

# 2023

# Sistemi bazirani na znanju

*Knowledge based systems*



# SADRŽAJ

01

**EKSPERTSKI SISTEMI**

03

**DROOLS**

02

**EKSPERTSKI  
SISTEMI BAZIRANI  
NA PRAVILIMA**



01

**EKSPERTSKI  
SISTEMI**



# Ekspertski sistemi

Grana veštačke inteligencije.

Osnovna ideja: Napraviti inteligentni program, tako što bi program inkorporirao esencijalno domen specifično znanje iz oblasti rešavanja problema

Simuliraju ljudsko rezonovanje u određenom vremenu



# Tipovi ekspertskih sistema

**Neural  
networks**

**Blackboard  
Systems**

**Belief  
(Bayesian)  
Networks**

**Case-Based  
Reasoning**

**Rule-Based  
Systems**



Šta su ovi sistemi  
i kako donose  
odluke?

# Tipovi ekspertskih sistema

**Neural  
networks**

**Blackboard  
Systems**

**Belief  
(Bayesian)  
Networks**

**Case-Based  
Reasoning**

**Rule-Based  
Systems**



Svi oni se zasnivaju na obradi podataka u cilju donošenja odluke.

# Upoređivanje sistema

	Neural Networks	Blackboard Systems	Belief Networks	Case-Based Systems	Rule-Based Systems
Donošenje odluke:	1. Odlučivanje NN	1. Odlučivanje crnom tablom	1. Odlučivanje acikličnim grafom	1. Odlučivanje pronalaženjem najbližeg slučaja iz baze slučajeva	1. Odlučivanje definisanim pravilima
Učenje/ Tačnost rezultata:	Da / Manje od 100%	Da / Manje od 100%	Da / Manje od 100%	Ne / Manje od 100%	Ne / Uvek 100%



# **Ekspertski sistemi bazirani na pravilima**

## ***Rule Based Expert Systems***

Softver koji sadrži domensko znanje eksperta.

Softver koji pokušava da reši problem tako što će ukloniti neodređenosti na isti način kao što bi to odradili stručnjaci iz određenog domena (koristeći svoje znanje)





# 02

## EKSPERTSKI SISTEMI BAZIRANI NA PRAVILIMA



# **Ekspertski sistemi bazirani na pravilima**

## ***Rule Based Expert Systems***

Ekspertski sistem koji koristi pravila i činjenice za donošenje odluka.

Ulazni podaci na koje primenjujemo pravila i koji se sladište nazivaju se činjenice **(facts)**.

Provera podataka (obrada) se uspostavlja pravilima **(rules)**. Da li podaci pripadaju datom scope-u, tj. da li zadovoljavaju dato pravilo?



# **Ekspertski sistemi bazirani na pravilima**

## ***Rule Based Expert Systems***

Ako podaci zadovoljavaju dato pravilo oni mogu aktivirati sledeće pravilo i tako aktivirati izvršavanje lanca pravila.

Načini izvršavanja lanca pravila:

**1. Forward chaining**

**2. Backward chaining**



# Ekspertski sistemi bazirani na pravilima - **Forward chaining**

Forward chaining ili rezonovanje dedukcijom. Ovaj način rezonovanja vođen je podacima (prepoznaš činjenicu pa doneseš zaključak, ne zna se unapred problem koji se rešava),



# Ekspertski sistemi bazirani na pravilima - **Backward chaining**

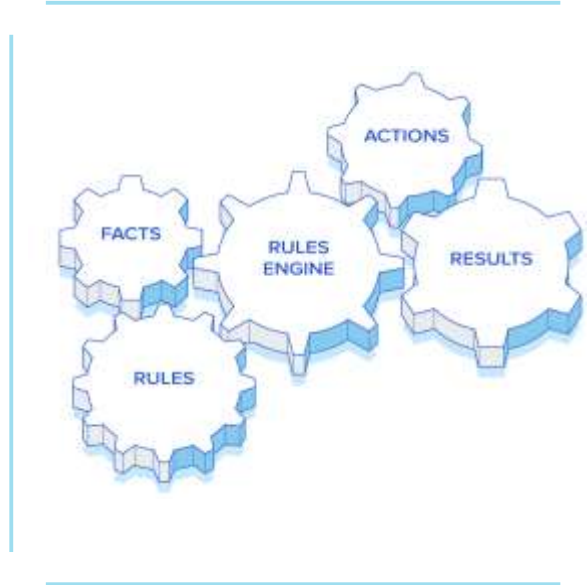
Backward chaining ili rezonovanje indukcijom. Ovaj način rezonovanja vođen je ciljem (rešava se talno određeni problem)



# Ekspertski sistemi bazirani na pravilima

Ekspertski sistemi oslanjaju se na bazu znanja (**knowledge-base**) i na rezoner (**rule engine**) koji uvek izvršava određene procedure rezonovanja koristeći bazu znanja i radnu memoriju (**facts**) kao ulaz.





# 03

## DROOLS



# Dokumentacija



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](https://docs.drools.org/7.49.0.Final/drools-docs/html_single/#_droolsreleasenoteschapter)





# Uputstvo za instalaciju



link za VS Code: <https://www.youtube.com/watch?v=Cc1ld2Cd8LU>

Requirements za Windows: git bash

Verzija: Bilo koja FINAL (poželjno poslednja)

link za ECLIPSE : [https://www.pi.github.io/tutorials/lectures/lsp/030\\_install\\_drools.html](https://www.pi.github.io/tutorials/lectures/lsp/030_install_drools.html)



# DROOLS

Business Rule Management System (BRMS)

**Open Source**

Red Hat/ JBOSS

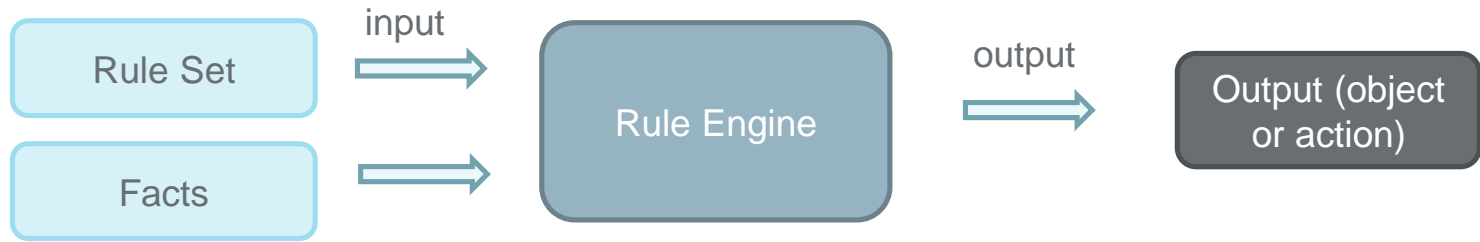
JAVA

Implementira prošireni **RETE algoritam**.

Drools je deo KIE ekosistema. Kie je ekosistem open source projekata fokusiranih na automatizaciji poslovanja.



# DROOLS - Kako se pretprocesiraju pravila?



The rules tell the Rule Engine what action to take when certain conditions are met.

Rule engine applies the rules to the facts.

The Rule Engine takes the specified action on the rules that fired to outcome object or action.



# Struktura .DRL fajlova

- Pravila se definišu u okviru .drl fajlova
- .drl fajlovi su statički resursi i smeštaju se u okviru src/main/resources direktorijuma
- Svako .drl fajl ima opštu strukturu prikazanu na slici

```
firstRule.drl x
vezbe-1-kjar > src > main > resources > firstRule.drl
1 package com.sample
2 // list any import classes here
3
4 // declare any global variables here
5
6 rule "Your first rule"
7   when
8     //conditions
9   then
10    //action
11  end
12
13 rule "Your second rule"
14   //include attributes such as "salience" here...
15   when
16     //conditions
17   then
18    //action
19  end
```



# Struktura .DRL fajlova

Opšta sturuktura:

- Definicija paketa: Kao u Javi definiše se paket za pravila
- Import sekcija: Potrebno je importovati sve klase koje će biti iskorišćene za realizaciju pravila
- (Opciono) sekcija za deklaraciju tipova i događaja
- Pravila

```
firstRule.drl x
vezbe-1-kjar > src > main > resources > firstRule.drl
1 package com.sample
2 // list any import classes here
3
4 // declare any global viriables here
5
6 rule "Your first rule"
7   when
8     //conditions
9   then
10    //action
11  end
12
13 rule "Your second rule"
14   //include attributes such as "salience" here...
15   when
16     //conditions
17   then
18     //action
19  end
```



# Struktura pravila

- Conditions se pišu uz pomoć DRL jezika
- LHS pravila se sastoji od conditional elements, koji služe kao filteri da definišu uslove koje Facts trebaju da zadovolje kako bi se pravilo izvršilo
- RHS pravila sadrži posledice aktiviranja pravila

**rule** "name"

**when**

(Conditions) – also called Left Hand Side of the Rule (LHS)

**then**

(Actions/Consequence) – also called Right Hand Side of the Rule (RHS)

**end**



# Saveti za pisanje pravila

- Pravila bi trebalo da budu što jednostavnija
- Kompleksna pravila po mogućnosti razdvojiti
- Pravila bi trebalo da budu nezavisna (bez eksplicitnih poziva iz drugih pravila)
- Povezivanja pravila se vrši dodavanjem informacija o domenu pravila
- Primer:
  - When we get a signal from a fire alarm, we infer that there is a fire
  - When there is a fire, we call the fire department
  - When the fire department is present, we let them in to do their work



# Drools – HelloWorld primer

- Pravilo je zadovoljeno ukoliko postoji u sesiji POJO koji kao vrednost polja status ima vrednost Message.HELLO
- Ukoliko takav Fact (POJO) postoji, u promenljivu myMessage se smešta vrednost message polja
- m predstavlja variable binding, odnosno čuva referencu objekta koji je zadovoljio pravilo (Preporuka ja da imena tih promenljivih počinju sa znakom \$ -> umesto m trebalo bi da bude **\$m**)

```
package com.ftn
import com.ftn.model.Message;

rule "Hello World"
    when
        $m: Message(status == Message.HELLO, myMessage: message)
    then
        System.out.println( myMessage);

        modify($m){setMessage("Goodbye cruel world"), setStatus( Message.GOODBYE) };
    end

rule "GoodBye"
    //include attributes such as "saliency" here...
    when
        Message(status == Message.GOODBYE, myMessage: message)
    then
        System.out.println(myMessage);
    end
```





# Drools – HelloWorld primer

- U okviru RHS se nalazi konsekvencija pravila
- Ispisuje se poruka objekta
- Menja se poruka i status
- Na kraju poziva se update metoda koja notifikuje Rule Engine da je činjenica promenjena i da se trebaju evaluirati promene u odnosu na ostala pravila

```
package com.ftn
import com.ftn.model.Message;

rule "Hello World"
    when
        $m: Message(status == Message.HELLO, myMessage: message)
    then
        System.out.println( myMessage);

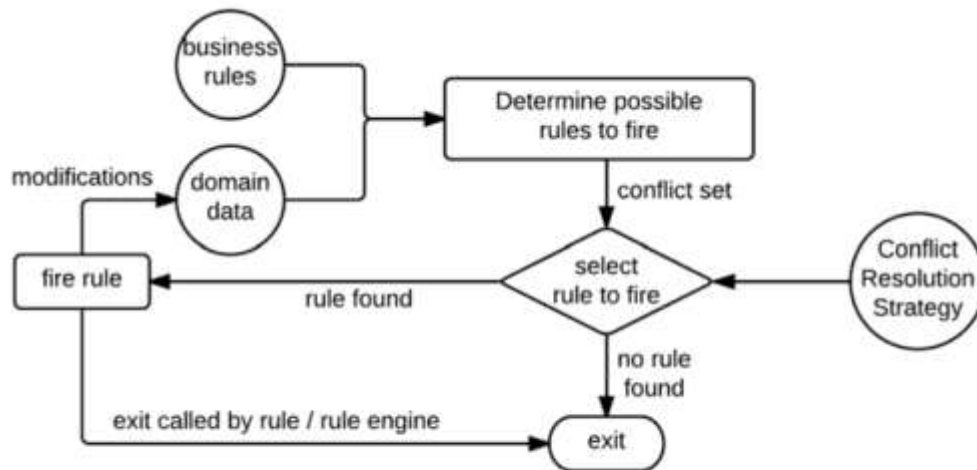
        modify($m){setMessage("Goodbye cruel world"), setStatus( Message.GOODBYE) };
    end

rule "GoodBye"
    //include attributes such as "salience" here...
    when
        Message(status == Message.GOODBYE, myMessage: message)
    then
        System.out.println(myMessage);
    end
```



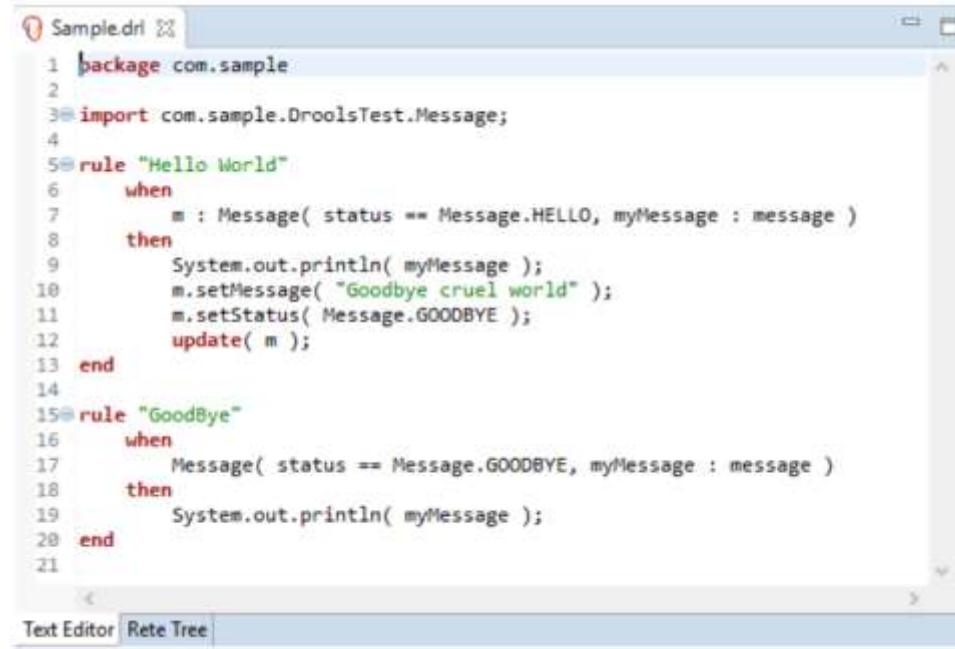
# Drools – HelloWorld primer

- Proces izvršavanja pravila:



# Drools – HelloWorld primer

- Nakon update metode postoji činjenica (fact) koja zadovoljava GoodBye pravilo
- GoodBye pravilo samo ispisuje poruku „Goodbye cruel world“



```
1 package com.sample
2
3 import com.sample.DroolsTest.Message;
4
5 rule "Hello World"
6   when
7     m : Message( status == Message.HELLO, myMessage : message )
8   then
9     System.out.println( myMessage );
10    m.setMessage( "Goodbye cruel world" );
11    m.setStatus( Message.GOODBYE );
12    update( m );
13  end
14
15 rule "GoodBye"
16   when
17     Message( status == Message.GOODBYE, myMessage : message )
18   then
19     System.out.println( myMessage );
20  end
21
```

The screenshot shows a text editor window titled "Sample.drl" containing two Drools rules. The first rule, "Hello World", is triggered when a Message object has a status of HELLO. It prints the message, updates it to "Goodbye cruel world", and changes its status to GOODBYE. The second rule, "GoodBye", is triggered when a Message object has a status of GOODBYE and simply prints the message. The interface includes a "Text Editor" tab and a "Rete Tree" tab at the bottom.



# Drools – HelloWorld primer

- Nakon update metode postoji činjenica (fact) koja zadovoljava GoodBye pravilo
- GoodBye pravilo samo ispisuje poruku „Goodbye cruel world“

```
package com.ftn
import com.ftn.model.Message;

rule "Hello World"
    when
        $m: Message(status == Message.HELLO, myMessage: message)
    then
        System.out.println( myMessage);

        modify($m){setMessage("Goodbye cruel world"), setStatus( Message.GOODBYE) };
    end

rule "GoodBye"
    //include attributes such as "salience" here...
    when
        Message(status == Message.GOODBYE, myMessage: message)
    then
        System.out.println(myMessage);
    end
```



# Drools – HelloWorld primer

- Jednostavni oblik pravila koji bi se izvršio svaki put kada Rule Engine pronade Message:

```
rule "Hello"  
    when  
        Message( )  
    then  
        System.out.println("Message exists!")  
    end
```



# Drools – HelloWorld primer

- Kako bi se pravila izvršila potrebno je proslediti Rule Engine-u potrebne podatke
- Sa slike mogu se identifikovati tri celine u main metodi:
  - Instanciranje Rule Engine
  - Prosleđivanje podataka Rule Engine-u
  - Aktiviranje pravila

```
1 package com.ftn.service;
2 import org.kie.api.KieServices;
3 import org.kie.api.runtime.KieContainer;
4 import org.kie.api.runtime.KieSession;
5
6 import com.ftn.model.Message;
7
8
9 public class Test{
10     public static void main(){
11         try{
12             // instanciranje
13             KieServices ks = KieServices.Factory.get();
14             KieContainer kContainer = ks.getKieClasspathContainer();
15             KieSession kSession = kContainer.newKieSession(kSessionName: "k-session");
16             // insertovanje fact-a
17             Message message = new Message();
18             message.setMessage(message: "Hello World");
19             message.setStatus(Message.HELLO);
20             kSession.insert(message);
21             kSession.fireAllRules();
22         } catch (Throwable t){
23             t.printStackTrace();
24         }
25     }
26 }
27
28
```



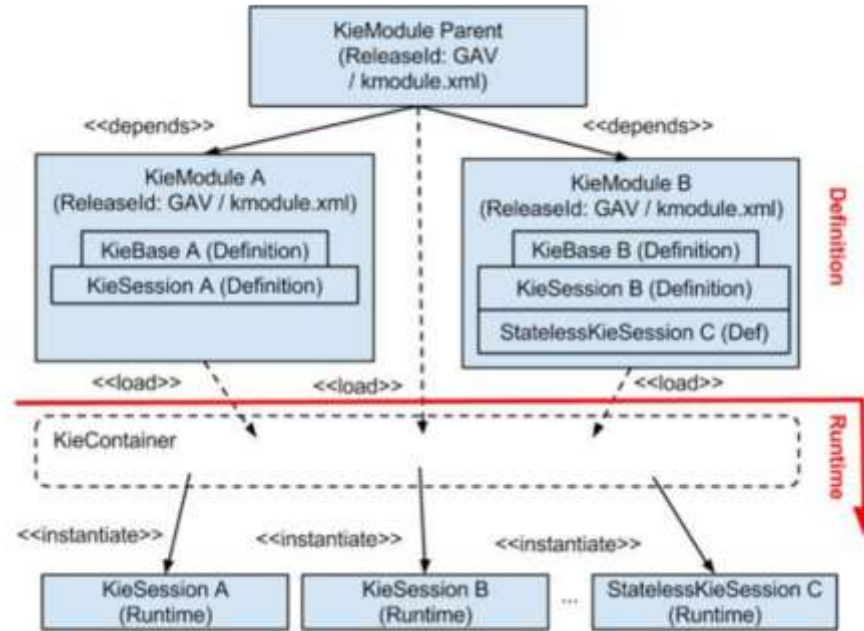
# DROOLS - Instanciranje Rule Engine-a

Postoje pet bitnih koncepata koje služe za konfigurisanje instanci rule engine-a:

- KieServices – KieSevices klasa omogućava pristup svim ostalim konceptima. Uz pomoć KieServices klase se pravi nova instanca KieContainer-a.
- KieContainer – definiše opseg pravila koje služe za pravljenje novih instanci rule engine-a. KieContainer hostuje KieModule i njihove dependencije.
- KieModule – Svaki KieModule sadrži business assets (business rules, business processes, decision tables...). KieModule je standardni Java-Maven projekat koji sadrži pravila. Specijalni fajl kmodule.xml definiše konfiguraciju asseta u modulu.
- KieBase – predstavlja kompajliranu verziju asseta.
- KieSession – predstavlja instancu rule engine-a koja sadrži pravila u KieBase



# DROOLS - Instanciranje Rule Engine-a





# DROOLS - Kie session

- Postoje dva tipa sesija:
  - Stateful Session (KieSession)
  - Stateless Session (StatelessKieSession)
- U opštem slučaju koriste se stateful sesije.
- Stateless sesije koriste za jednostavne slučajeve korišćenja, obično se posmatraju kao funkcije.  
Mogućnost primene:
  - Validacije
  - Kalkulacije
  - etc.



# DROOLS - Prosleđivanje podataka Rule Engine-u

- Rule engineu se prosleđuje POJO koji predstavlja činjenicu (fact)
- Prosleđivanje se vrši pomoću insert metode nad sesijom.
- Sam unos podataka neće aktivirati izvršavanje pravila. Da bi to uradili potrebno je pozvati fireAllRules metodu na sesijom.

```
1 package com.ftn.service;
2 import org.kie.api.KieServices;
3 import org.kie.api.runtime.KieContainer;
4 import org.kie.api.runtime.KieSession;
5
6 import com.ftn.model.Message;
7
8
9 public class Test{
10     public static void main(){
11         try{
12             // instanciranje
13
14             KieServices ks = KieServices.Factory.get();
15             KieContainer kContainer = ks.getKieClasspathContainer();
16             KieSession kSession = kContainer.newKieSession(kSessionName: "k-session");
17             // insertovanje fact-a
18             Message message = new Message();
19             message.setMessage(message: "Hello World");
20             message.setStatus(Message.HELLO);
21             kSession.insert(message);
22             kSession.fireAllRules();
23         } catch (Exception e){
24             t.printStackTrace();
25         }
26     }
27 }
```



# DROOLS - Fact handle

- Insert metoda vraća FactHandle nad prosleđenom činjenicom.
- FactHandle predstavlja referencu ka činjenici u sesiji i može se koristiti za update i delete činjenici direktno iz programa.
- Update metoda kao parametre prima FactHandle i izmenjeni objekat na osnovu koga menja vrednost činjenice.
- Delete metoda briše činjenicu iz sesije, kao parametar prima FactHandle.

```
FactHandle messageHandle = kSession.insert(message);  
message.setMessage(message: "new update");  
kSession.update(messageHandle, message);  
kSession.delete(messageHandle);  
kSession.fireAllRules();
```



# DROOLS - Facts

- Unos, brisanje i modifikovanje činjenica iz samih pravila moguće je upotrebom DRL keywords:
  - insert
  - delete
  - modify, (update metodu ne raditi !)



# DROOLS - šta nam omogućava?

Forward chaining

Backward chaining

Templates

**Complex Event Processing**



# Pitanja?

