Institut für Betriebswirtschaftslehre und Wirtschaftsinformatik

Abteilung für Wirtschaftsinformatik

Adding JSR-223 to BSF4ooRexx

The 2016 International Rexx Symposium

Rony G. Flatscher

Wirtschaftsuniversität Wien ■ Welthandelsplatz 1 ■ A-1020 Wien

© Prof. Dr. Rony G. Flatscher

Agenda

- Java scripting framework, a.k.a. JSR-223
  - Features
- The ooRexx JSR-223 scripting engine
  - Features
- Examples demonstrate
  - Java executing Rexx scripts
  - Rexx scripts fetching arguments from Java
- Roundup

© Rony G. Flatscher

Java Specification Request (JSR) – 223, 1

- Java Scripting Framework, package javax.script
  - **J**ava **S**pecification **R**equest group # 223, introduced with 1.6
  - Javadocs, as of 2016-08-16, cf.
    - <https://docs.oracle.com/javase/8/docs/api/javax/script/package-frame.html>
  - Java class javax.script.ScriptEngineManager
    - Lists available scripting engines (javax.script.ScriptEngineFactory)
      - Exploiting the Java service provider interface (SPI)
    - Loads a specific scripting engine by name, file extension or mime type
    - Allows interaction with the *global scope* (shared by all scripting engines) javax.script.Bindings (key/value pairs)

 © Rony G. Flatscher

Java Specification Request (JSR) – 223, 2

- Java Scripting Framework
  - Java class javax.script.SimpleScriptContext
    - Allows to fetch *engine* and *global scope* javax.script.Bindings
    - Supplies objects representing stdin (java.io.Reader), stdout (java.io.Writer) and stderr (java.io.Writer)
  - Java class javax.script.SimpleBindings
    - A java.util.Map backed key/value collection
    - Some key names with special meanings are defined as constants in javax.script.ScriptEngine:
      - ARGV, ENGINE, ENGINE_VERSION, FILENAME, LANGUAGE, LANGUAGE_VERSION, NAME

 © Rony G. Flatscher

Java Specification Request (JSR) – 223, 3

- Java Scripting Framework
  - Java class implementing javax.script.ScriptEngine
    - Represents a scripting engine, e.g. Javascript/Rhino/ECMAscript
    - Allows evaluating (executing) scripting code
    - Allows supplying Java objects via one of the javax.script.Bindings managed by the script engine's javax.script.ScriptContext
  - Optional Java interface javax.script.Invocable
    - Defines script function and script method invocations from Java
  - Optional Java interface javax.script.Compile
    - Allows to compile a script and reuse it

© Rony G. Flatscher

The JSR–223 for ooRexx, 1

- Package org.rexxla.bsf.engines.rexx.jsr223
  - Class RexxScriptEngineFactory
    - Extends javax.script.AbstractScriptEngine and implements the
      - Interface javax.script.ScriptEngineFactory
  - Class RexxScriptEngine
    - Extends javax.script.AbstractScriptEngine and implements the
      - Interface javax.script.ScriptEngine
      - Interface javax.script.Compilable
      - Interface javax.script.Invocable

 © Rony G. Flatscher

The JSR–223 for ooRexx, 2

- Package org.rexxla.bsf.engines.rexx.jsr223
  - Class RexxCompiledScript
    - Extends javax.script.CompiledScript and implements the
      - Interface javax.script.Invocable
    - Each Rexx script will be first "compiled" (tokenized) and then executed
      - Allows reusing the script without any overhead
- JSR-223 samples get installed with BSF4ooRexx
  - C.f. Directory "samples/Java/jsr223"

 © Rony G. Flatscher

JSR-223, Evaluating a Rexx-Script

- Create an instance of javax.script.ScriptEngineManager
  - Use e.g. getEngineByName("Rexx") to get an instance of the class RexxEngine
    - Will have a default SimpleScriptContext with a SimpleBindings defined for the engine level
    - The Rexx engine's .input, .output, .error monitors will be set to the Java reader (a java.io.Reader), writer (a java.io.Writer), errorWriter (a java.io.Writer) fields of the SimpleScriptContext
    - Before using it for the first time, the Rexx interpreter instance itself can be configured according to the native Rexx APIs, e.g.
      - Defining additional Rexx file extensions, exit handlers, etc.

 © Rony G. Flatscher

JSR-223, "Demo_01", 1

- Demonstrates how to execute a Rexx script stored in a Java String using javax.script (a.k.a. JSR-223)
- Note:
  - The output from the Rexx program is done via Java!

 © Rony G. Flatscher

JSR-223, "Demo_01", 2
Java Code (Demo_01.java)

```
import javax.script.ScriptEngineManager;
import org.rexxla.bsf.engines.rexx.jsr223.RexxScriptEngine;

public class Demo_01    // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");

        try
        {
            String rexxCode="say 'Hello, world from Rexx!'";
            rse.eval(rexxCode);
        }
        catch (Exception exc)
        {
            System.err.println(exc);
            System.exit(-1);
        }
    }
}
```

 © Rony G. Flatscher

JSR-223, "Demo_01", 3
Output of "java Demo_01"

```
Hello, world from Rexx!
```

 © Rony G. Flatscher

JSR-223, "Demo_02", 1

- Demonstrates how to execute a Rexx script stored in an external file
- Uses the Java class java.io.FileReader
- Note:
  - The output from the Rexx program is done via Java!

 © Rony G. Flatscher

JSR-223, "Demo_02", 2
Java Code (Demo_02.java)

```
import javax.script.ScriptEngineManager;
import org.rexxla.bsf.engines.rexx.jsr223.RexxScriptEngine;
import java.io.FileReader;

public class Demo_02    // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");

        try
        {
            rse.eval(new FileReader("demo_02.rex"));
        }
        catch (Exception exc)
        {
            System.err.println(exc);
            System.exit(-1);
        }
    }
}
```


© Rony G. Flatscher

JSR-223, "Demo_02", 3
Rexx Code (demo_02.rex)

```
say 'Hello, world from Rexx!'
```


© Rony G. Flatscher

JSR-223, "Demo_02", 4
Output of "java Demo_02"

```
Hello, world from Rexx!
```

 © Rony G. Flatscher

JSR-223, "Demo_03", 1

- Demonstrates how to execute a Rexx script stored in an external file
- Uses the Java class java.io.FileReader
- Note:
  - The output from the Rexx program is done via Java!
- Invoked Rexx script will show
  - The result of the "PARSE SOURCE" keyword statement
  - The Rexx arguments received from Java
    - There will always be a Rexx directory appended as the last argument with an entry named "SCRIPTCONTEXT"

 © Rony G. Flatscher

JSR-223, "Demo_03", 2
Java Code (Demo_03.java)

```
import java.io.FileReader;
import javax.script.ScriptEngineManager;
import org.rexxla.bsf.engines.rexx.RexxEngine;
import org.rexxla.bsf.engines.rexx.jsr223.*;

public class Demo_03    // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");

        try
        {
            rse.eval(new FileReader("demo_03.rex"));  // now let us execute the Rexx script
        }
        catch (Exception exc)
        {
            System.err.println(exc);
            System.exit(-1);
        }
    }
}
```

 © Rony G. Flatscher

JSR-223, "Demo_03", 3
Rexx Code (demo_03.rex)

```
parse source s
say "parse source: ["s"]"
say

say "received the following arguments:"
do i=1 to arg()
   say "   arg #" i": ["arg(i)"]"
end
```


© Rony G. Flatscher

JSR-223, "Demo_03", 4
Output of "java Demo_03"

```
parse source: [MACOSX SUBROUTINE
filename_created_by_org_rexxla_bsf_engines_rexx_jsr223_RexxScriptEngine-
6e2c634b_at_2016_08_22T02_16_45_308Z.rex]

received the following arguments:
   arg # 1: [a Directory]
```


© Rony G. Flatscher

JSR-223, "Demo_04", 1

- Like "Demo_03" before, except
  - The filename gets defined in the ScriptContext for the engine scope's Bindings
  - The RexxEngine object will use that filename from the engine scope's Bindings to determine the script's filename
    - The "PARSE SOURCE" keyword statement will therefore fetch that name

 © Rony G. Flatscher

JSR-223, "Demo_04", 2
Java Code (Demo_04.java)

```
import javax.script.*;
import java.io.FileReader;
import org.rexxla.bsf.engines.rexx.jsr223.RexxScriptEngine;

public class Demo_04     // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");

        try
        {
            String filename="demo_03.rex";       // define the filename
                // add the filename to the engine's SimpleBindings
            ScriptContext sc=rse.getContext();  // get the default ScriptContext
            sc.setAttribute(ScriptEngine.FILENAME,filename,ScriptContext.ENGINE_SCOPE);

            rse.eval(new FileReader(filename), sc);  // now let us execute the Rexx script
        }
        catch (Exception exc)
        {
            System.err.println(exc);
            System.exit(-1);
        }
    }
}
```


© Rony G. Flatscher

JSR-223, "Demo_04", 3
Output of "java Demo_04"

```
parse source: [MACOSX SUBROUTINE demo_03.rex]

received the following arguments:
   arg # 1: [a Directory]
```


© Rony G. Flatscher

JSR-223, "Demo_05", 1

- Demonstrates how to execute a Rexx script stored in an external file, more than once
  - Each invocation of a Rexx script will cause it to be compiled (tokenized) and stored as the "current" script object
- Invoked Rexx script will show
  - The result of the "PARSE SOURCE" keyword statement
  - The Rexx arguments received from Java

 © Rony G. Flatscher

JSR-223, "Demo_05", 2
Java Code (Demo_05.java)

```
import javax.script.*;
import java.io.FileReader;
import org.rexxla.bsf.engines.rexx.jsr223.*;

public class Demo_05    // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");
        try
        {
            String filename="demo_03.rex";      // define the filename
                // add the filename to the engine's SimpleBindings
            ScriptContext sc=rse.getContext();  // get the default ScriptContext
            sc.setAttribute(ScriptEngine.FILENAME,filename,ScriptContext.ENGINE_SCOPE);
            rse.eval(new FileReader(filename), sc);  // now let us execute the Rexx script
            System.err.println("\n... Java: about to reuse the last used Rexx script ...\n");
                // add arguments for the script to the ENGINE_SCOPE bindings
            sc.setAttribute(ScriptEngine.ARGV,
                        new Object[] {"one", null, java.util.Calendar.getInstance()},
                        ScriptContext.ENGINE_SCOPE);
                // the RexxScriptEngine always compiles the last script and
                // makes it available with the getCurrentScript() method
            rse.getCurrentScript().eval();          // now let us re-execute the Rexx script
        }
        catch (Exception exc){ System.err.println(exc); System.exit(-1); }
    }
}
```

 © Rony G. Flatscher

JSR-223, "Demo_05", 3
Output of "java Demo_05"

```
parse source: [MACOSX SUBROUTINE demo_03.rex]

received the following arguments:
   arg # 1: [a Directory]

... Java: about to reuse the last used Rexx script ...

parse source: [MACOSX SUBROUTINE demo_03.rex]

received the following arguments:
   arg # 1: [one]
   arg # 2: [The NIL object]
   arg # 3: [java.util.GregorianCalendar@277050dc]
   arg # 4: [a Directory]
```


© Rony G. Flatscher

JSR-223, "Demo_06", 1

- Demonstrates how to execute a Rexx script stored in an external file, more than once
  - Each invocation of a Rexx script will cause it to be compiled (tokenized) and stored as the "current" script object
- Invoked Rexx script will show
  - The result of the "PARSE SOURCE" keyword statement
  - The Rexx arguments received from Java
  - Accessing the supplied ScriptContext and interacting with it
  - Demonstrate how to determine whether a value is Java proxy and if so, send it the toString message

 © Rony G. Flatscher

JSR-223, "Demo_06", 2
Java Code (Demo_06.java)

```
import javax.script.*;
import java.io.FileReader;
import org.rexxla.bsf.engines.rexx.jsr223.*;

public class Demo_06    // demo evaluating a Rexx script
{
    public static void main (String args[])
    {
        ScriptEngineManager manager = new ScriptEngineManager();
        RexxScriptEngine rse=(RexxScriptEngine) manager.getEngineByName("Rexx");
        try
        {
            String filename="demo_06.rex";      // define the filename
                // add the filename to the engine's SimpleBindings
            ScriptContext sc=rse.getContext();  // get the default ScriptContext
            sc.setAttribute(ScriptEngine.FILENAME,filename,ScriptContext.ENGINE_SCOPE);
            rse.eval(new FileReader(filename), sc);  // now let us execute the Rexx script
            System.err.println("\n... Java: about to reuse the last used Rexx script ...\n");
                // add arguments for the script to the ENGINE_SCOPE bindings
            sc.setAttribute(ScriptEngine.ARGV,
                        new Object[] {"one", null, java.util.Calendar.getInstance()},
                        ScriptContext.ENGINE_SCOPE);
                // the RexxScriptEngine always compiles the last script and
                // makes it available with the getCurrentScript() method
            rse.getCurrentScript().eval();         // now let us re-execute the Rexx script
        }
        catch (Exception exc){ System.err.println(exc); System.exit(-1); }
    }
}
```

 © Rony G. Flatscher

JSR-223, "Demo_06", 3
Rexx Code (demo_06.rex)

```
parse source s
say "parse source: ["s"]"
say
-- demonstrate how to access and use the ScriptContext
slotDir=arg(arg())        -- last argument is a directory containing "SCRIPTCONTEXT"
sc=slotDir~scriptContext  -- fetch the ScriptContext object
say "ScriptContext field: ENGINE_SCOPE:" pp(sc~engine_scope)
say "ScriptContext field: GLOBAL_SCOPE:" pp(sc~global_scope)
say
-- import the Java class that defines some Constants like FILENAME, ARGV ...
seClz=bsf.importClass("javax.script.ScriptEngine")
say "ScriptEngine field: FILENAME="pp(seClz~filename)
say "ScriptEngine field: ARGV    ="pp(seClz~argv)
say
key=seClz~FILENAME      -- get string value for FILENAME entry in Bindings
say "value of ScriptEngine static field named ""FILENAME"":" pp(key)
say "  fetch filename from ScriptContext                :" pp(sc~getAttribute(key))
say "  fetch scope (engine or global) from Scriptcontext:" pp(sc~getAttributesScope(key))
say
key=seClz~ARGV          -- get string value for ARGV entry in Bindings
say "value of ScriptEngine static field named ""ARGV""    :" pp(key)
say "  fetch filename from ScriptContext                :" pp(sc~getAttribute(key))
say "  fetch scope (engine or global) from Scriptcontext:" pp(sc~getAttributesScope(key))
say "---"
say "Rexx received the following arguments:"
do i=1 to arg()
   val=arg(i)
   str="    arg("i")=["
   if val~isA(.bsf) then str=str || val~toString"]"
                    else str=str || val"]"
   say str
end
```


© Rony G. Flatscher

JSR-223, "Demo_06", 4
Output of "java Demo_06"

```
parse source: [MACOSX SUBROUTINE demo_06.rex]

ScriptContext field: ENGINE_SCOPE: [100]
ScriptContext field: GLOBAL_SCOPE: [200]

ScriptEngine field: FILENAME=[javax.script.filename]
ScriptEngine field: ARGV    =[javax.script.argv]

value of ScriptEngine static field named "FILENAME": [javax.script.filename]
  fetch filename from ScriptContext                : [demo_06.rex]
  fetch scope (engine or global) from Scriptcontext: [100]

value of ScriptEngine static field named "ARGV"    : [javax.script.argv]
  fetch filename from ScriptContext                : [The NIL object]
  fetch scope (engine or global) from Scriptcontext: [-1]
---
Rexx received the following arguments:
    arg(1)=[a Directory]

... Java: about to reuse the last used Rexx script …

parse source: [MACOSX SUBROUTINE demo_06.rex]

ScriptContext field: ENGINE_SCOPE: [100]
ScriptContext field: GLOBAL_SCOPE: [200]
```

---> Output continued on next slide ...

 © Rony G. Flatscher

JSR-223, "Demo_06", 5
Output of "java Demo_06" (Continued)

```
ScriptEngine field: FILENAME=[javax.script.filename]
ScriptEngine field: ARGV    =[javax.script.argv]

value of ScriptEngine static field named "FILENAME": [javax.script.filename]
  fetch filename from ScriptContext                : [demo_06.rex]
  fetch scope (engine or global) from Scriptcontext: [100]

value of ScriptEngine static field named "ARGV"    : [javax.script.argv]
  fetch filename from ScriptContext                : [[Ljava.lang.Object;@5387f9e0]
  fetch scope (engine or global) from Scriptcontext: [100]
---
Rexx received the following arguments:
    arg(1)=[one]
    arg(2)=[The NIL object]
arg(3)=[java.util.GregorianCalendar[time=1471834275785,areFieldsSet=true,areAllFieldsSet=tru
e,lenient=true,zone=sun.util.calendar.ZoneInfo[id="Europe/Vienna",offset=3600000,dstSavings=
3600000,useDaylight=true,transitions=139,lastRule=java.util.SimpleTimeZone[id=Europe/Vienna,
offset=3600000,dstSavings=3600000,useDaylight=true,startYear=0,startMode=2,startMonth=2,star
tDay=-1,startDayOfWeek=1,startTime=3600000,startTimeMode=2,endMode=2,endMonth=9,endDay=-
1,endDayOfWeek=1,endTime=3600000,endTimeMode=2]],firstDayOfWeek=1,minimalDaysInFirstWeek=1,E
RA=1,YEAR=2016,MONTH=7,WEEK_OF_YEAR=35,WEEK_OF_MONTH=4,DAY_OF_MONTH=22,DAY_OF_YEAR=235,DAY_O
F_WEEK=2,DAY_OF_WEEK_IN_MONTH=4,AM_PM=0,HOUR=4,HOUR_OF_DAY=4,MINUTE=51,SECOND=15,MILLISECOND
=785,ZONE_OFFSET=3600000,DST_OFFSET=3600000]]
    arg(4)=[a Directory]
```


© Rony G. Flatscher

Roundup and Outlook

- Roundup
  - The JSR-223 support for ooRexx allows any Rexx and ooRexx-programs to be easily executed by Java
  - BSF4ooRexx allows Java programmers to use two different access types to take fully advantage of ooRexx
    - Apache BSF (Bean scripting framework) and
    - JSR-223: the javax.script framework introduced with Java 1.6
  - There is no excuse for Java programmers not to use scripting engines like BSF4ooRExx for scripting Java applications anymore! ;)

 © Rony G. Flatscher