**The rule execution life cycle in Drools**

The rule execution life cycle in Drools is a key concept that governs how rules are processed, evaluated, and executed within the Drools engine. Understanding this life cycle is essential for effectively designing and managing business rules in a Drools-based application. Here's a detailed overview of the Drools rule execution life cycle:

**1. Phases of the Drools Rule Execution Life Cycle**

The Drools rule execution life cycle consists of several distinct phases, which are executed in a specific order:

* 1. **Knowledge Base and Knowledge Session Creation**
  2. **Fact Insertion**
  3. **Agenda Building**
  4. **Rule Activation**
  5. **Rule Firing**
  6. **After Rule Firing**
  7. **Rule Execution Control**

Let's go through each of these phases in detail:

**1.1 Knowledge Base and Knowledge Session Creation**

* Knowledge Base (KieBase): The Knowledge Base is a container that holds all the compiled rules, processes, and other assets. It is a read-only structure that is created from the rule definitions (DRL files) and remains constant throughout the application’s lifecycle.
* Knowledge Session (KieSession): The Knowledge Session is an instance that interacts with the Knowledge Base. It holds the facts (data objects) and manages the rule execution. There are two types of sessions:
* Stateless KieSession: Suitable for one-off rule executions where no state needs to be preserved between invocations.
* Stateful KieSession: Retains the state between invocations and is useful when rules need to be fired multiple times, or when new facts are inserted dynamically.

Example:

KieServices ks = KieServices.Factory.get();

KieContainer kc = ks.getKieClasspathContainer();

KieBase kBase = kc.getKieBase();

KieSession kSession = kBase.newKieSession();

**1.2 Fact Insertion**

* Fact Insertion: Facts are the data objects that the rules will evaluate. These facts are inserted into the Knowledge Session. Once a fact is inserted, the Drools engine begins evaluating the rules against these facts.

Example:

Customer customer = new Customer("John", 25, "GOLD");

kSession.insert(customer);

* Fact Insertion in Stateful Sessions: In a stateful session, facts can be inserted at any time during the session’s lifecycle. New facts can trigger the re-evaluation of the rules.

**1.3 Agenda Building**

* Agenda: The agenda is a list of rule activations, where each activation represents a rule that can potentially be fired. When facts are inserted into the session, the rules are evaluated, and those that match the facts are placed on the agenda.
* Pattern Matching: Drools uses the Rete algorithm, a powerful pattern-matching algorithm, to efficiently match facts with rule conditions.
* Conflict Resolution: If multiple rules are activated (i.e., multiple rules match the facts), the Drools engine uses conflict resolution strategies (like salience) to prioritize which rule should be fired first.

**1.4 Rule Activation**

* Activation: When the conditions of a rule are met by the facts in the session, the rule is activated. This means it is placed on the agenda and is eligible to be fired.
* Deactivation: A rule can be deactivated if the conditions no longer hold true due to changes in facts or retraction of facts.
* Agenda Groups: Rules can be organized into groups that are activated or deactivated together. This allows more control over the flow of rule execution.

Example:

rule "High Priority Rule"

salience 100

agenda-group "group1"

when

$customer : Customer(age >= 18, membership == "GOLD")

then

// Actions

end

**1.5 Rule Firing**

* Rule Execution: Once a rule is activated, the engine will fire the rule. Firing a rule means that the actions in the then part of the rule are executed.
* No-Loop: A rule can be prevented from re-firing itself if it modifies a fact that would satisfy its own condition again. This is controlled using the no-loop attribute.
* Execution Order: The order in which rules fire is determined by the conflict resolution strategies and the structure of the agenda. Salience, agenda groups, and activation groups all influence the order of rule firing.

Example:

rule "Apply Discount"

when

$customer : Customer(discountEligible == true)

then

$customer.applyDiscount();

update($customer);

end

**1.6 After Rule Firing**

* Post Rule Execution: After a rule fires, the session may need to be updated with the changes made by the rule's actions. For example, if a rule modifies a fact, that fact should be updated in the session using the update() method.
* Propagation: When facts are modified or new facts are inserted, this can trigger the re-evaluation of rules, causing more rules to activate and fire.
* Rule Execution Halt: The session continues to evaluate and fire rules until no more rules are eligible for firing, or the session is explicitly halted.

**1.7 Rule Execution Control**

* Control Over Execution: Drools provides several mechanisms to control the flow of rule execution:
* Halt: Stops the rule engine from firing any further rules.
* FireAllRules: Fires all the rules on the agenda that are eligible.
* FireUntilHalt: Continuously fires rules until the halt() method is called.

Example:

kSession.fireAllRules(); // Fire all eligible rules

* Rule Chaining: Rules can trigger other rules by modifying facts or inserting new facts, leading to a chain of rule executions.

**2. Example of Rule Execution Life Cycle**

Here’s a simple example to illustrate the rule execution life cycle in Drools:

// 1. Knowledge Base and Session Creation

KieServices ks = KieServices.Factory.get();

KieContainer kc = ks.getKieClasspathContainer();

KieBase kBase = kc.getKieBase();

KieSession kSession = kBase.newKieSession();

// 2. Fact Insertion

Customer customer = new Customer("Jane", 30, "PLATINUM");

kSession.insert(customer);

// 3. Agenda Building and 4. Rule Activation (handled by Drools engine)

// 5. Rule Firing

kSession.fireAllRules(); // All eligible rules fire

// 6. After Rule Firing (updates, etc. handled within rules)

kSession.dispose(); // Clean up session resources

**3. Conclusion**

The Drools rule execution life cycle is a structured process that ensures the efficient and predictable execution of business rules. From fact insertion and agenda building to rule activation and firing, each phase plays a crucial role in determining how rules are applied to the data in the session. Understanding this life cycle allows you to better design, optimize, and control the execution of rules in your Drools-based applications.