"
      activate
      if terminalIsRunning is true then
        do script with command cmd
      else
        do script with command cmd in window 1
      end if
    end tell
  else
    tell application "System Events" to set terminalIsRunning to
    ↪ exists application process "Terminal"
    tell application "Terminal"
      activate
      if terminalIsRunning is true then
        do script with command "java -cp \"" & script_path
        ↪ & "Contents/MacOS\" BSF4ooRexx & exit;"
      else
        do script with command "java -cp \"" & script_path &
        ↪ "Contents/MacOS\" BSF4ooRexx & exit;" in window 1
      end if
    end tell
  end if
end run
```

This AppleScript is described in further detail in chapter 3.3.1 ”Creating the BSF4ooRexx Application Bundle”.

5.2.4 Info.plist

This is the `Info.plist` of the BSF4ooRexx application bundle. It is described in further detail in chapter 3.3.1 “Creating the BSF4ooRexx Application Bundle”.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
        <key>AMIsApplet</key>
        <true/>
        <key>AMStayOpen</key>
        <false/>
        <key>CFBundleDevelopmentRegion</key>
        <string>English</string>
        <key>CFBundleDocumentTypes</key>
        <array>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>rxj</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>
                                        text/x-rexx-java
                                </string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>RexxJavaScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>rex</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>text/x-rexx</string>
                        </array>
```

```
                        <key>CFBundleTypeName</key>
                        <string>REXXScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>JREXX</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>
                                        text/x-rexx-java
                                </string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>RexxJavaScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>ORX</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>text/x-rexx</string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>REXXScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>REXX</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>text/x-rexx</string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>REXXScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>REXXJ</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
```

```
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>
                                        text/x-rexx-java
                                </string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>RexxJavaScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>CLS</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>bsf4oorexx.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>text/x-rexx</string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>REXXScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
                <dict>
                        <key>CFBundleTypeExtensions</key>
                        <array>
                                <string>rxo</string>
                        </array>
                        <key>CFBundleTypeIconFile</key>
                        <string>oorexx4ooo.icns</string>
                        <key>CFBundleTypeMIMETypes</key>
                        <array>
                                <string>
                                        text/x-rexx-java-ooo
                                </string>
                        </array>
                        <key>CFBundleTypeName</key>
                        <string>RexxOOoScript</string>
                        <key>CFBundleTypeRole</key>
                        <string>Viewer</string>
                </dict>
        </array>
        <key>CFBundleExecutable</key>
        <string>Application Stub</string>
        <key>CFBundleIconFile</key>
        <string>bsf4oorexx.icns</string>
        <key>CFBundleIdentifier</key>
        <string>com.apple.automator.BSF4ooRexx</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundleName</key>
        <string>BSF4ooRexx</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>1.1</string>
        <key>CFBundleSignature</key>
        <string>????</string>
```

```
        <key>CFBundleVersion</key>
        <string>247.1</string>
        <key>LSMinimumSystemVersion</key>
        <string>10.5</string>
        <key>LSMinimumSystemVersionByArchitecture</key>
        <dict>
                <key>x86_64</key>
                <string>10.6</string>
                <key>i386</key>
                <string>10.4</string>
                <key>ppc</key>
                <string>10.4</string>
        </dict>
        <key>LSUIElement</key>
        <false/>
        <key>NSAppleScriptEnabled</key>
        <string>YES</string>
        <key>NSMainNibFile</key>
        <string>ApplicationStub</string>
        <key>NSPrincipalClass</key>
        <string>NSApplication</string>
</dict>
</plist>
```

5.2.5 Menu Bar

The following source displays the menu bar, which is shown in figure 13. It is written in Java. For the typical MacOSX look and feel, it is necessary to write the commands from listing 23 in the main method [**21**].

```
System.setProperty("apple.laf.useScreenMenuBar", "true");
System.setProperty("com.apple.mrj.application.apple.menu.
 ↪ about.name", "BSF4ooRexx Menu Bar");
```

Listing 23: Adding MacOSX look and feel.

The dock icon will be set with the command shown in listing 24 [**21**].

```
com.apple.eawt.Application.getApplication()
 ↪.setDockIconImage(ImageIO.read(this.getClass()
 ↪.getClassLoader().getResource("bsf4oorexx.png")));
```

Listing 24: Defining the dock icon.

In Java, the Apple Command-Key, which is used for shortcuts, is handled with the `Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()` method [**22**].

```
/* Name: BSF4ooRexx.java
   Purpose: Menu bar for BSF4ooRexx
   Author: Manuel Paar, Juergen Hesse
   Version: 1.0

   ------------------- Apache Version 2.0 license --------------------
      Copyright (C) 2011 Manuel Paar, Juergen Hesse

      Licensed under the Apache License, Version 2.0 (the "License");
      you may not use this file except in compliance with the License.
      You may obtain a copy of the License at

          http://www.apache.org/licenses/LICENSE-2.0

      Unless required by applicable law or agreed to in writing,
      software distributed under the License is distributed on an "AS
      IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
      express or implied. See the License for the specific language
      governing permissions and limitations under the License.
   ---------------------------------------------------------------------
*/

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.imageio.*;

public class BSF4ooRexx implements Runnable, ActionListener
{
  private JFrame frame;
  private JMenuBar menuBar;
  private JMenu infoMenu;
  private JMenu installMenu;
  private JMenu samplesMenu;
  private JMenu utilMenu;
  private JMenuItem infoMenuItem;
  private JMenuItem downloadMenuItem;
  private JMenuItem uninstallMenuItem;
  private JMenuItem reinstallMenuItem;
  private JMenuItem samplesMenuItem;
  private JMenuItem utilMenuItem;
  private JMenuItem ooRexxTryMenuItem;

  public static void main(String[] args)
  {
    System.setProperty("apple.laf.useScreenMenuBar", "true");
    System.setProperty("com.apple.mrj.application.apple.menu.
    ↪about.name", "BSF4ooRexx Menu Bar");
    SwingUtilities.invokeLater(new BSF4ooRexx());
  }

  public void run()
  {
    try {
      com.apple.eawt.Application.getApplication().
      setDockIconImage(
         ImageIO.read(this.getClass().getClassLoader().
         ↪getResource("bsf4oorexx.png")));
    } catch (java.io.IOException e) {
        System.out.println(e.getMessage());
    }
```

```
    frame = new JFrame("BSF4ooRexx");
    menuBar = new JMenuBar();

    infoMenu = new JMenu("Information");
    infoMenuItem = new JMenuItem("ReleaseNotes,
    ↳ introductions to ooRexx, BSF4ooRexx overview");
    infoMenuItem.setAccelerator(KeyStroke.getKeyStroke('I',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    infoMenuItem.addActionListener(this);
    infoMenu.add(infoMenuItem);

    installMenu = new JMenu("Installation");
    downloadMenuItem = new JMenuItem("BSF4ooRexx Download Page");
    downloadMenuItem.setAccelerator(KeyStroke.getKeyStroke('D',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    downloadMenuItem.addActionListener(this);
    uninstallMenuItem = new JMenuItem("Uninstall BSF4ooRexx");
    uninstallMenuItem.setAccelerator(KeyStroke.getKeyStroke('U',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    uninstallMenuItem.addActionListener(this);
    reinstallMenuItem = new JMenuItem("Reinstall BSF4ooRexx");
    reinstallMenuItem.setAccelerator(KeyStroke.getKeyStroke('R',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    reinstallMenuItem.addActionListener(this);
    installMenu.add(downloadMenuItem);
    installMenu.add(uninstallMenuItem);
    installMenu.add(reinstallMenuItem);

    samplesMenu = new JMenu("Samples");
    samplesMenuItem = new JMenuItem("Simple to comprehensive
    ↳ BSF4ooRexx examples");
    samplesMenuItem.setAccelerator(KeyStroke.getKeyStroke('S',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    samplesMenuItem.addActionListener(this);
    samplesMenu.add(samplesMenuItem);

    utilMenu = new JMenu("Utilities");
    utilMenuItem = new JMenuItem("Some platform-independent utility
    ↳ scripts");
    utilMenuItem.setAccelerator(KeyStroke.getKeyStroke('U',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    utilMenuItem.addActionListener(this);
    ooRexxTryMenuItem = new JMenuItem("ooRexxTry");
    ooRexxTryMenuItem.setAccelerator(KeyStroke.getKeyStroke('T',
    ↳ Toolkit.getDefaultToolkit().getMenuShortcutKeyMask()));
    ooRexxTryMenuItem.addActionListener(this);
    utilMenu.add(utilMenuItem);
    utilMenu.add(ooRexxTryMenuItem);

    menuBar.add(infoMenu);
    menuBar.add(installMenu);
    menuBar.add(samplesMenu);
    menuBar.add(utilMenu);

    frame.setJMenuBar(menuBar);

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(0, 0);
    frame.setLocation(Short.MIN_VALUE, Short.MIN_VALUE);
    frame.setUndecorated(true);
    frame.setVisible(true);
  }
```

```
  public void actionPerformed(ActionEvent ev)
  {
    Object source = ev.getSource();

    try {
        if (source == infoMenuItem)
                Runtime.getRuntime().exec("open
                ↳ /Applications/BSF4ooRexx.app/Contents/
                ↳ information/");
        else if (source == downloadMenuItem)
                Runtime.getRuntime().exec("open
                ↳ http://wi.wu.ac.at/rgf/rexx/bsf4oorexx/");
        else if (source == uninstallMenuItem)
                JOptionPane.showMessageDialog(null, "Coming soon!",
                "BSF4ooRexx Uninstall", JOptionPane.OK_CANCEL_OPTION);
        else if (source == reinstallMenuItem)
                JOptionPane.showMessageDialog(null, "Coming soon!",
                "BSF4ooRexx Reinstall", JOptionPane.OK_CANCEL_OPTION);
        else if (source == samplesMenuItem)
                Runtime.getRuntime().exec("open
                ↳ /Applications/BSF4ooRexx.app/Contents/samples/");
        else if (source == utilMenuItem)
                Runtime.getRuntime().exec("open
                ↳ /Applications/BSF4ooRexx.app/Contents/utilities/");
        else if (source == ooRexxTryMenuItem)
                Runtime.getRuntime().exec("rexxj2.sh
                ↳ /Applications/BSF4ooRexx.app/Contents/
                ↳ utilities/ooRexxTry.rxj");

    } catch (java.io.IOException e) {
        System.out.println(e.getMessage());
    }
  }
}
```

5.2.6 /etc/profile

The following `/etc/profile` file is the result of the tasks carried out in the chapters 3.2.4 ”Setting Up the Framework” and 3.2.5.1 ”Create and Adapt Environment Variables”.

```
# System-wide .profile for sh(1)

if [ -x /usr/libexec/path_helper ]; then
        eval `/usr/libexec/path_helper -s`
fi

if [ "${BASH-no}" != "no" ]; then
        [ -r /etc/bashrc ] && . /etc/bashrc
fi

export PATH=$PATH:/System/Library/Frameworks/BSF4ooRexx.framework
↳ /Commands
export CLASSPATH=$CLASSPATH:/Applications/OpenOffice.org.app/Contents
↳ /program:/Applications/OpenOffice.org.app/Contents/basis-link
↳ /ure-link/share/java/juh.jar:/Applications/OpenOffice.org.app
↳ /Contents/basis-link/ure-link/share/java/jurt.jar:/Applications
↳ /OpenOffice.org.app/Contents/basis-link/ure-link/share/java/ridl.jar
export CLASSPATH=$CLASSPATH:/Applications/OpenOffice.org.app/Contents
↳ /basis-link/program/classes/unoil.jar
export CLASSPATH=$CLASSPATH:/System/Library/Frameworks
↳ /BSF4ooRexx.framework/Classes/bsf-rexx-engine.jar
export CLASSPATH=$CLASSPATH:/System/Library/Frameworks
↳ /BSF4ooRexx.framework/Classes/bsf-v400-20090910.jar
```

5.2.7 Setup Script for the BSF4ooRexx Application Bundle

```
#!/bin/sh
# Name: setup_BSF4ooRexx.sh
# Purpose: Setup script for BSF4ooRexx
# Author: Juergen Hesse, Manuel Paar
# Version: 1.0
#
# ------------------- Apache Version 2.0 license --------------------
#    Copyright (C) 2011 Juergen Hesse, Manuel Paar
#
#    Licensed under the Apache License, Version 2.0 (the "License");
#    you may not use this file except in compliance with the License.
#    You may obtain a copy of the License at
#
#        http://www.apache.org/licenses/LICENSE-2.0
#
#    Unless required by applicable law or agreed to in writing,
#    software distributed under the License is distributed on an "AS
#    IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either
#    express or implied. See the License for the specific language
#    governing permissions and limitations under the License.
# -------------------------------------------------------------------

FRAMEWORK_SYS_DIR=/System/Library/Frameworks
FRAMEWORK=BSF4ooRexx.framework
BSF4OOREXX_FRAMEWORK=$FRAMEWORK_SYS_DIR/$FRAMEWORK

BIN_DIR=/usr/bin
JAVA_LIB_DIR=/usr/lib/java

OOO_APP_DIR=/Applications/OpenOffice.org.app
OOO_UNO_PATH=$OOO_APP_DIR/Contents/program
OOO_JAVA_PATH=$OOO_APP_DIR/Contents/basis-link/ure-link/share/java
OOO_UNOIL_PATH=$OOO_APP_DIR/Contents/basis-link/program/classes

echo ""
echo "Copying framework data to $FRAMEWORK_SYS_DIR..."
echo ""

sudo cp -Rv $FRAMEWORK $FRAMEWORK_SYS_DIR/
chmod -R a+rX $BSF4OOREXX_FRAMEWORK

echo ""
echo "Creating symbolic links in $BIN_DIR..."
echo ""

sudo ln -s $BSF4OOREXX_FRAMEWORK/Commands/BSF.CLS          $BIN_DIR
sudo ln -s $BSF4OOREXX_FRAMEWORK/Commands/BSF_OnTheFly.cls $BIN_DIR
sudo ln -s $BSF4OOREXX_FRAMEWORK/Commands/rexxj2.sh        $BIN_DIR
sudo ln -s $BSF4OOREXX_FRAMEWORK/Commands/rgf_util2.rex    $BIN_DIR

echo ""
echo "Creating symbolic links in $JAVA_LIB_DIR..."
echo ""

sudo ln -s $BSF4OOREXX_FRAMEWORK/Libraries/libBSF4ooRexx.dylib
 ↳ $JAVA_LIB_DIR

echo ""
echo "Adding framework's Commands directory to PATH..."
echo ""
```

```
sudo echo "" >> /etc/profile
sudo echo "export PATH=\$PATH:$BSF4OOREXX_FRAMEWORK/Commands" >>
 ↳ /etc/profile

if [ -e $OOO_APP_DIR ]
    then
        echo ""
        echo "Adding OpenOffice JARs to CLASSPATH..."
        echo ""

        OOO_JUH=$OOO_JAVA_PATH/juh.jar
        OOO_JURT=$OOO_JAVA_PATH/jurt.jar
        OOO_RIDL=$OOO_JAVA_PATH/ridl.jar

        sudo echo "export CLASSPATH=\$CLASSPATH:$OOO_UNO_PATH:
         ↳ $OOO_JUH:$OOO_JURT:$OOO_RIDL" >> /etc/profile
        sudo echo "export CLASSPATH=\$CLASSPATH:
         ↳ $OOO_UNOIL_PATH/unoil.jar" >> /etc/profile

        sudo $OOO_UNO_PATH/unopkg add --shared $BSF4OOREXX_FRAMEWORK
         ↳ /Libraries/ScriptProviderForooRexx.oxt
fi

echo ""
echo "Adding BSF4ooRexx JARs to CLASSPATH..."
echo ""

sudo echo "export CLASSPATH=\$CLASSPATH:$BSF4OOREXX_FRAMEWORK
 ↳ /Classes/bsf-rexx-engine.jar" >> /etc/profile
sudo echo "export CLASSPATH=\$CLASSPATH:$BSF4OOREXX_FRAMEWORK
 ↳ /Classes/bsf-v400-20090910.jar" >> /etc/profile
```

This shell script creates all the necessary symbolic links for BSF4ooRexx, as described in chapters 3.2.4 “Setting Up the Framework” and 3.2.5 “Integration in OpenOffice”. At the moment of writing this document, this has to be executed manually. In the future, it is planned to create an appropriate installer program that executes this step automatically.

6 List of Figures

7 List of Listings

8 Bibliography

[1] Anonymous. (2011, January) ooRexx Homepage. [Online]. http://www.oorexx.org/charter.html

[2] Anonymous. (2011, January) ooRexx Homepage. [Online]. http://www.oorexx.org/about.html

[3] Rony G. Flatscher, "The 2009 Edition of BSF4Rexx," Business Informatics, Vienna University for Economics and Business Administration, Vienna, 2009.

[4] Anonymous. (2011, Feb.) w3counter.com. [Online]. http://w3counter.com/globalstats.php

[5] Anonymous. (2011, January) Wikipedia (Mac_OS_X). [Online]. http://www.wikipedia.org/wiki/Mac_OS_X

[6] Knut Lorenzen, *Mac OS X 10.6 Snow Leopard*.: mitp Verlag, 2009.

[8] Anonymous. (2011, January) developer.apple.com (Java). [Online]. http://developer.apple.com/library/mac/#releasenotes/Java/JavaSnowLeopardUpdate3LeopardUpdate8RN/NewandNoteworthy/NewandNoteworthy.html

[7] Anonymous. (2011, January) developer.apple.com (Java Overview). [Online]. http://developer.apple.com/library/mac/#documentation/Java/Conceptual/Java14Development/01-JavaOverview/JavaOverview.html

[9] Alexander Neumann. (2010, November) heise.de. [Online]. http://www.heise.de/newsticker/meldung/Oracle-und-Apple-machen-bei-OpenJDK-fuer-Mac-OS-X-gemeinsame-Sache-1135729.html

[10] Anonymous. (2011, Feb.) developer.apple.com (CFRunLoop). [Online]. http://developer.apple.com/library/mac/#documentation/CoreFoundation/Reference/CFRunLoopRef/Reference/reference.html

[11] Amit Singh. (2003, December) osxbook.com. [Online]. http://osxbook.com/book/bonus/ancient/whatismacosx//programming.html

[12] Anonymous. (2011, January) developer.apple.com (Frameworks). [Online]. http://developer.apple.com/library/mac/#documentation/MacOSX/Conceptual/BPFrameworks/Concepts/WhatAreFrameworks.html%23//apple_ref/doc/uid/20002303-BBCEIJFI

[13] Apple Inc. (2011, Feb.) XCode Developer Tools Technology Overview. [Online]. http://developer.apple.com/technologies/tools/xcode.html

[14] Rony G. Flatscher. (2011, January) BSF4ooRexx Homepage. [Online]. http://wi.wu-wien.ac.at:8002/rgf/rexx/bsf4rexx/current/

[15] Anonymous. (2011, January) openoffice.org (unopkg). [Online]. http://wiki.services.openoffice.org/wiki/Documentation/DevGuide/Extensions/unopkg

[16] Bert Altenburg. (2011, Feb.) www.fischer-bayern.de. [Online]. http://www.fischer-bayern.de/as/as4as/AS4AS_g.pdf

[18] Anonymous. (2011, Feb.) developer.apple.com (CoreFoundationKeys). [Online]. http://developer.apple.com/library/ios/#documentation/general/Reference/InfoPlistKeyReference/Articles/CoreFoundationKeys.html

[17] Anonymous. (2011, Feb.) highball.se. [Online]. http://highball.se/2011/01/how-to-open-vim-in-terminal-when-double-click-on-a-file/

[19] Anonymous. (2011, Feb.) developer.apple.com (LaunchServicesKeys). [Online]. http://developer.apple.com/library/ios/#documentation/general/Reference/InfoPlistKeyReference/Articles/LaunchServicesKeys.html

[20] Anonymous. (2011, Feb.) ss64.com. [Online]. http://ss64.com/osx/lsregister.html

[21] Miles Dennis. (2011, Feb.) milesdennis.com. [Online]. http://www.milesdennis.com/2010/05/mac-menus-for-java-applications.html

[22] Anonymous. (2011, Feb.) devdaily.com. [Online]. http://www.devdaily.com/blog/post/jfc-swing/how-program-apple-command-key-keystroke-java-swing-mac-osx

[23] Anonymous. (2011, January) rixstep.com. [Online]. http://rixstep.com/2/20060901,00.shtml

9 License