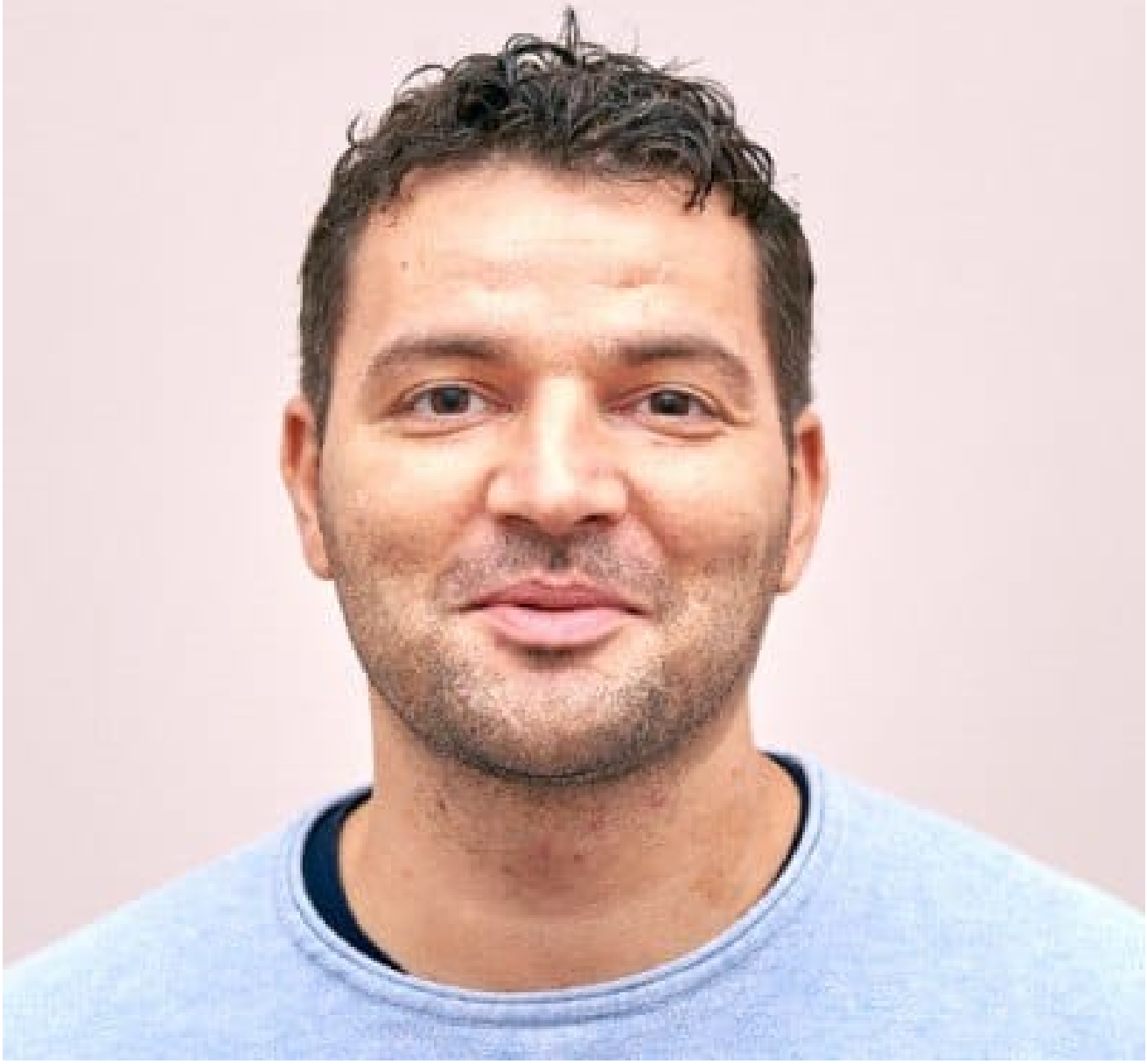
Drools Spring Integration

Last updated: January 8, 2024

Written by: baeldung (https://www.baeldung.com/author/baeldung)



Reviewed by: Slaviša Avramović
(https://www.baeldung.com/editor/slavisa-author)

Spring (https://www.baeldung.com/category/spring) +

Drools (https://www.baeldung.com/tag/drools)

Rule Engine (https://www.baeldung.com/tag/rule-engine)

1. Introduction

In this quick tutorial, we're going to integrate Drools with Spring. If you're just getting started with Drools, check out this intro article. (/drools)

2. Maven Dependencies

Let's start by adding the following dependencies to our *pom.xml* file:

```
<dependency>
    <groupId>org.drools</groupId>
    <artifactId>drools-core</artifactId>
    <version>9.44.0.Final</version>
</dependency>
<dependency>
    <groupId>org.kie</groupId>
    <artifactId>kie-spring</artifactId>
    <version>7.74.1.Final</version>
</dependency>
```

The latest versions can be found here for drools-core (https://mvnrepository.com/artifact/org.drools/drools-core) and here for kie-spring (https://mvnrepository.com/artifact/org.kie/kie-spring).

3. Initial Data

Let's now define the data which will be used in our example. We're going to calculate the fare of a ride based on the distance traveled and the night surcharge flag.

Here's a simple object which will be used as a *Fact:*

```
public class TaxiRide {
    private Boolean isNightSurcharge;
    private Long distanceInMile;

    // standard constructors, getters/setters
}
```

Let's also define another business object which will be used for representing fares:

```
public class Fare {
    private Long nightSurcharge;
    private Long rideFare;

    // standard constructors, getters/setters
}
```

Now, let's define a business rule for calculating taxi fares:

```
global com.baeldung.spring.drools.model.Fare rideFare;
dialect  "mvel"

rule "Calculate Taxi Fare - Scenario 1"
    when
        taxiRideInstance:TaxiRide(isNightSurcharge == false &&
distanceInMile < 10);
    then
        rideFare.setNightSurcharge(0);
        rideFare.setRideFare(70);
end
```

As we can see, a rule is defined to calculate the total fare of the given *TaxiRide*.

This rule accepts a *TaxiRide* object and checks if the *isNightSurcharge* attribute is *false* and the *distanceInMile* attribute value is less than 10, then calculate the fare as 70 and sets the *nightSurcharge* property to 0.

The calculated output is set to *Fare* object for further use.

4. Spring Integration

4.1. Spring Bean Configuration

Now, let's move on to the Spring integration.

We're going to define a Spring bean configuration class – which will be responsible for instantiating the *TaxiFareCalculatorService* bean and its dependencies:

```
@Configuration
@ComponentScan("com.baeldung.spring.drools.service")
public class TaxiFareConfiguration {
    private static final String drlFile = "TAXI_FARE_RULE.drl";

    @Bean
    public KieContainer kieContainer() {
        KieServices kieServices = KieServices.Factory.get();

        KieFileSystem kieFileSystem = kieServices.newKieFileSystem();

kieFileSystem.write(ResourceFactory.newClassPathResource(drlFile));
        KieBuilder kieBuilder = kieServices.newKieBuilder(kieFileSystem);
        kieBuilder.buildAll();
        KieModule kieModule = kieBuilder.getKieModule();

        return kieServices.newKieContainer(kieModule.getReleaseId());
    }
}
```

KieServices is a singleton which acts as a single point entry to get all services provided by Kie. *KieServices* is retrieved using *KieServices.Factory.get().*

Next, we need to get the *KieContainer* which is a placeholder for all the object that we need to run the rule engine.

KieContainer is built with the help of other beans including *KieFileSystem*, *KieBuilder*, and *KieModule.*

Let's proceed to create a *KieModule* which is a container of all the resources which are required to define rule knowledge known as *KieBase.*

```
KieModule kieModule = kieBuilder.getKieModule();
```

> *KieBase* is a repository which contains all knowledge related to the application such as rules, processes, functions, type models and it is hidden inside *KieModule*. The *KieBase* can be obtained from the *KieContainer.*

Once *KieModule* is created, we can proceed to create *KieContainer* – which contains the *KieModule* where the *KieBase* has been defined. The *KieContainer* is created using a module:

```
KieContainer kContainer =
kieServices.newKieContainer(kieModule.getReleaseId());
```

4.2. Spring Service

Let's define a service class which executes the actual business logic by passing the *Fact* object to the engine for processing the result:

```
@Service
public class TaxiFareCalculatorService {

    @Autowired
    private KieContainer kieContainer;

    public Long calculateFare(TaxiRide taxiRide, Fare rideFare) {
        KieSession kieSession = kieContainer.newKieSession();
        kieSession.setGlobal("rideFare", rideFare);
        kieSession.insert(taxiRide);
        kieSession.fireAllRules();
        kieSession.dispose();
        return rideFare.getTotalFare();
    }
}
```

Finally, a *KieSession* is created using *KieContainer* instance. A *KieSession* instance is a place where input data can be inserted. The *KieSession* interacts with the engine to process the actual business logic defined in rule based on inserted Facts.

Global (just like a global variable) is used to pass information into the engine. We can set the Global using *setGlobal("key", value);* in this example, we have set *Fare* object as Global to store the calculated taxi fare.

As we discussed in Section 4, **a *Rule* requires data to operate on**. We're inserting the *Fact* into session using *kieSession.insert(taxiRide);*

Once we are done with setting up the input *Fact*, we can request engine to execute the business logic by calling *fireAllRules().*

Finally, we need to clean up the session to avoid memory leak by calling the *dispose()* method.

5. Example in Action

Now, we can wire up a Spring context and see in action that Drools works as expected:

```
@Test
public void
whenNightSurchargeFalseAndDistLessThan10_thenFixWithoutNightSurcharge() {
    TaxiRide taxiRide = new TaxiRide();
    taxiRide.setIsNightSurcharge(false);
    taxiRide.setDistanceInMile(9L);
    Fare rideFare = new Fare();
    Long totalCharge = taxiFareCalculatorService.calculateFare(taxiRide,
rideFare);

    assertNotNull(totalCharge);
    assertEquals(Long.valueOf(70), totalCharge);
}
```

6. Conclusion

In this article, we learned about Drools Spring integration with a simple use case.

As always, the implementation of the example and code snippets are available over on GitHub (https://github.com/eugenp/tutorials/tree/master/spring-drools).

COURSES

ALL COURSES (/COURSES/ALL-COURSES)
BAELDUNG ALL ACCESS (/COURSES/ALL-ACCESS)
BAELDUNG ALL TEAM ACCESS (/COURSES/ALL-ACCESS-TEAM)
THE COURSES PLATFORM (HTTPS://COURSES.BAELDUNG.COM)

SERIES

JAVA “BACK TO BASICS” TUTORIAL (/JAVA-TUTORIAL)
JACKSON JSON SERIES (/JACKSON)
APACHE HTTPCLIENT SERIES (/HTTPCLIENT-SERIES)
REST WITH SPRING SERIES (/REST-WITH-SPRING-SERIES)
SPRING PERSISTENCE SERIES (/PERSISTENCE-WITH-SPRING-SERIES)
SECURITY WITH SPRING (/SECURITY-SPRING)
SPRING REACTIVE SERIES (/SPRING-REACTIVE-SERIES)

ABOUT

ABOUT BAELDUNG (/ABOUT)
THE FULL ARCHIVE (/FULL_ARCHIVE)
EDITORS (/EDITORS)
OUR PARTNERS (/PARTNERS/)
PARTNER WITH BAELDUNG (/PARTNERS/WORK-WITH-US)
EBOOKS (/LIBRARY/)
FAQ (HTTPS://WWW.BAELDUNG.COM/LIBRARY/FAQ)
BAELDUNG PRO (/MEMBERS/)

TERMS OF SERVICE (/TERMS-OF-SERVICE)
PRIVACY POLICY (/PRIVACY-POLICY)