Redhat JBoss Drools

https://www.drools.org/

Knowledge Based Systems

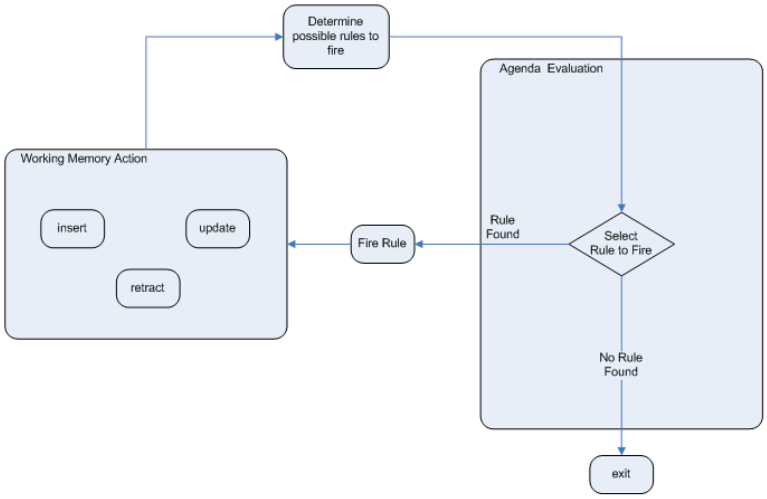
Drools

- Drools is a Business Rules Management System (BRMS) solution. It provides:
  - a core Business Rules Engine (BRE)
  - a web authoring and rules management application (Drools Workbench)
  - an Eclipse IDE plugin for core development
  - complex event processing (CEP)

  https://www.redhat.com/en/technologies/jboss-middleware/business-rules

Rule Engine: Introduction

- The rule engine is the computer program that delivers Knowledge Representation and Reasoning functionality to the developer – it combines facts with rules



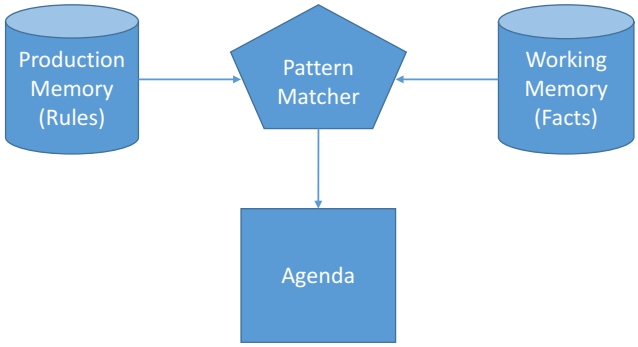
Rule Engine: Introduction

The engine cycles repeatedly through two phases:

- Rule Runtime Actions – execution of the consequence actions (RHS) or the main Java application process; once the consequence has finished or the main Java application process calls fireAllRules() the engine switches to the Agenda Evaluation phase
- Agenda Evaluation – This attempts to select a rule to fire – If more than one rule is activated, a conflict resolution strategy is used – If no rule is found it exits, otherwise it fires the found rule, switching the phase back to Rule Runtime Actions

Drools modules

- Production memory: contains all the rule definition (the drl files)
- Working memory: created with the session and we can add facts to it with the method insert
- Agenda: contains all the rules that can be fired
- Pattern Matcher: is the algorithm that is used to match the rules on the facts given



Facts

- Facts are objects of Java Classes/Beans

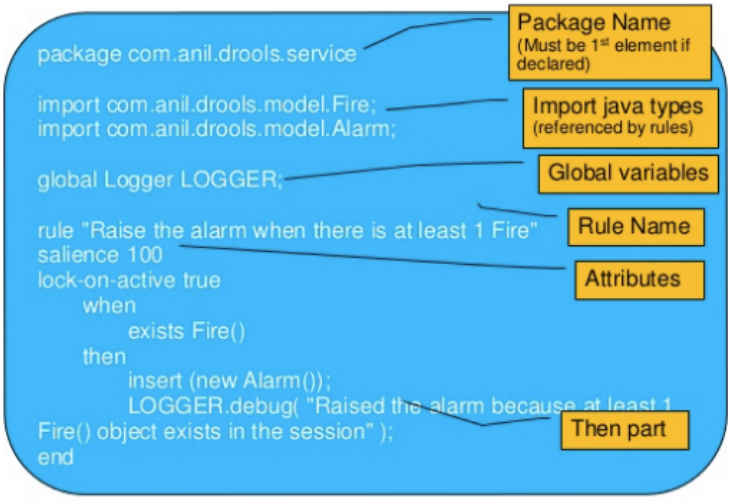
```
public class CashFlow {
	private Date date;
	private double amount;
	private int type; long accountNo;
	// getter and setter methods here
}
public class Account {
	private long accountNo;
	private double balance;
	// getter and setter methods here
}
public AccountPeriod {
	private Date start;
	private Date end;
	// getter and setter methods here
}
```

Rules

```
rule "increase balance for credits”
when
  ap : AccountPeriod()
acc : Account( $accountNo :
  accountNo )
CashFlow( type == CREDIT,
  accountNo == $accountNo,
  date >= ap.start && <= ap.end,
  $amount : amount )
then
  acc.balance  += $amount;
end
```

```
rule "decrease balance for debits”
when
  ap : AccountPeriod()
acc : Account( $accountNo :
  accountNo )
CashFlow( type == DEBIT,
  accountNo == $accountNo,
  date >= ap.start && <= ap.end,
  $amount : amount )
then // RHS: Java syntax
  acc.balance -= $amount;
end
```

Sample Drools Rule



Rule Attributes

- Rule attributes provide a declarative way to influence the behaviour of the rule
  - no-loop – when a rule's consequence modifies a fact if may cause the rule to activate again, causing an infinite loop
  - lock-on-active – this is a stronger version of no-loop, because the change could now be caused not only by the rule itself but by other rules too
  - Salience – is a form of priority where rules (all of whom match) with higher salience values are given higher priority when ordered in the activation queue
  - additional attributes in Drools documentation

Drools Operators

- < <= > >=
  - Person(firstName < $otherFirstName)
- [not] matches (against Java regex)
  - Cheese(type matches "(Bufallo)?\\S*Mozarella")
- [not] contains (check field within array/Collection)
  - CheeseCounter(cheeses contains "stilton")
- soundslike
  - Cheese(name soundslike 'foobar')
  - // matches cheese "fubar" or "foobar"
- str
  - Message(routingValue str[startsWith] "R1")
- [not] in
  - Cheese(type in ("stilton", "cheddar", $cheese))

Drools Conditional Elements

- and / or
  - Cheese(cheeseType: type) or Person(favouriteCheese == cheeseType)
- not
  - not Bus(colour == “red”)
- exists
  - exists Bus(colour == “red”)
- forall
  - forall($bus : Bus (type == ‘english’)
    Bus(this == $bus, color == ‘red’))
- eval
  - eval(p1.getList().containsKey(p2.getItem()))

Drools Conditional Elements

- from
  $order : Order()
  $item : OrderItem(value > 100) from $order.items
- collect
  $system : System()
  $alarms : ArrayList(size >= 3) from collect(Alarm (system == $system, status == ‘pending’))
- accumulate
  $order : Order()
  $total : Number(doubleValue > 100) from accumulate(OrderItem(order == $order, $value : value), sum($value))

Why use Rule Engine?

- Separates application from dynamic logic
  - Rules can be modified by different groups
  - No need to recompile or redeploy
  - All rules are in one place
- Declarative Programming
  - Readable and anyone can easily modify rules
- Centralization of Knowledge
  - Repository of business policy
- Speed and Scalability
  - Rete algorithm

Business Logic Integration Platform

- Since Drools 5, the Business Logic Integration Platform provides a unified and integrated platform for **Rules**, **Workflow** and **Event Processing**
- Drools consist out of several projects:
  - Drools Expert (rule engine)
  - Drools Guvnor (web UI for Business Rule authoring and management) (now it's called Drools Workbench)
  - jBPM (Process/Workflow) (BPM – Business Process Management)
  - Drools Fusion (event processing / temporal reasoning)
  - Drools Planner (automated planning)

Drools Expert & Drools Rule Format

- Drools has an enhanced and optimized implementation of the Rete algorithm for object oriented systems called as ReteOO
- Drools Expert is a declarative, rule based, coding environment
- Drools Rule Formats:
  - Drools Rule Language (DRL)
  - Domain-specific language (DSL)
  - Decision tables (MS Excel files)
  - Guided rule editor
  - XML

Drools Rule Language: Executing Rules

- KnowledgeSession provides the way of exposing objects to be ruled
- Stateless Knowledge Session
  - Doesn’t maintains reference to objects after first call and can be thought of as plain functions
  - Typical use cases include validation, routing, etc
- Stateful Knowledge Session
  - Longer lived, maintains reference to object and allow iterative changes over time
  - Typical use cases include diagnosis, monitoring, etc
  - In contrast to a Stateless session, the dispose() method must be called afterwards to ensure there are no memory leaks

Drools Rule Language: Executing Rules

- A Drools project (or KIE project) contain a file kmodule.xml defining the KieBase and KieSession that can be created
- A Knowledge Base (KieBase), also known as container, is what we call our collection of compiled definitions, such as rules and processes
- The following code snippet compiles all the DRL files found on the classpath and put the result of this compilation, a KieModule, in a KieContainer

```
KieServices kieServices = KieServices.Factory.get();
KieContainer kContainer = kieServices.getKieClasspathContainer();
```

Drools Rule Language: Executing Rules

- If there are no errors, we are now ready to create our kie session from the KieContainer and execute against some data
- KnowledgeSession (kie session) provides the way of exposing objects to be ruled

Drools Rule Language: Executing Rules

KnowledgeSession provides the way of exposing objects to be ruled

- Stateless Knowledge Session

```
StatelessKnowledgeSession ksession = kContainer.newStatelessKnowledgeSession();
Account account = new Account(1001, 15000);
Ksession.execute( account );
```

- Statefull Knowledge Session

```
StatelessKnowledgeSession ksession = kContainer.newKieSession();
Account account = new Account(1001, 15000);
ksession.insert( account );
ksession.fireAllRules();
```

Drools Rule Language

Knowledge base can be updated inside rule's body (RHS):

- insert()
  - Inserted object will be used by rules engines inside current session
- update()
  - Updates existing object in working memory for the rest of rules
- delete()
  - Removed object will not be ruled on current execution

Drools Eclipse IDE

- The Eclipse based IDE provides users with an environment to edit and test rules in various formats, and integrate it deeply with their applications
- Download Drools and Eclipse plugin: https://www.drools.org/download/download.html
- Defining a Drools Runtime
  - Go to preferences window
  - Under Drools category, select “Installed Drools runtimes”
  - Use the default jar files as included in the Drools Eclipse plugin by clicking “Create a new Drools X runtime”

Drools Alternatives

- IBM ILOG Jrules
- Fair Isaac’s Blaze Advisor
- Jess
- OpenRules
- Zilonis
- take