Learn Drools: Part 1

When we implement a complex software often we require to maintains a set of rules which will be applied to set of data to take action on them. In a regular term, we called them Rule Engine.

If there is a small set of rules we can create our own Rule Engine which will maintain a certain order and applied on incoming data to take the decision or categorize the data.

The advantage of maintaining own rule engine is that we have a pure control over the Rule Engine's algorithm so we can easily change the Algorithm logic , To be simple we don’t have to rely on the third party for the pattern matching logic.

On Other hand, if rules are ever changing or there is a huge set of rules probably we will not want to maintain our own Rule engine as it increases our development cost not only that how the Rule engine performs and it’s optimization, Who wants to take this responsibility?

It would be nice if we delegate that work to some third party which is tested and trusted. So we can clearly separate the data and logic

Third Party maintains the Rule Engine application and we just define the Rules as a strategy, on top of it, it would be nice if it is declarative so Business Analyst can understand the logic.

**Drools:**

Drools is this kind of Business Logic integration Platform (BLiP). Drools is Open source project written in java, Red Hat and Jboss maintains Drools.

Drools has two main parts

**Authoring** : By Authoring we create Rules file for Drools this is (.drl ) the file which contains the Rule definition in a declarative way. In this file ,we can write a set of Rules which will be fired at the run time. It is Developer's responsibility to write these Rules as per business requirements.

2. **Runtime** : By Runtime we create working memory , It is same as Session in Hibernate. As Rules file contains a set of rules, Runtime creating memory load these rules and apply on the incoming data , In drools, we called incoming data as facts.

**Rules :**

A rule is nothing but the logic which will be applied on incoming data. It has two main parts when and then.

**When** : It determines the condition,on which condition the Rule will be fired.

**Then** : It is the action If rules met the condition, its defines what work this rule performs

Syntax :

***rule <Rule Name>***

***when***

***<condition>***

***then***

***<Action>***

***End***

**Problem : We will greet a Person based on current time.**

We will define the rules in Drools files , Drool will load these rules and fire on the incoming data.

**Step 1.** Create a .drl (droolRule.drl)file where we will define the rules.

package com.rules

import com.example.droolsExample.Person

rule "Good Morning"

when

person: Person(time >= 0 , time < 12)

then

person.setGreet("Good Morning " + person.getName());

end

rule "Good Afternoon"

when

person: Person(time >= 12 , time < 16)

then

person.setGreet("Good Afternoon " + person.getName());

end

rule "Good Night"

when

person: Person(time >= 16 , time <= 24)

then

person.setGreet("Good Night " + person.getName());

end

Please note that here we create three rules “Good Morning”, “Good Afternoon “ and “Good Night.” In when section we check the current time based on the Person POJO’s time property . and in then part we set the greet message accordingly.

**Step 2:** Create Person pojo class

package com.example.droolsExample;

public class Person {

private String name;

private int time;

private String greet;

public String getGreet() {

return greet;

}

public void setGreet(String greet) {

this.greet = greet;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public int getTime() {

return time;

}

public void setTime(int time) {

this.time = time;

}

}

**Step 3:** We create a class named **DroolsTest.java**, In this class, we take some steps.

Step a : Load the rule file i.e **droolsTest.drl** by using InputStream.

Step b: create a package using above rule and add them into drools **PackageBuilder**.

Step c: Create a **RuleBase** by using above Package. **Rulebase** is same as **Sessionfactory** in Hibernate it is costly.

Step d: Create a Working memory from this **RuleBase**. It is same as Session class in Hibernate. This Working memory manages the rules and incoming data. Apply the rules on the data.

Step e: Add incoming data into Working memory here we create a Person Object and add it into **Working Memory**

Step f: Fire all rules.

**DroolsTest.java**

package com.example.droolsExample;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.Reader;

import org.drools.compiler.compiler.DroolsParserException;

import org.drools.compiler.compiler.PackageBuilder;

import org.drools.core.RuleBase;

import org.drools.core.RuleBaseFactory;

import org.drools.core.WorkingMemory;

public class DroolsTest {

public static void main(String[] args) throws DroolsParserException,

IOException {

DroolsTest droolsTest = new DroolsTest();

droolsTest.executeDrools();

}

public void executeDrools() throws DroolsParserException, IOException {

PackageBuilder packageBuilder = new PackageBuilder();

String ruleFile = "/com/rules/droolsRule.drl";

InputStream resourceAsStream = getClass().getResourceAsStream(ruleFile);

Reader reader = new InputStreamReader(resourceAsStream);

packageBuilder.addPackageFromDrl(reader);

org.drools.core.rule.Package rulesPackage = packageBuilder.getPackage();

RuleBase ruleBase = RuleBaseFactory.newRuleBase();

ruleBase.addPackage(rulesPackage);

WorkingMemory workingMemory = ruleBase.newStatefulSession();

Person person = new Person();

person.setName("Shamik Mitra");

person.setTime(7);

workingMemory.insert(person);

workingMemory.fireAllRules();

System.out.println(person.getGreet());

}

}

**Output:**

**Good Morning Shamik Mitra,**

As we set the time 7 so it satisfies the Good Morning Rule condition and fire this rule.