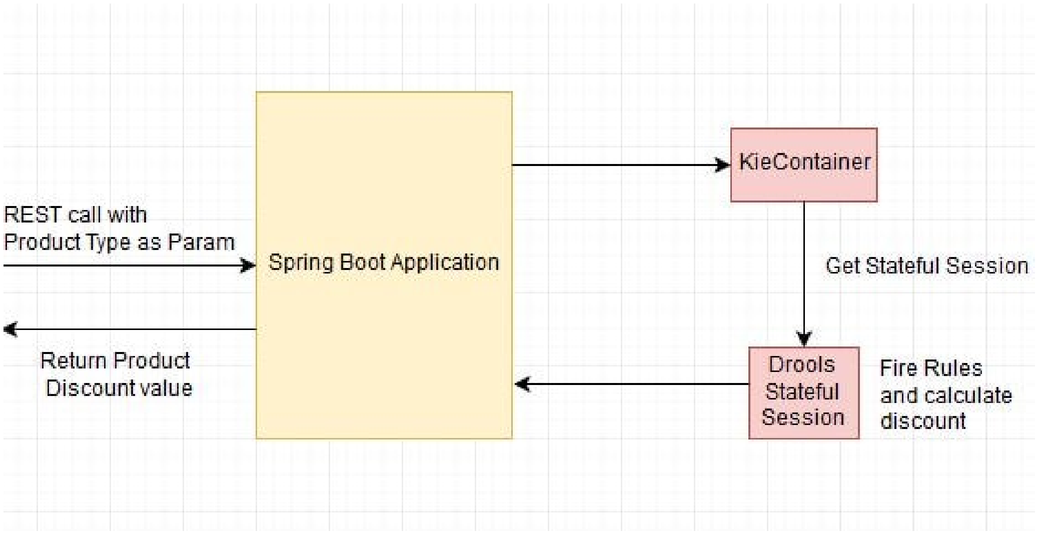
Search Tutorials

Custom Search

Drools Tutorials- Drools Stateful Session + Integration with Spring Boot

Overview

In this tutorial we will create a Spring Boot Application and integrate with JBoss Drools. Like previous drools examples we will create an application for a jewellery shop to calculate discount based on the type of jewellery. We have previously integrated Drools with Spring MVC. (/drools/drools_spring) We will be using Stateful Drools session. We will be exposing a REST API which returns the discount based on the type of product value passed as request parameter.


REST call with Product Type as Param
Spring Boot Application
Return Product Discount value
KieContainer
Get Stateful Session
Drools Stateful Session
Fire Rules and calculate discount

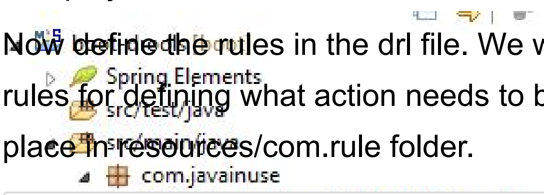
Video

This tutorial is explained in the below Youtube Video.

Define the model class Product.java as follows-

```
package com.javainuse.model;

public class Product {

        private String type;
        private int discount;

        public String getType() {
                return type;
        }

        public void setType(String type) {
                this.type = type;
        }

        public int getDiscount() {
                return discount;
        }

        public void setDiscount(int discount) {
                this.discount = discount;
        }

}
```

The project structure is as follows-



Now define the rules in the drl file. We will use the type property of the Product class for defining the rules for defining what action needs to be taken if a particular condition is met. The .drl file should be place in resources/com.rule folder.

```
package rules

import com.javainuse.model.Product

rule "Offer for Diamond"
        when
                productObject: Product(type=="diamond")
        then
                productObject.setDiscount(15);
        end
rule "Offer for Gold"
        when
                productObject: Product(type=="gold")
        then
                productObject.setDiscount(25);
        end
```

The Drools 6.0 project consists of a meta-data file META-INF/kmodule.xml. The file is located under the source folder as shown in below snapshot. A named session ksession-rule is created. The ksession rule is applicable for all drl file contained in the pacakge rules. Currently we have only one drl file of package rules.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001
        xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.or
        <modelVersion>4.0.0</modelVersion>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<kmodule xmlns="http://jboss.org/kie/6.0.0/kmodule">
   <kbase name="rules" packages="rules">
        <ksession name="rulesSession"/>
    </kbase>
</kmodule>
```

```
        <name>boot-drools</name>
        <description>Demo project for Spring Boot Drools</description>
```

Define the JewelleryShopController as follows to expose the GET REST API.

```
package com.javainuse.controller;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RequestParam;
import org.springframework.web.bind.annotation.RestController;

import com.javainuse.model.Product;
import com.javainuse.service.JewelleryShopService;

@RestController
public class JewelleryShopController {

        private final JewelleryShopService jewelleryShopService;

        @Autowired
        public JewelleryShopController(JewelleryShopService jewelleryShopService) {
                this.jewelleryShopService = jewelleryShopService;
        }

        @RequestMapping(value = "/getDiscount", method = RequestMethod.GET, produces
        public Product getQuestions(@RequestParam(required = true) String type) {

                Product product = new Product();
                product.setType(type);
                jewelleryShopService.getProductDiscount(product);
                return product;
        }
}
```

```
        </dependencies>
        <build>
```

```
                <plugins>
                        <plugin>
                                <groupId>org.springframework.boot</groupId>
```

Define the JewelleryShopService as follows to get the Stateful session and execute the drools rules.

```
package com.javainuse.service;

import org.kie.api.runtime.KieContainer;
import org.kie.api.runtime.KieSession;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.javainuse.model.Product;

@Service
public class JewelleryShopService {

        private final KieContainer kieContainer;

        @Autowired
        public JewelleryShopService(KieContainer kieContainer) {
                this.kieContainer = kieContainer;
        }

        public Product getProductDiscount(Product product) {
                //get the stateful session
                KieSession kieSession = kieContainer.newKieSession("rulesSession");
                kieSession.insert(product);
                kieSession.fireAllRules();
                kieSession.dispose();
                return product;
        }
}
```

Finally create the Main class with SpringBootApplication annotation. Also create a bean of type of KieContainer using JavaConfig

```
package com.javainuse;

import org.kie.api.KieServices;
import org.kie.api.runtime.KieContainer;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.context.annotation.Bean;

@SpringBootApplication
public class SpringBootDroolsHelloWorldApp {

        public static void main(String[] args) {
                SpringApplication.run(SpringBootDroolsHelloWorldApp.class, args);

        }

        @Bean
        public KieContainer kieContainer() {
                return KieServices.Factory.get().getKieClasspathContainer();
        }

}
```

We are done with the code changes, now start the application. Open a browser and hit the url- **http://localhost:8080/getDiscount?type=gold**. We get the output with the discount value as follows-

Download Source Code

Download it - Drools Spring Boot Integration example (/zip/spring/boot/boot-drools.rar)

Popular Posts

- Spring Boot +JSON Web Token(JWT) Hello World Example (/spring/boot-jwt)
- Angular 7 + Spring Boot Application Hello World Example (/spring/ang7-hello)

- Build a Real Time Chat Application using Spring Boot + WebSocket + RabbitMQ (/spring/boot-websocket-chat)
- Pivotal Cloud Foundry Tutorial - Deploy Spring Boot Application Hello World Example (/pcf/pcf-hello)
- Deploying Spring Based WAR Application to Docker (/devOps/docker/docker-war)
- EIP patterns using Apache Camel (/camel/camel_EIP)
- Spring Cloud- Netflix Eureka + Ribbon Simple Example (/spring/spring_ribbon)
- Spring Cloud- Netflix Hystrix Circuit Breaker Simple Example (/spring/spring_hystrix_circuitbreaker)
- Spring Boot + Swagger Example Hello World Example (/spring/boot_swagger)
- Spring Boot Batch Simple example (/spring/bootbatch)
- Spring Boot + Apache Kafka Example (/spring/spring-boot-apache-kafka-hello-world)
- Spring Boot Admin Simple Example (/spring/boot-admin)
- Spring Boot Security - Introduction to OAuth (/spring/spring-boot-oauth-introduction)
- Spring Boot OAuth2 Part 1 - Getting The Authorization Code (/spring/spring-boot-oauth-authorization-code)
- Spring Boot OAuth2 Part 2 - Getting The Access Token And Using it to Fetch Data. (/spring/spring-boot-oauth-access-token)
- JBoss Drools Hello World-Stateful Knowledge Session using KieSession (/drools_hello_kie)
- Understand Drools Stateful vs Stateless Knowledge Session (/drools_states)
- JBoss Drools- Understanding Drools Decision Table using Simple Example (/drools/drools_decision)

6 Advantages of Microservices
White Paper by Hazelcast.
Getting Started with Microservices Using Hazelcast IMDG and Spring Boot
LEARN MC

See Also

- Spring Boot Interview Questions (/spring/SpringBootInterviewQuestions)
- Spring Batch Interview Questions (/spring/sprbatch_interview)
- Spring AOP Interview Questions (/spring/spring-AOP-interview-quesions)
- Angular 2 Interview Questions (/angular/ang2_intvw)
- Apache Camel Interview Questions (/camel/Apache_Camel_Questions)
- JBoss Fuse Interview Questions (/camel/JBoss_Fuse_Questions)
- Drools Interview Questions (/drools/drools_intvw)
- Java 8 Interview Questions (/java/java8_intvw)
- Spring Cloud Interview Questions (/spring/spring-cloud-interview-questions)
- Microservices Interview Questions (/spring/microservices-interview-quesions)
- Java HashMap and ConcurrentHashMap Interview Questions (/java/java_map_intvw)
- Mule ESB frequently asked interview questions (/misc/muleintvw)
- Apache Kafka Interview Questions (/misc/apache-kafka-interview-questions)
- Tosca Testing Tool Interview Questions (/misc/tosca-testing-tool-interview-questions)
- Top Maven Build Tool Interview Questions (/misc/maven-interview-questions)
- Top Gradle Build Tool Interview Questions (/misc/gradle-interview-questions)

- Miscellaneous Topics (/misc)

NOW
PLAYING