Journal of Statistical Software

MMMMMM YYYY, Volume VV, Issue II. *doi: 10.18637/jss.v000.i00*

R6 Generator Maven Plugin: Bridging Java and R6 for Package Development

Robert Challen
University of Exeter

Abstract

This article introduces a novel approach to integrating Java code into R, that offers distinct advantages in terms of ease of development and maintenance to Java programmers wishing to expose their libraries as R packages. It builds on the low level interface provided by the **rJava** package, Java code analysis tools, and the Maven compiler framework, to programmatically generate R package code. The R6 Generator Maven Plugin provides a Java annotation processor and R code generator that creates an R package exposing an **R6** based interface to Java code. By working at compile time it ensures the bi-directional transfer of data between R and Java is type-safe and fast, it allows a simple route to maintainence of R ports of Java libraries, and it facilitates package documentation.

Keywords: Java, R6, Maven.

1. Introduction

The R6 Generator Maven Plugin provides a low friction route to integrating Java code into R programs. **rJava** is a low level interface for using Java libraries in R via the Java Native Interface (JNI). However use of this low level interface has a steep learning curve for both R and Java programmers. The **jsr223** package aims to made to reduce this friction in the situation where dynamic interaction between R and Java is needed, by integration of Java based scripting languages, such as Kotlin or Groovy. This is a relatively complex solution for the simpler situation where a developer wishes to package Java based functions for use in R programs.

Strengths and weaknesses of Java and R R ease of data import and integration, spatial support R visualisation R data wrangling R REPL Java ease of multithreaded processing. Java dependency and classpath management. Java VM debugging versus R debugging

The existing approaches - Integration between Java and R has existed since the release of

rJava in 2003, however successful use of Java libraries within R has remained complex and rJava, rjsr223 jsr223 reference - review of Java integration JNI & rJava features and limitations rJava 2 level api jsr223 benefits and limitations Data only integration e.g. Apache Arrow.

2. Use cases

- Java library development for R use
- Adaption of java library for R use

3. Desiderata / Design rationale

R as REPL, java as backend

No knowledge of JNI, rJava or intermediate languages required. High performance - minimise interpretation, Compile time - strongly typed inputs to R - minimise data transfer overhead. minimise use of reflection Accurate data transformations and round trip Dependency management - Separation of concerns & isolation - java based API layer to isolate Java changes from R API changes Predictability - Runtime library - predictable R data in Java code isolating java from R type system Seamless R use of library - R6 class hierarchy - native R facade to Java code

4. Terminology and concepts

Maven plugin - Runtime dependency - R types for java

5. Feature documentation

5.1. Minimal example

```
1  package uk.co.terminological.rjava.test;
2
3  import org.slf4j.Logger;
4  import org.slf4j.LoggerFactory;
5
6  import uk.co.terminological.rjava.RClass;
7  import uk.co.terminological.rjava.RMethod;
8  import uk.co.terminological.rjava.types.RDataframe;
9
10 /**
11  * This class is a very basic example of the features of the rJava maven plugin. <br/>
12  * The class is annotated with an @RClass to identify it as part of the R API. <br/>
13  */
14 @RClass
15 public class MinimalExample {
16
```

```
    static Logger log = LoggerFactory.getLogger(MinimalExample.class);

    @RMethod(examples = {
        "J = testRapi::JavaApi$new()",
        "minExample = J$MinimalExample$new()",
        "minExample$demo(dataframe=tibble::tibble(input=c(1,2,3)), message='Hello world')"
    })
    /**
     * Documentation of the method can be done in JavaDoc and these will be present in the R documentation
     * @param dataframe - a dataframe with an arbitrary number of columns
     * @param message - a message
     * @return the dataframe unchanged
     *
     */
    public RDataframe demo(RDataframe dataframe, String message) {
        log.info("this dataframe has nrow="+dataframe.nrow());
        log.info(message);
        return dataframe;
    }

}
```

```
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.8</maven.compiler.source>
        <maven.compiler.target>1.8</maven.compiler.target>
        <r6.version>master-SNAPSHOT</r6.version>
    </properties>
...
    <dependencies>
        <dependency>
            <groupId>com.github.terminological</groupId>
            <artifactId>r6-generator-runtime</artifactId>
            <version>${r6.version}</version>
        </dependency>
    </dependencies>
...
    <!-- Resolve runtime library on github -->
    <repositories>
        <repository>
            <id>jitpack.io</id>
            <url>https://jitpack.io</url>
        </repository>
    </repositories>

    <!-- Resolve maven plugin on github -->
    <pluginRepositories>
        <pluginRepository>
            <id>jitpack.io</id>
            <url>https://jitpack.io</url>
        </pluginRepository>
    </pluginRepositories>
...
            <plugin>
                <groupId>com.github.terminological</groupId>
                <artifactId>r6-generator-maven-plugin</artifactId>
                <version>${r6.version}</version>
                <configuration>
```

```
37                     <packageData>
38                         <title>A test library</title>
39                         <version>0.01</version>
40                         <debug>true</debug>
41                         <rjavaOpts>
42                             <rjavaOpt>-Xmx256M</rjavaOpt>
43                         </rjavaOpts>
44                         <packageName>testRapi</packageName>
45                         <license>MIT</license>
46                         <description>An optional long description of the package</description>
47                         <maintainerName>test forename</maintainerName>
48                         <maintainerFamilyName>optional surname</maintainerFamilyName>
49                         <maintainerEmail>test@example.com</maintainerEmail>
50                     </packageData>
51                     <outputDirectory>${project.basedir}/r-library</outputDirectory>
52                 </configuration>
53                 <executions>
54                     <execution>
55                         <id>generate-r-library</id>
56                         <goals>
57                             <goal>generate-r-library</goal>
58                         </goals>
59                     </execution>
60                 </executions>
61             </plugin>
```

5.2. R6 class generation

5.3. Maven packaging

5.4. Generated R6 documentation

5.5. Datatype transformation

Java to R

R to Java - runtime library

5.6. Debugging Java code

- Datasets for testing Java code / serialisation of inputs
-

6. Benefits

- Low friction
- Minimal dependencies introduced.
- Java Library dependency packaging
- Accurate bidirectional transfer of data between R and Java
- Type safety
- Code completion and type hinting in R
- Fluent use of R data in java
- R Documentation
- Separation of concerns
- Maintenance

7. Limitations

- Fat Jar bloat
- Recompilation and iterative development
- Naming collisions
- Concurrent use of Multiple rJava libraries
- Java class caching

8. Future development

- Tighter integration of multithreading and promises.
- Matrix support
- Named rows in dataframes
- R test case generation
- Java object bindings
- Purrr style lists in data frames

9. Conclusion

Affiliation:

Robert Challen
University of Exeter
EPSRC Centre for Predictive Modelling in Healthcare, University of Exeter, Exeter, Devon, UK.
E-mail: rc538@exeter.ac.uk
URL: https://github.com/terminological/r6-generator-maven-plugin

Journal of Statistical Software http://www.jstatsoft.org/
published by the Foundation for Open Access Statistics http://www.foastat.org/
MMMMMM YYYY, Volume VV, Issue II *Submitted:* yyyy-mm-dd
doi:10.18637/jss.v000.i00 *Accepted:* yyyy-mm-dd