# Drools Backward Chaining Simple Example

Overview

In [previous chapter](https://www.javainuse.com/drools_states) we implemented a drools project to understand difference between Stateful and Stateless session. In this tutorial we will try to understand what is Backward chaining using example.

What is Forward and Backward chaining?

Chaining is the process of performing a sequence of steps in a particular order.

Forward Chaining-

We first analyze the data in a particular sequence. Based on the results of the data we arrive at a decision. For example if want to find out for a particular test the five students out of 100 who came first. Here we will take the data and then based on the analysis decide the results. This is forward chaining

Backward Chaining-

We first take a decision and then check if the decision is true or no by backtracking through sequence of events. For example if i want to find out if a particular student has passed or not? Then i will take a decision that the student has passed. Then analyze the data by backtracking through the sequence of analysis of data. Accordingly its decided if the decision is correct or not.  
  
Based on requirement either of the two apporaches can be used. Sometimes the combination of both forward and backward chaining is also used. Backward chaining is often referred to as derivation queries and drools implements it with query construct.

Lets Begin

For our example here we will insert facts related to location of Eiffel tower and then verify if some of our decisions are correct.  
Create the location class as follows-

package com.javainuse.model;

import org.kie.api.definition.type.Position;

public class Location {

@Position(0)

private String entity;

@Position(1)

private String location;

public Location(String entity, String location) {

this.entity = entity;

this.location = location;

}

public String getEntity() {

return entity;

}

public void setEntity(String entity) {

this.entity = entity;

}

public String getLocation() {

return location;

}

public void setLocation(String location) {

this.location = location;

}

@Override

public boolean equals(Object o) {

if (this == o) {

return true;

}

if (o == null || getClass() != o.getClass()) {

return false;

}

Location location1 = (Location) o;

if (entity != null ? !entity.equals(location1.entity)

: location1.entity != null) {

return false;

}

if (location != null ? !location.equals(location1.location)

: location1.location != null) {

return false;

}

return true;

}

@Override

public int hashCode() {

int result = entity != null ? entity.hashCode() : 0;

result = 31 \* result + (location != null ? location.hashCode() : 0);

return result;

}

@Override

public String toString() {

return "Location{" + "entity='" + entity + '\'' + ", location='"

+ location + '\'' + '}';

}

}

In rules.drl we use the query construct. Recursion is an important part of derivation queries and allows for search of trees.  
The rules.drl file will be as follows-

package rules

import com.javainuse.model.Location;

dialect "mvel"

query isContainedIn( String x, String y )

Location( x, y; )

or

( Location( z, y; ) and isContainedIn( x, z; ) )

end

rule "check decision"

when

isContainedIn("Eiffel tower", "World"; )

then

System.out.println( "Decision taken--Eiffel tower is in the World" );

System.out.println( "-----------------------------" );

System.out.println( "-----------------------------" );

System.out.println( "-----------All Facts-----------------" );

end

rule "check decision false"

when

not isContainedIn("Eiffel tower", "World"; )

then

System.out.println( "Decision taken--Eiffel tower is not in the World" );

System.out.println( "-----------------------------" );

System.out.println( "-----------------------------" );

System.out.println( "-----------All Facts-----------------" );

end

rule "get all facts"

when

isContainedIn(entity, location; )

then

System.out.println( "Entity " + entity + " is in " + location );

end

Finally we define DroolsTest class. Here load the facts and the rules in the drools working memory and firing all the rules. We make use of stateful session for firing the rules.

package com.javainuse.main;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

import com.javainuse.model.Location;

public class DroolsTest {

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

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

KieSession ksession = kContainer.newKieSession("ksession-rule");

ksession.insert(new Location("Europe", "World"));

**// ksession.insert(new Location("France", "Europe"));**

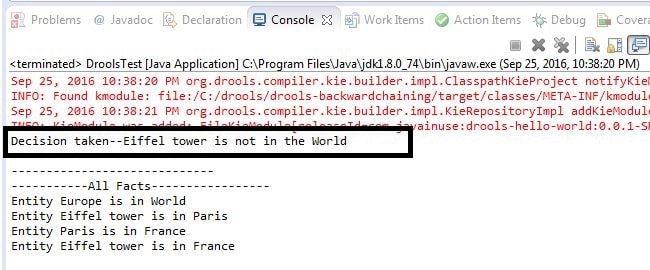
ksession.insert(new Location("Paris", "France"));

ksession.insert(new Location("Eiffel tower", "Paris"));

ksession.fireAllRules();

}

}

On running the DroolsTest class as a java application we get the output as-  


In rules.drl we have 2 decisions to make-

* Eiffel Tower is located in the world.
* Eiffel Tower is not located in the world.

As we have commented a fact that France is in Europe, so in our facts we have Eiffel Tower is in Paris and France but not in the world.  
So our decisions

* Eiffel Tower is located in the world is false
* Eiffel Tower is not located in the world is true

We will now modify the DroolsTest class by uncommenting the fact that France is in Europe.

package com.javainuse.main;

import org.kie.api.KieServices;

import org.kie.api.runtime.KieContainer;

import org.kie.api.runtime.KieSession;

import com.javainuse.model.Location;

public class DroolsTest {

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

KieServices ks = KieServices.Factory.get();

KieContainer kContainer = ks.getKieClasspathContainer();

KieSession ksession = kContainer.newKieSession("ksession-rule");

ksession.insert(new Location("Europe", "World"));

**ksession.insert(new Location("France", "Europe"));**

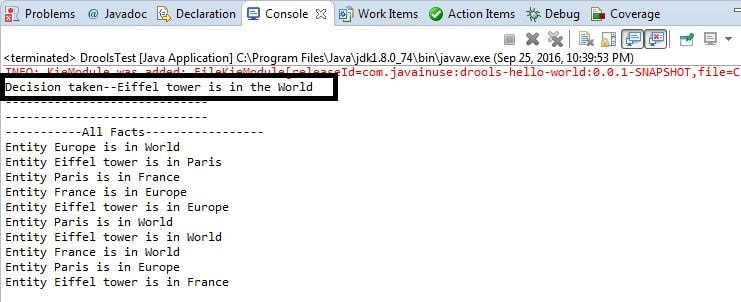
ksession.insert(new Location("Paris", "France"));

ksession.insert(new Location("Eiffel tower", "Paris"));

ksession.fireAllRules();

}

}

On running the DroolsTest class as a java application we get the output as-  


So our decisions

* Eiffel Tower is located in the world is true
* Eiffel Tower is not located in the world is false