How and why I turned my old Java projects into a first-class serverless component

by Mario Fusco & Matteo Mortari

Drools
GraalVM
QUARKUS
Agenda

➢ A quick introduction
➢ From drl to Java: the executable model
Agenda

➢ A quick introduction
➢ From drl to Java: the executable model

➢ What is GraalVM
➢ AoT compilation
➢ Limitations
➢ Refactoring Drools to (natively) compile on GraalVM
Agenda

➢ A quick introduction
➢ From drl to Java: the executable model

➢ What is GraalVM
➢ AoT compilation
➢ Limitations
➢ Refactoring Drools to (natively) compile on GraalVM

➢ What is Quarkus
➢ Quarkus features
➢ Integrating Drools with Quarkus
➢ Writing a Quarkus extension for Drools
Introducing Drools

- **Easier to understand** → Requirements can be more naturally translated into rules. It is more likely for a technically skilled business analyst to verify, validate or even change a rule than a piece of Java code

- **Improved maintainability** → We don’t care about *how* to implement a solution only *what* needs to be done to solve a problem

- **Deals with evolving complexity** → It’s easier to modify a rule than a Java program and to determine the impact of this change on the rest of the application

- **Modularity** → Each rule models an isolated and small portion of your business logic and is not part of a monolithic program

- **Clear separation of business logic from the rest of the system** → Business and infrastructural code have very different lifecycles

- **Complex Event Processing** → Facts can be handled like timestamped events allowing temporal reasoning on them
A simple rule set

**Rule: RaiseAlarm**

- ** WHEN ** exists Fire()

- ** THEN **

  - insert( new Alarm( "house1" ) );
  - System.out.println( "Raise the Alarm" );

** Rule: CancelAlarm **

- ** WHEN ** not Fire()

- ** THEN **

  - delete( a );
  - System.out.println( "Cancel the Alarm" );

**Rule: TurnSprinklerOn**

- ** WHEN **

  - s : Sprinkler( on == false )
  - f : Fire( room == s.room )

- ** THEN **

  - modify( s ) { setOn( true ) }
  - System.out.println( "Turn on the sprinkler for room " + f.getRoom().getName() );

**Rule: TurnSprinklerOff**

- ** WHEN **

  - s : Sprinkler( on == true )
  - not Fire( room == s.room )

- ** THEN **

  - modify( s ) { setOn( false ) }
  - System.out.println( "Turn off the sprinkler for room " + s.getRoom().getName() );

**Rule: OK**

- ** WHEN **

  - not Alarm()
  - not Sprinkler( on == true )

- ** THEN **

  - System.out.println( "Everything is ok" );

Pattern-matching against objects in the Working Memory

Code executed when a match is found
What is GraalVM™

➢ A **polyglot VM** with cross-language JIT supporting
  • Java Bytecode and JVM languages
  • Interop with different languages
  • Dynamic languages through Truffle API

➢ Cross-language interop out of the box
  • Simple AST-based interpreter
  • JIT across language boundaries

➢ Support for **native binary compilation** (SubstrateVM)
  • faster boot-up
  • lower memory footprint
AoT compilation with GraalVM

- Static analysis
- Closed world assumption
- Dead code elimination: classes, fields, methods, branches
AoT compilation with GraalVM

- Static analysis
- Closed world assumption
- Dead code elimination: classes, fields, methods, branches

🚀 Fast process start
🔍 Less memory
💾 Small size on disk
GraalVM Limitations

Dynamic Classloading
Deploying jars, wars, etc. at runtime impossible

```java
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        return defineClass(name, bytecode, 0, bytecode.length);
    }
}
```
GraalVM Limitations

Dynaminc Classloading
Deloying jars, wars, etc. at runtime impossible

```java
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass( String name, byte[] bytecode ) {
        return defineClass( name, bytecode, 0, bytecode.length );
    }
}
```
GraalVM Limitations

Dynamic Classloading

Deploying jars, wars, etc. at runtime impossible

```
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        return defineClass(name, bytecode, 0, bytecode.length);
    }
}
```

JVMTI, JMX + other native VM interfaces

No agents → No JRebel, Byteman, profilers, tracers

Miscellaneous

› Security Manager
› finalize() (deprecated anyway)
› InvokeDynamic and MethodHandle
GraalVM Limitations

Dynamic Classloading
Deloying jars, wars, etc. at runtime impossible

```java
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass( String name, byte[] bytecode ) {
        return defineClass( name, bytecode, 0, bytecode.length );
    }
}
```

JVMTI, JMX + other native VM interfaces
No agents → No JRebel, Byteman, profilers, tracers

Miscellaneous
➢ Security Manager
➢ finalize()(deprecated anyway)
➢ InvokeDynamic and MethodHandle

Unsupported
GraalVM Limitations

Dynamic Classloading
Deloying jars, wars, etc. at runtime impossible

```
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass( String name, byte[] bytecode ) {
        return defineClass( name, bytecode, 0, bytecode.length );
    }
}
```

JVMTI, JMX + other
native VM interfaces
No agents → No JRebel, Byteman, profilers, tracers

Miscellaneous
➢ Security Manager
➢ finalize() (deprecated anyway)
➢ InvokeDynamic and MethodHandle

Reflection
Requires registration (closed world assumption)
GraalVM Limitations

Dynaminc Classloading
Deloying jars, wars, etc. at runtime impossible

```
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        return defineClass(name, bytecode, 0, bytecode.length);
    }
}
```

JVMTI, JMX + other native VM interfaces
No agents → No JRebel, Byteman, profilers, tracers

Miscellaneous
- Security Manager
- finalize() (depreceated anyway)
- InvokeDynamic and MethodHandle

Reflection
Requires registration (closed world assumption)

- `H:ReflectionConfigurationFiles=src/main/resources/reflection.json`
rule "Older than Mark" when
  $p1: Person( name == "Mark" )
  $p2: Person( name != "Mark", age > $p1.age )
then
  System.out.println( $p1.getName() +
    " is older than " + $p2.getName() );
end

Variable<Person> markV = declarationOf( Person.class );
Variable<Person> olderV = declarationOf( Person.class );

Rule rule = rule( "Older than Mark" )
  .build(
      pattern(markV)
        .expr("exprA", p -> p.getName().equals( "Mark" ),
          alphaIndexedBy( String.class, ConstraintType.EQUAL, 1, p -> p.getName(), "Mark" ),
          reactOn( "name", "age" )),
      pattern(olderV)
        .expr("exprB", p -> !p.getName().equals("Mark"),
          alphaIndexedBy( String.class, ConstraintType.NOT_EQUAL, 1, p -> p.getName(), "Mark" ),
          reactOn( "name" )
        .expr("exprC", markV, (p1, p2) -> p1.getAge() > p2.getAge(),
          betaIndexedBy( int.class, ConstraintType.GREATER_THAN, 0, p -> p.getAge(), p -> p.getAge() ),
          reactOn( "age" )
        ),
    on(olderV, markV).execute((p1, p2) -> System.out.println( p1.getName() + " is older than " + p2.getName() ))
  );
Drools on GraalVM - Executable Model

```
rule "Older than Mark" when
  $p1: Person( name == "Mark" )
  $p2: Person( name != "Mark", age > $p1.age )
then
  System.out.println( $p1.getName() + 
    " is older than " + $p2.getName() );
end
```

```java
Variable<Person> markV = declarationOf( Person.class );
Variable<Person> olderV = declarationOf( Person.class );

Rule rule = rule( "Older than Mark" )
  .build(
    pattern(markV)
      .expr("exprA", p -> p.getName().equals( "Mark" ),
        alphaIndexedBy( String.class, ConstraintType.EQUAL, 1, p -> p.getName(), "Mark" ),
        reactOn( "name", "age" ) ),
    pattern(olderV)
      .expr("exprB", p -> !p.getName().equals("Mark"),
        alphaIndexedBy( String.class, ConstraintType.NOT_EQUAL, 1, p -> p.getName(), "Mark" ),
        reactOn( "name" ) )
      .expr("exprC", markV, (p1, p2) -> p1.getAge() > p2.getAge(),
        betaIndexedBy( int.class, ConstraintType.GREATER_THAN, 0, p -> p.getAge(), p -> p.getAge() ),
        reactOn( "age" ) ),
    on(olderV, markV).execute((p1, p2) -> System.out.println( p1.getName() + " is older than " + p2.getName() ))
  );
```
Drools on GraalVM - Executable Model

```java
rule "Older than Mark" when
   $p1: Person( name == "Mark" )
   $p2: Person( name != "Mark", age > $p1.age )
then
   System.out.println( $p1.getName() + " is older than " + $p2.getName() );
end
```

Variable<Person> markV = declarationOf( Person.class );
Variable<Person> olderV = declarationOf( Person.class );

Rule rule = rule( "Older than Mark" )
   .build(
      pattern(markV)
         .expr( "exprA", p -> p.getName().equals( "Mark" ),
               alphaIndexedBy( String.class, ConstraintType.EQUAL, 1, p -> p.getName(), "Mark" ),
               reactOn( "name", "age" ) ),
      pattern(olderV)
         .expr( "exprB", p -> !p.getName().equals( "Mark" ),
               alphaIndexedBy( String.class, ConstraintType.NOT_EQUAL, 1, p -> p.getName(), "Mark" ),
               reactOn( "name" )
         .expr( "exprC", markV, (p1, p2) -> p1.getAge() > p2.getAge(),
               betaIndexedBy( int.class, ConstraintType.GREATER_THAN, 0, p -> p.getAge(), p -> p.getAge() ),
               reactOn( "age" ) ),
      on(olderV, markV).execute((p1, p2) -> System.out.println( p1.getName() + " is older than " + p2.getName() ))
   );
```

Indexes and reactivity explicitly defined
Executable Model at a glance

➢ A pure Java DSL for Drools rules authoring
➢ A pure Java canonical representation of a rule base
➢ Automatically generated by Maven plugin or Quarkus extension
  • Can be embedded in jar
  • Faster boot
➢ Improve Backward/Forward compatibility
➢ Allow for faster prototyping and experimentation of new features
➢ Prerequisite to make Drools natively compilable on GraalVM
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        throw new UnsupportedOperationException();
    }
}

Drools on GraalVM – other refactors

Dynamic class definition is no longer necessary
Drools on GraalVM – other refactors

Dynamic class definition is no longer necessary

```java
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        throw new UnsupportedOperationException();
    }
}
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<kmodule xmlns="http://jboss.org/kie/6.0.0/kmodule">
    <kbase name="simpleKB"
          packages="org.drools.simple.project">
        <ksession name="simpleKS" default="true"/>
    </kbase>
</kmodule>
```

```javascript
var m = KieServices.get().newKieModuleModel();
var kb = m.newKieBaseModel("simpleKB");
kb.setEventProcessingMode(CLOUD);
kb.addPackage("org.drools.simple.project");
var ks = kb.newKieSessionModel("simpleKS");
ks.setDefault(true);
ks.setType(STATEFUL);
ks.setClockType(ClockTypeOption.get("realtime"));
```
Drools on GraalVM – other refactors

Dynamic class definition is no longer necessary

```java
public class InternalClassLoader extends ClassLoader {
    public Class<?> defineClass(String name, byte[] bytecode) {
        throw new UnsupportedOperationException();
    }
}
```

```xml
<xml version="1.0" encoding="UTF-8">
<kmodule xmlns="http://jboss.org/kie/6.0.0/kmodule">
    <kbase name="simpleKB">
        <packages="org.drools.simple.project">
            <ksession name="simpleKS" default="true"/>
        </packages>
    </kbase>
</kmodule>
```

```java
var m = KieServices.get().newKieModuleModel();
var kb = m.newKieBaseModel("simpleKB");
kb.setEventProcessingMode(CLOUD);
kb.addPackage("org.drools.simple.project");
var ks = kb.newKieSessionModel("simpleKS");
ks.setDefault(true);
ks.setType(STATEFUL);
ks.setClockType(ClockTypeOption.get("realtime"));
```

```java
org.kie.api.io.KieResources =
    org.drools.core.io.impl.ResourceFactoryServiceImpl
org.kie.api.marshalling.KieMarshallers =
    org.drools.core.marshalling.MarshallerProviderImpl
org.kie.api.concurrent.KieExecutors =
    org.drools.core.concurrent.ExecutorProviderImpl
```

```java
Map<Class<?>, Object> serviceMap = new HashMap<>();
void wireServices() {
    serviceMap.put(ServiceInterface.class,
        Class.forName("org.drools.ServiceImpl").newInstance());
    // ... more services here
}
```
Demo 1
Drools on GraalVM
JVM vs. Native

Command being timed: "/run-java.sh Mario 18"
User time (seconds): 1.24
System time (seconds): 0.18
Percent of CPU this job got: 183%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.73
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 103864
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 22893
Voluntary context switches: 2124
Involuntary context switches: 35
Swaps: 0
File system inputs: 0
File system outputs: 8
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0

Command being timed: "/org.drools.graal.kjar.droolsmain Mario 18"
User time (seconds): 0.00
System time (seconds): 0.00
Percent of CPU this job got: 75%
Elapsed (wall clock) time (h:mm:ss or m:ss): 0:00.00
Average shared text size (kbytes): 0
Average unshared data size (kbytes): 0
Average stack size (kbytes): 0
Average total size (kbytes): 0
Maximum resident set size (kbytes): 16068
Average resident set size (kbytes): 0
Major (requiring I/O) page faults: 0
Minor (reclaiming a frame) page faults: 395
Voluntary context switches: 1
Involuntary context switches: 0
Swaps: 0
File system inputs: 0
File system outputs: 0
Socket messages sent: 0
Socket messages received: 0
Signals delivered: 0
Page size (bytes): 4096
Exit status: 0
Introducing QUARKUS

A Framework for writing (fast and lightweight) Java applications
Introducing QUARKUS

➢ A Framework for writing (fast and lightweight) Java applications
➢ (Optionally) allowing generation of native executable via GraalVM
Introducing QUARKUS

➢ A Framework for writing (fast and lightweight) Java applications
➢ (Optionally) allowing generation of native executable via GraalVM
➢ Based on existing standard
  • Servlet
  • JAX-RS
  • JPA, JDBC
  • CDI
  • Bean Validation
  • Transactions
  • Logging
  • Microprofile
Introducing QUARKUS

➢ A Framework for writing (fast and lightweight) Java applications
➢ (Optionally) allowing generation of native executable via GraalVM
➢ Based on existing standard
  ● Servlet
  ● JAX-RS
  ● JPA, JDBC
  ● CDI
  ● Bean Validation
  ● Transactions
  ● Logging
  ● Microprofile
➢ Out-of-the-box integration with libraries that you already know
Why Quarkus

- Lower memory usage
- Faster startup
- Optimized for short-lived processes
- Kubernetes Native
- Live reload
- Microservices
Why Quarkus

Lower memory usage

Optimized for short-lived processes

Live reload

Faster startup

Kubernetes Native

Microservices

Memory (RSS) in Megabytes

<table>
<thead>
<tr>
<th></th>
<th>REST</th>
<th>REST + CRUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 MB</td>
<td>140 MB</td>
<td>130 MB</td>
</tr>
<tr>
<td>74 MB</td>
<td></td>
<td>218 MB</td>
</tr>
</tbody>
</table>

Boot + First Response Time in Seconds

<table>
<thead>
<tr>
<th></th>
<th>REST</th>
<th>REST + CRUD</th>
</tr>
</thead>
<tbody>
<tr>
<td>.014 sec</td>
<td>.75 sec</td>
<td>4.3 sec</td>
</tr>
<tr>
<td>.055 sec</td>
<td>2.5 sec</td>
<td>9.5 sec</td>
</tr>
</tbody>
</table>

MEMORY & BOOT + FIRST RESPONSE TIME
Introducing Kogito

• A cloud-native development, deployment and execution platform for business automation:
  ○ Rules and Decisions
  ○ Processes and Cases

• ... under the covers
  ○ the backbone is code generation based on business assets
  ○ executable model for the process/rule/decision definitions
  ○ type safe data model that encapsulates variables
  ○ REST api for each public business process/decision/rule
A simple Quarkus-based REST endpoint using Kogito

```java
@Path("/candrink/{name}/{age}")
public class CanDrinkResource {

    @Inject @Named("canDrinkKS")
    RuleUnit<SessionMemory> ruleUnit;

    @GET
    @Produces(MediaType.TEXT_PLAIN)
    public String canDrink( @PathParam("name") String name, @PathParam("age") int age ) {
        SessionMemory memory = new SessionMemory();

        Result result = new Result();
        memory.add(result);
        memory.add(new Person( name, age ));

        ruleUnit.evaluate(memory);

        return result.toString();
    }
}
```
Demo 2
Drools on Quarkus
Thanks ... Questions?