

2023

DRL- DROOLS Rule Language sintaksa

Knowledge based systems



SADRŽAJ

01	Global	09	Property-reactive beans
02	Insert	10	Java expressions
03	Modify	11	contains, memberOf
04	Delete	12	from clause
05	Rule Attributes	13	collect
06	Agenda-group	14	accumulate
07	Kontrolisanje petlji	15	exists, forAll
08	Declarative types	16	Traits
			Nasleđivanje pravila
			Debugging



Dokumentacija



LET'S START
THE JOURNEY

handbook: <https://kiegroup.github.io/dmn-feel-handbook/#dmn-feel-handbook>

link : https://docs.drools.org/7.49.0.Final/drools-docs/html_single/#_droolsreleasenoteschapter



01 Global

- Služe za interakciju između pravila i koda/podataka koje se nalaze an konteksta Rule Engine-a.
- Definišu se u okviru DRL pravila, sintaksa:

`global {tip} {naziv};`

Npr: `global ServiceClass someService;`

- Mogu da budu bilo šta što se može definisati u Java kodu, poput:
 - Web servisi, DAO, Parametarske vrednosti za konfigurisanje pravila,...



01 Global

- Tipični scenariji korišćenja:
 - U LHS pravila, kao način da se parametrizuje uslov
 - U LHS pravila, kao način da se unesu nove informacije u sesiju
 - U RHS pravila, kao način da se „pokupe“ informacije iz sesije
 - U RHS pravila, kao način za interakciju sa eksternim sistemima



01 Global

- Kako bi koristili globalnu promenljivu u sesiji potrebno je postaviti vrednost globalne promenjive upotrebom setGlobal metode:

```
KieServices ks = KieServices.Factory.get();  
KieContainer kContainer = ks.getKieClasspathContainer();  
KieSession kSession = kContainer.newKieSession(kSessionName: "k-session");  
  
kSession.setGlobal(identifier: "newGlobal", value: 2);
```



01 Global

- KieSession metode za rad sa globalnim promenljivim:
 - void setGlobal(String indentifier, Object value)
 - Globals getGlobals()
 - Object getGlobal(String indentifier)



02 Insert

- U then delu pravila nove podatke (činjenice) moguće je ubaciti sa ključnom reči insert.
- Tako uneti podaci će biti evaluirani od strane svih pravila

```
rule "Create Coupons for Silver Customers"  
  when  
    $o: Order($customer: customer)  
    $c: Customer(this == $customer, category == Category.SILVER)  
  then  
    insert(new Coupon($c, $o, Coupon, CouponType.Points));  
end
```



03 Modify

- Upotrebom ključnih reči modify i update moguće je izmeniti postojeće činjenice u radnoj memoriji
- Opšta preporuka je da se uvek koriste modify blokovi
- Izmenjeni podaci će biti ponovo evaluirani od strane svih pravila

```
rule "Categorize Customer - Gold"
when
    $c: Customer(category == Customer.Category.NA)
    $o: Order(customer == $c, orderLines.size()>10)
then
    modify($c){setCategory(Customer.Category.GOLD);}
    // or update, but update doesn't work as expected :)
end
```



04 Delete

- delete briše činjenice iz radne memorije

```
rule "Expire coupons"  
  when  
    $now: Date()  
    $cp: Coupon( validUntil <$now)  
  then  
    delete($cp)  
  end
```



05 Rule attributes

- Drools pravila su data-driven
- Jedini načini da se pravilo aktivira je da se rule enginu dodaju podaci koji odgovaraju uslovim tog pravila
- U određenim slučajevima želimo da se pravila čiji su uslovi zadovoljeni filtriraju
- Jedan od mehanizama koji omogućava to su atributi



05 Rule attributes

- Atributi predstavljaju dodatak pravilima koji omogućava njihovu modifikaciju
- Opcioni su, pravilo može imati proizvoljan broj atributa
- Navode se posle imena pravila, pre when dela
- Evaluiraju se nakon što je when deo nekog pravila zadovoljen
- Moguće je koristiti podatke iz when dela kao i globalne varijable za postavljanje njihovih vrednosti
- Često korišćeni atributi:
 - enabled
 - salience
 - no-loop
 - lock-on-active

```
rule "simple attribute example"
enabled false
|
| when
|   Customer()
| then
|   System.out.println("we have a customer")
end
```



06 Kontrolisanje petlji

- Pravila mogu da se aktiviraju na osnovu rezultata izvršavanja drugih pravila
- Time je omogućeno razbijanje kompleksnosti u jednostavna pravila koja međusobno interaguju pomoću podataka u radnoj memoriji
- Zbog toga potrebno je voditi računa da ne dođe do neželjenog aktiviranja pravila



06 Kontrolisanje petlji

- Najjednostavniji slučaj neželjene aktivacije – beskonačna petlja kada pravilo modifikuje podatke u radnoj memoriji koje izazove njegovu ponovnu aktivaciju

```
rule "Apply 10% discount on notepads"
  when
    $i: Item(name == "notepad", $sp:salePrice)
  then
    modify($i){setSalePrice($sp *0.9);}
  end
```



06 Kontrolisanje petlji no-loop

- no-loop atribut sprečava da pravilo neposredno aktivira samog sebe

```
rule "Apply 10% discount on notepads"  
  no-loop // ili no-loop true  
  when  
    $i: Item(name == "notepad", $sp:salePrice)  
  then  
    modify($i){setSalePrice($sp *0.9);}  
end
```



06 Kontrolisanje petlji no-loop

- Ukoliko neko drugo pravilo ponovo izmeni podatke čija će izmena rezultovati aktivacijom prvog pravila no-loop atribut neće sprečiti ponovno aktiviranje prvog pravila

```
rule "Give extra 2% discount for orders larger than 15 items"
  no-loop true
  when
    | $o: Order(totalItems > 15)
  then
    | modify($o){increaseDiscount(0.02);}
  end
```

```
rule "Give extra 2% discount for orders larger than $100"
  no-loop true
  when
    | $o: Order(total > 100.00)
  then
    | modify($o){increaseDiscount(0.02);}
  end
```



06 Kontrolisanje petlji lock-on-active

- lock-on-active atribut sprečava da se pravilo ponovo izvrši za isti objekat/fact
- Promena aktivne agenda grupe će omogućiti ponovno izvršavanje pravila

```
rule "Give extra 2% discount for orders larger than $100"  
|  
  lock-on-active true  
|  
  when  
|  
    $o: Order(total > 100.00)  
|  
  then  
|  
    modify($o){increaseDiscount(0.02);}  
|  
end
```



06 Kontrolisanje petlji model properties

- Mana no-loop i lock-on-active atributa je što se odnose na cele objekte
- Nekad je poželjno da se pravila ponovo izvrše kada se izmene određena svojstva
- Jedan od načina je da se u modelu dodaju odgovarajući
- flag-ovi



06 Kontrolisanje petlji model properties

```
rule "Add 2% discount for orders larger than $100"  
  when  
    $o: Order(total > 100.00, has100DollarsDiscount == false)  
  then  
    modify($o){  
      increaseDiscount(0.02);  
      setHas100DollarsDiscount(true);  
    }  
  end
```

```
rule "Add 2% discount for orders larger than $100"  
  when  
    $o: Order(totalItems > 15, has100DollarsDiscount == false)  
  then  
    modify($o){  
      increaseDiscount(0.02);  
      setHas15DollarsDiscount(true);  
    }  
  end
```



06 Kontrolisanje petlji model properties

- Dodavanje flag-ova ima dva osnovna problema:
 - Vremenom će doći do saturacija modela sa dodanim svojstima koje nemaju direktnu relaciju sa stvarnim sadržajem modela
 - Sa velikim brojem pravila broj flag-ova koje treba proveriti, da bi pravila funkcionisala na jednostavan način, može biti ogroman



07 Agenda group

- U projektima pravila se obično mogu razvrstati po grupama, pri čemu ima smisla u jednom trenutku aktivirati samo pravila iz neke određene grupe, npr.:
 - pravila za validaciju unosa,
 - pravila za primenu sniženja,
 - pravila za korisnike i sl.
- Grupisanjem pravila se sprečava da rule engine u svakom trenutku izvršava sva pravila koja su definisana
- Drools omogućava grupisanje pravila pomoć agenda-group atributa



07 Agenda group

- U projektima pravila se obično mogu razvrstati po grupama, pri čemu ima smisla u jednom trenutku aktivirati samo pravila iz neke određene grupe, npr.:
 - pravila za validaciju unosa,
 - pravila za primenu sniženja,
 - pravila za korisnike i sl.
- Grupisanjem pravila se sprečava da rule engine u svakom trenutku izvršava sva pravila koja su definisana
- Drools omogućava grupisanje pravila pomoć agenda-group atributa



07 Agenda group

- agenda-group atribut definiše ključ, koji se može aktivirati pozivom odgovarajuće metode KieSession objekta

drl fajl



```
rule "Promotion: more than 10 pencils get 10% discount"
    agenda-group "promotions"
    when
        OrderLine(item.getName() == "pencil", quantity >10)
    then
        insert(new (Discount(0.10)));
    end
```

java kod



```
kSession.getAgenda().getAgendaGroup(name: "promotions").setFocus();
kSession.fireAllRules();
```



08 Declared types

- Drools omogućava definisanje modela na dva načina:
 - Upotrebom Java klasa
 - Definisanjem u DRL fajlovima (declared types)
- Declared types se definišu pre pravila u DRL fajlovima

```
declare SpecialOrder extends Order
    whatsSoSpecialAboutIt: String
    order: Order
    applicableDiscount: Discount
end
```



08 Declared types

- Declared types:
 - Definišu se između declare i end ključnih reči
 - Mogu da definišu objektne attribute (poput String) i primitivne attribute (poput long)
 - Mogu da nasleđuju druge tipove, uključujući Java klase i druge declared types
 - Geteri/Seteri za attribute se automatski generišu



08 Declared types

- Declared types objektima moguće je pristupiti u java kodu korišćenjem refleksije
- Preporuka je da se koriste samo ako će im se pristupati u okviru pravila

```
FactType type = kSession.getKieBase().getFactType(  
    packageName: "chapter04.declaredTypes", typeName: "SpecialOrder");  
Object instance = type.newInstance();  
type.set(instance, field: "relevance", value: 2L);  
Object attr = type.get(instance, field: "relevance");
```




09 Property-reactive beans

- Upotrebom modify/update ključnih reči notifikuje se rule engine da pravila koja rade sa sličnim tipovima objekata treba da re-evaluiraju taj objekat
- Re-evaluacija se obično odnosi na čitav objekat
 - Kada se promeni neko svojstvo objekta, pravila taj objekat tretiraju kao novi objekat
- Problemi nastaju kada ne želimo da se re-evaluacija desi za neke promene



09 Property-reactive beans

- Drools poseduje mehanizam za praćenje promena atributa bean-ova i reakciju na njihove promene
- Potrebno je anotirati Java klasu ili Declared type anotacijom `@PropertyReactive`



```
package com.ftn.model;
import org.kie.api.definition.type.PropertyReactive;

@PropertyReactive
public class SomeClass {
```

```
declare PropertyReactiveOrder
    @PropertyReactive
    discount: Discount
    totalItems: Integer
    total: Double
end
```

09 Property-reactive beans

- Nakon obeležavanja property reactive bean-a u pravilma je moguće obeležiti atributi koji se filtriraju
- Obeležavanje se vrši uz pomoć @Watch anotacije
- @Watch anotacija navodi se na kraju svakog uslova koje treba filtrirati
- U RHS obavezno koristiti modify keyword

```
rule "Larger than 20 items orders are special orders"  
  when  
    $o: PropertyReactiveOrder(totalItems > 20)    @Watch  
  then  
    modify($o){  
      setDiscount(new Discount(0.05));  
    }  
end
```



09 Property-reactive beans

@ Watch

- Primeri korišćenja:
 - @Watch(discount, total): This only monitors changes in the discount and total attributes
 - @Watch(!discount): This means that we should disregard changes to the discount attribute of the bean
 - @Watch(*, total): This means that we should disregard all the attributes of the bean, except for the total attribute
 - @Watch(*, total, totalItems): This means that we should only pay attention to changes of the total and totalItems attributes, other changes are disregarded



10 Java expressions

- U condition delu pravila moguće je koristiti Java expressions koji vraćaju boolean vrednosti:
 - `Person(age == 50)`
 - `Person(age > 100 && (age % 10 == 0))`
 - `Person(Math.round(weight / (height * height)) < 25.0)`
- Napomena: sve metoda u LHS delu pravila treba da budu read only
- Pogrešno: `Person(incrementAndGetAge() == 10)`



10 Java expressions

AND

- Zarez(',') predstavlja implicitni AND:
 - `Person(age > 50, weight > 80)`
 - `Person(age > 50 && weight > 80)`
- Iako `&&` i `'&'` imajo istu semantiku imajo različite prioritete
 - `&&` je prioritetnejše od `||`
 - `&&` i `||` imajo večji prioritet od `'&'`



10 Java expressions

OR

- Pravila bi trebalo da budu što jednostavnija
- Ukoliko se u pravilima koristi OR potrebno je proveriti da li je moguće rastaviti pravilo

```
rule "Add 5$ discount for minors or seniors"  
  when  
    $o: Order(customer.getAge() <18 || customer.getAge()>60)  
  then  
    $o.increaseDiscount(0.05);  
end
```



10 Java expressions

OR

```
rule "Add 5$ discount for minors"
  when
    | $o: Order(customer.getAge() <18)
  then
    | $o.increaseDiscount(0.05);
  end
```

```
rule "Add 5$ discount for seniors"
  when
    | $o: Order(customer.getAge()>60)
  then
    | $o.increaseDiscount(0.05);
  end
```



10 Java expressions

NOT

```
rule "warn about empty working memory"  
  when  
    | not(Order()) or Item()  
  then  
    System.out.println("empty memory")  
end
```

```
rule "warn about empty working memory"  
  when  
    | not(Order())  
    | not(Item())  
  then  
    System.out.println("empty memory")  
end
```



10 Java expressions

NOT

```
rule "warn about empty working memory"
  when
    | not(Order() or Item())
  then
    System.out.println("empty memory")
  end
```

```
rule "warn about empty working memory"
  when
    | not(Order())
    | not(Item())
  then
    System.out.println("empty memory")
  end
```



11 contains, memberOf

- Pripadnost objekta činjenice kolekciji može se proveriti uz pomoć contains i memberOf operatora:

```
rule "print orders with pencils in them"  
  when  
    $i: Item(name == "pencil")  
    $ol: OrderLine(item == $i)  
    $o: Order($ol memberOf orderLines,  
              orderLines contains $ol)  
  then  
    System.out.println("order with pencils: " + $o);  
end
```



12 from clause

- Drools omogućava proveru uslova nad objektima / činjenicama koje se ne nalaze u radnoj memoriji uz pomoć from klauzula:

```
rule "For every notebook order apply points coupon"  
  when  
    $o: Order($c:customer, $lines: orderLines)  
    OrderLine($item: item) from $lines  
    Item(name == "notebook") from $item  
  then  
    insert(new Coupon($c, $o, CouponType.POINTS));  
  end
```



13 collect

- collect se koristi da pronade i grupise sve činjenice koje zadovoljavaju određeni uslov u kolekciju:

```
rule "Grouping orders"  
  when  
    $list: List() from collect(Order())  
  then  
    System.out.println("we found: " + $list.size() + " orders.");  
  end
```

- Ako ne postoje činjenice koje odgovaraju filteru vratiće praznu listu. To se može sprečiti dodavanjem uslova listi:

```
$list: List(size>0) from collect(Order())
```



14 accumulate

- accumulate omogućava primenu transformacija nad objektima koji zadovoljavaju određeni uslov
- Najčešći scenariji korišćenja pronalazak prosečne, minimalne, maksimalne vrednosti određenog atributa objekata
- Korišćenjem predefinisanih accumulate funkcija
- **Napomena: Rad sa funkcijama init, action, reverse i result se neće priznavati.**

```
rule "10+ items with sale price over 20 get discount"
when
    $o: Order($lines:orderLines)
    Number(intValue >=10) from accumulate(
        OrderLine(
            item.salePrice > 20.00,
            $q: quantity
        )from $lines,
        sum($q)
    )
then
    $o.increaseDiscount(0.05);
end
```



14 accumulate

- accumulate funkcije:
 - count: This keeps a count of a variable that matches a condition
 - sum: This sums up a variable, as shown in the previous example
 - average: This gets the average value of a variable for all the matches in the accumulate
 - min: From all the variable values obtained in the accumulate condition, this returns the minimal one
 - max: From all the variable values obtained in the accumulate condition, this returns the maximum one
 - collectList: This stores all the values of a specified variable in the condition in an ordered list and returns them at the end of the accumulate
 - collectSet: This stores all the values of a specified variable in the condition in a unique
 - elements' set and returns them at the end of the accumulate



14 accumulate

- Moguće je iskoristiti više accumulate funkcija odjednom:

```
rule "multi-function accumulate example"  
  when  
    accumulate(  
      Order($total:total),  
      $maximum: max($total),  
      $minimum: min($total),  
      $avg: average($total),  
    )  
  then  
    //....  
end
```



15 exists, forAll

- exists proverava da li se određeni objekti nalaze u radnoj memoriji:

```
rule "with exists"  
  when  
    | exists(Order())  
  then  
    System.out.println("we have orders");  
end
```

```
rule "without exists"  
  when  
    | $o: Order()  
  then  
    System.out.println("we have orders: " + $o);  
end
```



15 exists, forAll

- forAll proverava dva uslova, vraća true ukoliko postoji objekat koji zadovoljava oba uslova, ili ako ne postoje objekti koji zadovoljavaju uslove

```
rule "make sure all orders have at least one line"
when
    exists(Order())
    forAll(
        Order()
        Order(orderLines.size()>0)
    )
then
    System.out.println("all orders have lines");
end
```



16 Traits

- Dodavanje novih osobina činjenicama, bez izmene modela, je moguće upotrebom Trait-ova
- Iz OO perspektive Trait bi predstavljao pridev koji dodatno opisuje objekat
- Za korišćenje potrebno je:
 - definisati trait
 - označiti (odgovarajuće) klase model anotacijom @Traitable



16 Traits

- Definisanje je moguće odraditi na dva načina:
 - Pomoću Java klase koja je označena @Trait anotacijom
 - Pomoću declared tipa dodavanjem modifikatora trait

```
declare trait KidFriendly  
| kidsAppeal: String  
end
```



16 Traits

- Dodavanje trait-ova se vrši upotrebnom ključne reči don:

```
rule "toy items are kid frinedly"  
  no-loop  
  when  
    $i: TraitableItem(name contains "toy")  
  then  
    KidFriendly kf = don($i, KidFriendly.class);  
    kf.setKidAppeal("can play with it");  
  end
```

- Uklanjanje trait-ova se vrši upotrebnom ključne reči shed



Nasledivanje pravila

```
rule "A"  
when  
    s: String(this == "A")  
then  
    System.out.println(s);  
end  
  
rule "B" extends "A"  
when  
    i: Integer(intValue > 2)  
then System.out.println(i);  
end
```

```
rule "A"  
when  
    s: String(this == "A")  
then  
    System.out.println(s);  
end  
  
rule "B"  
when  
    s: String(this == "A")  
    i: Integer(intValue > 2)  
then  
    System.out.println(i);  
end
```



DEBUGGING - RHS

- Debug RHS pravila moguće je odraditi na klasičan način kao u JAVI.



DEBUGGING - LHS

- Preporuke:
 - DRL resurse razdvajati po celinama
 - Ukoliko aplikacija se sastoji od nezavisnih komponenti, razdvojiti ih u posebne KieBase
 - Koristiti KieHelper umesto KieContainer classpath
 - Ukoliko aplikacija uz pomoć globala poziva servise, napraviti mock varijante tih servisa



DEBUGGING - LHS

- Tipično se javljaju tri tipa problema u LHS:
 - Compilation errors
 - Runtime errors
 - Rules not being triggered when they should



DEBUGGING - LHS - Event Listener

- Kako bi ispratili ponašanje pravila moguće je dodati odgovarajući Event Listener
- RuleRuntimeEventListener
 - DebugRuleRuntimeEventListener
- AgendaEventListener
 - DebugAgendaEventListener
- RuleRuntimeEventListener
 - DebugRuleRuntimeEventListener



Pitanja?

