Learn Drools:Part V(Truth Maintenance)

In our previous article, we see how Inference is useful when we need to repetitively use a condition across the rules.

By Inference we insert the condition as a POJO and use that fact across the rules. A perfect way to separate the condition from its usage.

But think about a condition where two rules are mutually exclusive in that scenario it is far better to use Truth maintenance which is nothing but a truth table.

Let understand it by an example.

***Suppose , For the IT department if the designation is a Manager we give him laptop but if an employee is not a Manager we give him a desktop.***

So in drools we have to create two rules, first, rule checks if the designation is a manager then give him laptop while other rule checks if the designation is not a manager then give him desktop.

Furthermore, these two rules are mutually exclusive. For an employee/fact, only one rule can be true not both. So we can maintain a truth table for that , in Drools we called it **Drools Truth Maintenance System (TMS)**, by TMS we can logically insert the condition ,note that not inserted but logically inserted. The advantage of logically insert is that when the condition is false the fact/POJO we insert in drools Knowledge Base automatically retracted.

To understand the advantage of TMS, think a scenario say an employee current designation is not a manager so according to rule he got Desktop and we logically insert the Not a manager condition in the Drools Knowledge base for that employee or fact but after some days he got promoted as a manager now the fact/condition we inserted in Drools Knowledgebase for this employee has to be retracted and new fact/condition is Manager has to be Inserted and now he is eligible for Laptop.

Using TMS it will be done automatically we don’t have to care about this things.

Find the solution to the above problem statement.

**employee.drl**

package com.rules

import com.example.droolsExample.pojo.Employee

import com.example.droolsExample.pojo.Department

import com.example.droolsExample.pojo.ITManager

import com.example.droolsExample.pojo.ITEmployee

rule "IT Manager Inference1"

when

$dept: Department(name=="IT");

$emp: Employee(dept == $dept,manager==true);

then

insertLogical(new ITManager($emp));

end

rule "IT Employee Inference1"

when

$dept: Department(name=="IT");

$emp: Employee(dept == $dept,manager==false);

then

insertLogical(new ITEmployee($emp));

end

rule "give Manager Laptop"

when

$emp: Employee();

$itManager: ITManager(emp == $emp);

then

$emp.setMessage("Give Laptop");

System.out.println($emp.getName()+ ": "+$emp.getDept().getName()+ ":"+$emp.getMessage());

end

rule "give Employee Desktop"

when

$emp: Employee();

$itEmployee: ITEmployee(emp == $emp);

then

$emp.setMessage("Give Desktop");

System.out.println($emp.getName()+ ": "+$emp.getDept().getName()+ ":"+$emp.getMessage());

end

**Employee.java**

package com.example.droolsExample.pojo;

public class Employee {

String name;

boolean manager;

String message;

Department dept;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public boolean isManager() {

return manager;

}

public void setManager(boolean manager) {

this.manager = manager;

}

public String getMessage() {

return message;

}

public void setMessage(String message) {

this.message = message;

}

public Department getDept() {

return dept;

}

public void setDept(Department dept) {

this.dept = dept;

}

@Override

public String toString() {

return "Employee [name=" + name + ", manager=" + manager + ", message="

+ message + ", dept=" + dept + "]";

}

}

**Department.java**

package com.example.droolsExample.pojo;

public class Department {

String name;

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

}

**ITManager.java**

package com.example.droolsExample.pojo;

public class ITManager {

private Employee emp;

public ITManager(Employee emp)

{

this.emp=emp;

}

public Employee getEmp() {

return emp;

}

public void setEmp(Employee emp) {

this.emp = emp;

}

}

**ITEmployee.java**

package com.example.droolsExample.pojo;

public class ITEmployee {

private Employee emp;

public ITEmployee(Employee emp)

{

this.emp=emp;

}

public Employee getEmp() {

return emp;

}

public void setEmp(Employee emp) {

this.emp = emp;

}

}

**DroolTest.java**

package com.example.droolsExample;

import java.io.IOException;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.Reader;

import java.util.List;

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;

import org.drools.core.event.AgendaEventListener;

import org.kie.api.runtime.process.ProcessInstance;

import org.kie.api.runtime.rule.FactHandle;

import com.example.drools.listener.Activation;

import com.example.drools.listener.TrackingAgendaEventListener;

import com.example.droolsExample.pojo.Department;

import com.example.droolsExample.pojo.Employee;

public class DroolsTest {

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

IOException {

DroolsTest droolsTest = new DroolsTest();

//droolsTest.executeDrools();

droolsTest.executeDroolsEmployee("/com/rules/employeeTruthTable.drl");

}

public void executeDroolsEmployee(String ruleFile) throws DroolsParserException, IOException {

PackageBuilder packageBuilder = new PackageBuilder();

//String ruleFile = "/com/rules/employee.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();

Department dep = new Department();

dep.setName("Civil");

Department dep1 = new Department();

dep1.setName("IT");

Employee emp = new Employee();

emp.setName("Shamik Mitra");

emp.setManager(true);

emp.setDept(dep1);

Employee emp2 = new Employee();

emp2.setName("Swastika Mitra");

emp2.setManager(false);

emp2.setDept(dep1);

Employee emp1 = new Employee();

emp1.setName("Samir Mitra");

emp1.setManager(true);

emp1.setDept(dep);

workingMemory.insert(dep);

workingMemory.insert(dep1);

workingMemory.insert(emp);

workingMemory.insert(emp1);

workingMemory.insert(emp2);

workingMemory.fireAllRules();

}

**Output:**

Shamik Mitra: IT:Give Laptop

Swastika Mitra: IT:Give Desktop