**Drools**

**What is Drools?**

* Drools is a rule engine that uses rule based approach to implement the expert system.
* Rule never be called directly
* Specific instances cannot be passed to rules directly
* Rules gets executed based on the data matching

**Advantages of rule engine:**

                1)Declarative approach : it specifies what to do  not how to do in a declarative way.

                2) Tools integration : it supports tools like eclipse , debugging and audit logging

                3) Understandable rules: rules are easy to learn , it doesnot require more technical knowledge

                4) Logic and data separation: Data are in domain objects(Fact objects) and logic is in rules give good separation of data and logic.

                5) Centralization of Knowledge: It creates the repository of knowledge.

                6) Speed: it uses patter matching algorithm like Rete to match the domain objects to rules. These algorithms are efficiently proven.

 When to choose Ruleengine ?

* When there is no satisfied traditional programming approach to solve the problem then we can go for rule based approach.
* When there is less technical resource but they are logically strong (domain experts) then we can go for rule based approach.

Basics:

There are two types of knowledge sessions used in rule engine.

1. Stateless knowledge session : it acts like a function call . it pass some data and get the result back. Ex: is the person eligible , calculate something

* DRL – Drools Rule Language
* StatelessKnowledgeSession ksession = kbase.newStatelessKnowledgeSession();
* Applicant applicant = **new** Applicant( "Mr John Smith", 16 );
* Application application = **new** Application();
* assertTrue( application() );
* ksession.execute( Arrays.asList( **new** Object[] { application, applicant } ) );
* assertFalse( application() );

1. Stateful knowledge session : Stateful sessions are longer lived and allow changes over the time. It retains the object state in memory and allows further updates.

StatefulKnowledgeSession ksession = kbase.newStatefulKnowledgeSession();

String[] names = **new** String[]{"kitchen", "bedroom", "office", "livingroom"};

Map<String,Room> name2room = **new** HashMap<String,Room>();

**for**( String name: names ){

    Room room = **new** Room( name );

    name2room.put( name, room );

    ksession.insert( room );

    Sprinkler sprinkler = **new** Sprinkler( room );

    ksession.insert( sprinkler );

}

ksession.fireAllRules()

In the above ex, the rules are fired after the fireAllRules() is called. When we insert an object into rule engine, the rule will not be fired . it will be stored in agenda referred as activation. Agenda is the set of activations to be fired once the fireAllRules() is invoked.

By default all the rules have the default priority as 0. So there may be conflict in executing the rules with same conditions. So we can set the priority to the rule using salience. Salience is the keyword used to set the rule priority. The salience takes the positive integer for the priority . if we set the negative integer , then that rule will be executed last in order.

Ex:

**rule "B to C" salience 5**

**when**

**State(name=="B",state==State.FINISHED)**

**stateObjB:State(name=="C" , state==State.NOTRUN)**

**then**

**stateObjB.setState(State.FINISHED);**

**update(stateObjB);**

**end**

* Ksession.insert(obj) : - inserts an object into working memory
* Ksession.insert(obj) :- update the objects in working memory
* Ksession.retract :- which removes the objects from working memory

**Simple Batch Execution Examples:**

StatelessKnowledgeSession ksession = kbase.newStatelessKnowledgeSession();

List cmds = **new** ArrayList();

cmds.add( CommandFactory.newInsertObject( **new** Cheese( "stilton", 1), "stilton") );

cmds.add( CommandFactory.newStartProcess( "process cheeses" ) );

cmds.add( CommandFactory.newQuery( "cheeses" ) );

ExecutionResults bresults = ksession.execute( CommandFactory.newBatchExecution( cmds ) );

Cheese stilton = ( Cheese ) bresults.getValue( "stilton" );

QueryResults qresults = ( QueryResults ) bresults.getValue( "cheeses" );

Refer drools-spring-integration project

<https://github.com/vasanthaperiyasamy/Drools-Spring.git>