**DROOLS RULE ENGINE**

Drools is a business rule management system (BRMS) and an open-source rule engine. It is developed by the JBoss community, a division of Red Hat. Drools allows you to define and execute business rules in a declarative manner, separate from the application code.

**Dependencies to be added to our build.gradle file**

|  |
| --- |
| implementation group: 'org.drools', name: 'drools-core', version: '7.74.1.Final'  implementation group: 'org.kie', name: 'kie-spring', version: '7.74.1.Final' |

**KIE**

Kie (Knowledge Is Everything) is closely associated with Drools. Kie is a set of projects under the Drools and jBPM umbrella that provide a unified environment for business rules management (Drools) and business process management (jBPM). The integration of Drools and jBPM under the Kie umbrella aims to offer a comprehensive solution for knowledge management and decision automation.

Kie integrates Drools and jBPM into a unified platform for knowledge management and decision automation.

Kie provides a set of common services, APIs, and tools for working with both Drools and jBPM, making it easier to use them together in a seamless manner.

**Kie Container:**

The Kie Container is a runtime environment that allows you to deploy and execute rules and processes. It provides a unified way to work with Drools and jBPM artifacts in an application.

**ARCHITECTURE DIAGRAM**

PasswordPolicyConfig

{minlenghth=20,maxlenght=30}

password

String():pass

Strimg:encoded

**Working**

**Memory**

kieServices

**Drl file**

Rule ”check min len”

When:

$password:password

$rule=PasswordRuleObj()

Check($password.length<maxlength)

Then:

Sopln(“bad password”)

**1.kieSession(facts)**

**2.insert(paswordPolicy)**

**3.insert (password)**

**4. fire all rules (password)**

**kieContainer**

**kieRepository**

**(releaseId)**

kieFileSystem

kieBuilder

KEY TERMS

* ***Facts*** – represents data that serves as input for rules
* ***Working Memory* –**a storage with *Facts,* where they are used for pattern matching and can be modified, inserted and removed
* ***Rule*** – represents a single rule which associates *Facts* with matching actions. It can be written in Drools Rule Language in the *.drl*files or as *Decision Table* in an excel spreadsheet
* ***Knowledge Session*** – it holds all the resources required for firing rules; all *Facts* are inserted into session, and then matching rules are fired
* ***Knowledge Base*** – represents the knowledge in the Drools ecosystem, it has the information about the resources where *Rules* are found, and also it creates the *Knowledge Session*
* ***Module******–***A module holds multiple Knowledge Bases which can hold different sessions

**PARTS IN A DRL FILE**

* ***package –***this is the package name we specify in the *kmodule.xml,*the rule file is located inside this package
* ***import***– this is similar to Java *import* statement, here we need to specify the classes which we are inserting in the *KnowledgeSession*
* ***global –*** this is used to define a global level variable for a session; this can be used to pass input parameter or to get an output parameter to summarize the information for a session
* ***dialect*** – a dialect specifies the syntax employed in the expressions in the condition section or action section. By default the dialect is Java. Drools also support dialect *mvel*; it is an expression language for Java-based applications. It supports the field and method/getter access
* ***rule*** – this defines a rule block with a rule name
* ***when*** – this specifies a rule condition, in this example the conditions which are checked are *Applicant* having *experienceInYears* more than ten years and *currentSalary* in a certain range
* ***then –***this block executes the action when the conditions in the *when* block met. In this example, the *Applicant*role is set as Manager