DROOLS - SAMPLE DROOLS PROGRAM

In this chapter, we will create a Drools project for the following problem statement:

> *Depending upon the city and the kind of product CombinationofCityandProduct, find out the local tax related to that city.*

We will have two DRL files for our Drools project. The two DRL files will signify two cities in consideration *PuneandNagpur* and four types of products *groceries, medicines, watches, andluxurygoods*.

- The tax on medicines in both the cities is considered as zero.
- For groceries, we have assumed a tax of Rs 2 in Pune and Rs 1 in Nagpur.

We have used the same selling price to demonstrate different outputs. Note that all the rules are getting fired in the application.

Here is the model to hold each itemType:

```
package com.sample;

import java.math.BigDecimal;

public class ItemCity {
   public enum City {
      PUNE, NAGPUR
   }

   public enum Type {
      GROCERIES, MEDICINES, WATCHES, LUXURYGOODS
   }

   private City purchaseCity;
   private BigDecimal sellPrice;
   private Type typeofItem;
   private BigDecimal localTax;

   public City getPurchaseCity() {
      return purchaseCity;
   }

   public void setPurchaseCity(City purchaseCity) {
      this.purchaseCity = purchaseCity;
   }

   public BigDecimal getSellPrice() {
      return sellPrice;
   }

   public void setSellPrice(BigDecimal sellPrice) {
      this.sellPrice = sellPrice;
   }

   public Type getTypeofItem() {
      return typeofItem;
   }

   public void setTypeofItem(Type typeofItem) {
      this.typeofItem = typeofItem;
   }

   public BigDecimal getLocalTax() {
```

```
        return localTax;
    }

    public void setLocalTax(BigDecimal localTax) {
        this.localTax = localTax;
    }
}
```

DRL Files

As suggested earlier, we have used two DRL files here: Pune.drl and Nagpur.drl.

Pune.drl

This is the DRL file that executes rules for Pune city.

```
// created on: Dec 24, 2014
package droolsexample

// list any import classes here.
import com.sample.ItemCity;
import java.math.BigDecimal;

// declare any global variables here
dialect "java"
rule "Pune Medicine Item"

    when
        item : ItemCity (purchaseCity == ItemCity.City.PUNE,
                         typeofItem == ItemCity.Type.MEDICINES)

    then
        BigDecimal tax = new BigDecimal(0.0);
        item.setLocalTax(tax.multiply(item.getSellPrice()));
end

rule "Pune Groceries Item"

    when
        item : ItemCity(purchaseCity == ItemCity.City.PUNE,
                        typeofItem == ItemCity.Type.GROCERIES)

    then
        BigDecimal tax = new BigDecimal(2.0);
        item.setLocalTax(tax.multiply(item.getSellPrice()));
end
```

Nagpur.drl

This is the DRL file that executes rules for Nagpur city.

```
// created on: Dec 26, 2014
package droolsexample

// list any import classes here.
import com.sample.ItemCity;
import java.math.BigDecimal;

// declare any global variables here
dialect "java"
rule "Nagpur Medicine Item"

    when
        item : ItemCity(purchaseCity == ItemCity.City.NAGPUR,
                        typeofItem == ItemCity.Type.MEDICINES)

    then
```

```
      BigDecimal tax = new BigDecimal(0.0);
      item.setLocalTax(tax.multiply(item.getSellPrice()));
end

rule "Nagpur Groceries Item"

   when
      item : ItemCity(purchaseCity == ItemCity.City.NAGPUR,
                      typeofItem == ItemCity.Type.GROCERIES)

   then
      BigDecimal tax = new BigDecimal(1.0);
      item.setLocalTax(tax.multiply(item.getSellPrice()));
end
```

We have written the DRL files based on city, as it gives us extensibility to add any number of rule files later if new cities are being added.

To demonstrate that all the rules are getting triggered from our rule files, we have used two item types *medicinesandgroceries*; and medicine is tax-free and groceries are taxed as per the city.

Our test class loads the rule files, inserts the facts into the session, and produces the output.

Droolstest.java

```
package com.sample;

import java.math.BigDecimal;

import org.drools.KnowledgeBase;
import org.drools.KnowledgeBaseFactory;

import org.drools.builder.KnowledgeBuilder;
import org.drools.builder.KnowledgeBuilderError;
import org.drools.builder.KnowledgeBuilderErrors;
import org.drools.builder.KnowledgeBuilderFactory;
import org.drools.builder.ResourceType;

import org.drools.io.ResourceFactory;
import org.drools.runtime.StatefulKnowledgeSession;

import com.sample.ItemCity.City;
import com.sample.ItemCity.Type;

/* This is a sample class to launch a rule. */

public class DroolsTest {
   public static final void main(String[] args) {
      try {

         // load up the knowledge base
         KnowledgeBase kbase = readKnowledgeBase();
         StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();

         ItemCity item1 = new ItemCity();
         item1.setPurchaseCity(City.PUNE);
         item1.setTypeofItem(Type.MEDICINES);
         item1.setSellPrice(new BigDecimal(10));
         ksession.insert(item1);

         ItemCity item2 = new ItemCity();
         item2.setPurchaseCity(City.PUNE);
         item2.setTypeofItem(Type.GROCERIES);
         item2.setSellPrice(new BigDecimal(10));
         ksession.insert(item2);

         ItemCity item3 = new ItemCity();
         item3.setPurchaseCity(City.NAGPUR);
```

```
            item3.setTypeofItem(Type.MEDICINES);
            item3.setSellPrice(new BigDecimal(10));
            ksession.insert(item3);

            ItemCity item4 = new ItemCity();
            item4.setPurchaseCity(City.NAGPUR);
            item4.setTypeofItem(Type.GROCERIES);
            item4.setSellPrice(new BigDecimal(10));
            ksession.insert(item4);

            ksession.fireAllRules();

            System.out.println(item1.getPurchaseCity().toString() + " "
               + item1.getLocalTax().intValue());

            System.out.println(item2.getPurchaseCity().toString() + " "
               + item2.getLocalTax().intValue());

            System.out.println(item3.getPurchaseCity().toString() + " "
               + item3.getLocalTax().intValue());

            System.out.println(item4.getPurchaseCity().toString() + " "
               + item4.getLocalTax().intValue());

         } catch (Throwable t) {
            t.printStackTrace();
         }
      }

      private static KnowledgeBase readKnowledgeBase() throws Exception {

         KnowledgeBuilder kbuilder = KnowledgeBuilderFactory.newKnowledgeBuilder();

         kbuilder.add(ResourceFactory.newClassPathResource("Pune.drl"), ResourceType.DRL);
         kbuilder.add(ResourceFactory.newClassPathResource("Nagpur.drl"), ResourceType.DRL);

         KnowledgeBuilderErrors errors = kbuilder.getErrors();

         if (errors.size() > 0) {
            for (KnowledgeBuilderError error: errors) {
               System.err.println(error);
            }
            throw new IllegalArgumentException("Could not parse knowledge.");
         }

         KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();
         kbase.addKnowledgePackages(kbuilder.getKnowledgePackages());

         return kbase;
      }
   }
```

If you run this program, its output would be as follows:

```
PUNE 0
PUNE 20
NAGPUR 0
NAGPUR 10
```

For both Pune and Nagpur, when the item is a medicine, the local tax is zero; whereas when the item is a grocery product, the tax is as per the city. More rules can be added in the DRL files for other products. This is just a sample program.

Call an External Function form a DRL File

Here we will demonstrate how to call a static function from a Java file within your DRL file.

First of all, create a class **HelloCity.java** in the same package **com.sample**.

```
package com.sample;

public class HelloCity {
   public static void writeHello(String name) {
      System.out.println("HELLO " + name + "!!!!!!");
   }
}
```

Thereafter, add the import statement in the DRL file to call the writeHello method from the DRL file. In the following code block, the changes in the DRL file Pune.drl are highlighted in yellow.

```
// created on: Dec 24, 2014
package droolsexample

// list any import classes here.
import com.sample.ItemCity;
import java.math.BigDecimal;

import com.sample.HelloCity;

//declare any global variables here
dialect "java"

rule "Pune Medicine Item"

   when
      item : ItemCity(purchaseCity == ItemCity.City.PUNE, typeofItem ==
ItemCity.Type.MEDICINES)

   then
      BigDecimal tax = new BigDecimal(0.0);
      item.setLocalTax(tax.multiply(item.getSellPrice()));
      HelloCity.writeHello(item.getPurchaseCity().toString());
end

rule "Pune Groceries Item"

   when
      item : ItemCity(purchaseCity == ItemCity.City.PUNE, typeofItem ==
ItemCity.Type.GROCERIES)

   then
      BigDecimal tax = new BigDecimal(2.0);
      item.setLocalTax(tax.multiply(item.getSellPrice()));
end
```

Run the program again and its output would be as follows:

```
HELLO PUNE!!!!!!
PUNE 0
PUNE 20
NAGPUR 0
NAGPUR 10
```

The difference now in the output is marked in yellow which shows the output of the static method in the Java class.

The advantage to call a Java method is that we can write any utility/helper function in Java and call the same from a DRL file.