**Lab Exercise 18- Calendar Attribute in Drools**

The **Calendar** attribute in Drools allows you to specify the days or times when rules are allowed to fire. It helps in controlling when rules can be executed based on a calendar configuration, which can be useful for scheduling rule execution only on specific days or during specific time ranges (e.g., business hours).

**Objective:**

* Learn how to use the **Calendar** attribute in Drools.
* Set up a rule that only executes during a specific time window (e.g., business hours or weekdays).
* Implement a scenario to demonstrate the calendar-based rule execution.

**Scenario:**

In this lab, we will simulate a system where certain actions can only be taken during **business hours** (e.g., 9 AM to 5 PM, Monday to Friday). The rule should only fire during this period. Outside of business hours, the rule will not execute.

**Step 1: Define Java Model Class (Order.java)**

We’ll reuse the Order.java class to represent an order, and rules will execute based on the calendar configuration (e.g., during business hours).

package com.example.model;

public class Order {

private String id;

private boolean processed;

public Order(String id) {

this.id = id;

this.processed = false;

}

public String getId() {

return id;

}

public boolean isProcessed() {

return processed;

}

public void setProcessed(boolean processed) {

this.processed = processed;

}

@Override

public String toString() {

return "Order{id='" + id + "', processed=" + processed + '}';

}

}

**Step 2: Define Drools Rules (orderProcessingCalendarRules.drl)**

In this rule file, we use the **calendar** attribute to define when the rules can fire. For example, we can allow rule execution only during weekdays (Monday to Friday, 9 AM to 5 PM).

package com.example.rules;

import com.example.model.Order;

rule "Process Order During Business Hours"

calendars "businessHours"

when

$order : Order(processed == false)

then

$order.setProcessed(true);

System.out.println("Order " + $order.getId() + " processed during business hours.");

end

**Step 3: Define kmodule.xml**

The kmodule.xml file is used to define the knowledge base and session configurations. Ensure the calendar configuration is included:

<?xml version="1.0" encoding="UTF-8"?>

<kmodule xmlns="http://jboss.org/kie/6.0.0/kmodule">

<kbase name="orderProcessingKBase" packages="com.example.rules">

<ksession name="orderKSession" type="stateful"/>

</kbase>

</kmodule>

**Step 4: Define the Calendar in the Test Class (OrderProcessingCalendarTest.java)**

We will now implement the test class where we configure the **calendar** to simulate business hours (9 AM to 5 PM, Monday to Friday). The rule will only fire during this period.

package com.example.model;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

import java.util.HashMap;

import java.util.Map;

public class OrderProcessingCalendarTest {

public static void main(String[] args) throws InterruptedException {

// Load KieServices and KieContainer

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

// Create a stateful Kie session

KieSession kSession = kContainer.newKieSession("orderKSession");

// Define the business hours (Monday to Friday, 9 AM to 5 PM)

Map<String, org.kie.api.time.Calendar> calendars = new HashMap<>();

calendars.put("businessHours", (timestamp) -> {

java.util.Calendar cal = java.util.Calendar.getInstance();

cal.setTimeInMillis(timestamp);

int dayOfWeek = cal.get(java.util.Calendar.DAY\_OF\_WEEK);

int hourOfDay = cal.get(java.util.Calendar.HOUR\_OF\_DAY);

// Allow execution Monday to Friday, 9 AM to 5 PM

boolean isWeekday = dayOfWeek >= java.util.Calendar.MONDAY && dayOfWeek <= java.util.Calendar.FRIDAY;

boolean isBusinessHours = hourOfDay >= 9 && hourOfDay <= 17;

return isWeekday && isBusinessHours;

});

// Register the calendar with the KieSession

kSession.getCalendars().set("businessHours", calendars.get("businessHours"));

// Create an order and insert it into the session

Order order = new Order("Order-001");

kSession.insert(order);

// Fire all rules

kSession.fireAllRules();

// Dispose of the session

kSession.dispose();

}

}

**Explanation:**

1. **Calendar Configuration**:
   * The calendar is defined using a lambda function that allows rule execution only on weekdays (Monday to Friday) between 9 AM and 5 PM.
   * cal.get(Calendar.DAY\_OF\_WEEK) checks if the current day is a weekday, and cal.get(Calendar.HOUR\_OF\_DAY) checks if the time is within business hours.
2. **Calendar in the Rule**:
   * In the rule, we use the calendars attribute (calendars "businessHours") to restrict rule execution to only when the calendar condition is satisfied (i.e., during business hours).
3. **Stateful Session**:
   * A **stateful session** is used to keep track of the facts (orders) in memory. When the session is fired, the calendar determines if the rule can execute or not.

**Step 5: Run the Application**

When you run the OrderProcessingCalendarTest.java class, the rule will only fire if the current time is during the business hours defined in the calendar.

**Expected Output:**

If you run the application during business hours (Monday to Friday, 9 AM to 5 PM), the output will be:

Order Order-001 processed during business hours.

If you run the application outside of business hours (e.g., on the weekend or at night), the rule will not fire, and there will be no output.

**Key Points:**

* **Calendar Attribute**: The calendar attribute in Drools restricts when a rule can fire based on a custom calendar condition. In this example, the rule only fires during business hours.
* **Stateful Session**: The stateful session holds the order in memory, and the calendar controls when the rule can execute.

This lab exercise demonstrates how to use the **Calendar** attribute in Drools to schedule rules to execute only during specific time periods, such as business hours.

4o