"The 2009 Edition of BSF4Rexx"
Part 2, Introduction to BSF4Rexx

2009 International Rexx Symposium
Chilworth, England (May 2009)

Rony G. Flatscher (Rony.Flatscher@wu.ac.at)
Wirtschaftsuniversität Wien, Austria (http://www.wu.ac.at)
Agenda

- Open Issues
- Architecture
- New OO-APIs in ooRexx 4.0
- New features
- Roundup and Outlook
BSF4Rexx

http://wi.wu-wien.ac.at/rgf/rexx/bsf4rexx/current

• BSF with a Rexx engine
  – Allows the usage of Rexx from BSF
    • Any Java program can invoke Rexx
    • Rexx scripts are able to communicate with Java objects, if made available by the Java program
  – Allows Java to be used as a huge Rexx function library
    • The public methods and public fields of every Java object and Java class object can be used by Rexx
    • If necessary, Java can be started up by Rexx
BSF4Rexx
Architecture

Java Program

BSF (Java)

RexxEngine

BSF Registry

RexxAndJava

JNI

BSF4Rexx

(C++)

BSF()
BsfDropFuncs()
BsfInvokedBy()
BsfLoadFuncs()
BsfLoadJava()
BsfQueryAllFunctions()
BsfQueryRegisteredFunctions()
BsfUnloadJava()
BsfVersion()
Entering ooRexx

- **BSF.CLS**
  - An ooRexx module containing
    - Supporting BSF via the proxy class **BSF**
    - Supporting BSF routines, e.g. `bsf.import(...)`
    - Services like making the most important and pre-registered Java classes directly available via the environment symbol `.bsf4rexx`
  - Will load Java transparently, if not yet loaded
    - Rexx programs
BSF4Rexx with **BSF.CLS**

Architecture

**Java Program**

- BSF (Java)
  - BSF Registry
  - RexxEngine
  - RexxAndJava

**JNI**

- BSF4Rexx (C++)
  - BSF()
  - BsfDropFuncs()
  - BsfInvokedBy()
  - BsfLoadFuncs()
  - BsfLoadJava()
  - BsfQueryAllFunctions()
  - BsfQueryRegisteredFunctions()
  - BsfUnloadJava()
  - BsfVersion()

**ooRexx environment (e.g. "bsf4rexx")**

- BSF.cls
  - ooRexx scripts
Open Issues = Input for BSF4Rexx 4.0

• Real-time handling of events
  – E.g. no canceling possible

• Creating Java proxy objects for Java interfaces
  – E.g. Java Filter interface class
    • At the moment one needs to create a Java class which implements the Java interface and control that from ooRexx

• Creating ooRexx proxy objects to which Java methods can be forward to
  – implementing Java methods in ooRexx
New ooRexx 4.0 API, 1

- RexxStart() was only means of executing
- New way of executing Rexx code
  - RexxCreateInterpreter()
    - Using same interpreter instance multiple times
      - LoadPackage()
      - LoadPackageFromData()
      - CallProgram()
  - Terminate()
New ooRexxx 4.0 API, 2

New datatypes
- Bitness independent
- Most important ooRexx types

- API modelled after JNI
  - RexxInstance
    - Attach(), Detach()
  - ThreadContext
  - MethodContext
  - CallContext/ExitContext
• *RexxProxy* (a Java class)
  – Represents an ooRexx object at the Java side
  – Allows for sending messages from Java to ooRexx
  – Represents a `java.lang.reflect.InvocationHandler`
  – Allows for creating a `java.lang.reflect.Proxy`
    • Arbitrary list of interfaces
      – String, interface classes, or an object's interfaces
  – Allows for submitting a slot (any Rexx object, named "userData")
• New BsfCreateRexxProxy()
  – Argument(s)
    • Rexx object to be proxied
    • Optional slot "userData" (a Rexx object)
      – Will be part of the sendback directory, always added as the last argument
    • Optional list of Java interface classes
      – Either Java interface class object, name of a Java interface class or a Java object
  – Returns either a RexxProxy or a Proxy reference
    • Can be used as argument values to Java method invocations
• Callbacks via *RexxProxy*
  – Java method `invoke(methodName, argArray)`
  – One slot argument appended
    • Always the last argument, type `.Directory`
    • Entry "USERDATA"
    • Entry "METHODNAME"
  – Arguments turned into BSF-proxies, if possible
    • Problem: getting at routine `BSF.WRAP (BSF.CLS)`
  – Return value
    • If Rexx object, then another RexxProxy gets created and returned
• Return values from Rexx
  – If string object, string value is returned
  – If BSF or UNO proxy, beanName is returned
    • Will be looked up in the BSFRegistry and associated Java object will be used
  – If a Rexx object, then a *RexxProxy* gets created and returned to Java
• Errors from Java
  – New APIs allow for creating Rexx syntax conditions carrying the Java error messages
  – Not possible to supply Java Throwable for syntax conditions

• Errors from Rexx
  – Throwing a Java RexxException
  – Carrying as message the Rexx error message
  – Supplies Rexx condition object as a RexxProxy
Roundup, Current State

- Alpha
- Needs a *lot* of testing
- Need for finding out about environment, context available at callback time
  - What about routines, classes that got required and the object's method code rely upon?
- Goal
  - Make it 100% compatible with previous version of BSF4Rexx