

EKSPERTNI SISTEMI

Razvoj - II deo

Bojan Tomić

E-mail: bojan.tomic@fon.rs

Drools – osnovne karakteristike

- Sajt: <https://www.drools.org/>
- Grupa alata, ali jedinstvena platforma za razvoj sistema zasnovanih na pravilima
 - **Drools Expert** (skup desktop alata za razvoj ES - rule engine)
 - Drools Workbench (skup web alata za razvoj ES - Business Rules Manager)
 - Drools Fusion (obrada događaja)
 - jBPM (poslovni procesi)
 - Opta Planner (automatizovano planiranje)

Drools - osnovne karakteristike

- Napisan u Javi, besplatan, open source
- Koristi **pravila** i **Java objekte** za predstavljanje znanja
- Koristi ulančavanje unapred
- Razvoj, puštanje u rad i testiranje svih vrsta ES
- Strategije za rešavanje konflikta:
 - prioritet pravila (saliency)
 - Svako pravilo se izvršava samo jednom (no-loop, lock-on-active)

Drools Expert – osnovne karakteristike

- Baza znanja se čuva u DRL formatu (format karakterističan samo za ovaj shell)
- Alati za razvoj i testiranje dati kao Eclipse plug-in
- Nema gotov GUI za krajnjeg korisnika
- Poseban jezik za pisanje pravila
- Dat je u formi framework-a (skupa gotovih klasa)
- Unos činjenica u vidu Java objekata
- Veoma lako se integriše u bilo koju Java aplikaciju (direktan pristup Java klasama)

Drools Expert - prednosti

- Prednosti
 - Relativno brzo upoznavanje sa shell-om ako je poznata Java i Eclipse
 - Mogućnost unosa činjenica iz bilo kog izvora preko Java objekata
 - Lako povezivanje sa Java aplikacijama
 - Lako ugrađivanje u Java aplikacije
 - Brzo izvršavanje ([Rete algoritam](#) za ulančavanje unapred [Charles Forgy, 1982])
 - Može da se koristi i za velike baze znanja
 - Razvoj svih vrsta ES (desktop i distribuiranih)

Drools Expert - mane

- Mane
 - Poseban jezik za pisanje pravila (veoma sličan Javi)
 - Nema već gotov korisnički GUI (ni desktop ni web)
 - Nema već gotove module za povezivanje sa bazama podataka ili fajlovima, već je potrebno koristiti odgovarajuća rešenja za Java objekte

Drools Expert - sintaksa

- Pravila se nalaze u DRL fajlovima (DRL - Drools Rule Language)
- Jedan fajl može da sadrži više pravila, a ES se može sastojati iz više DRL fajlova.
- Struktura DRL fajla:

```
package ime_paketa_za_pravila;
```

```
import java_klasa_1;
```

```
import java_klasa_2;
```

```
...
```

```
//pravila
```

Drools Expert - sintaksa

package

- Paketi služe za podjelu pravila na grupe u slučaju da je domenski problem složen
- Svaki DRL fajl mora da pripada tačno jednom paketu (package naredba je obavezna)
- To **nisu Java paketi** tj. ne moraju da odgovaraju imenu foldera
- Primer:

```
package finansijska_pravila;
```


Drools Expert - sintaksa

`import`

- Import naredba je **ista kao u Javi** i služi za uvoženje svih Java klasa koje će se koristiti u okviru pravila
- Primer:

```
import finansije.Izvestaj;
```

Drools Expert - sintaksa

- Komentari se pišu kao u Javi sa dodatkom još jednog specijalnog znaka za komentar - #
- Primeri:

```
//Jednolinijski komentar
```

```
# Jednolinijski komentar - moze i ovako
```

```
/*
```

```
Viselinijijski
```

```
komentar
```

```
*/
```

Drools Expert - sintaksa

- Pravila - sintaksa:

```
rule "naziv"  
    //atributi pravila  
when  
    //uslov(LHS-Left hand side)  
then  
    //zaključak(RHS-Right hand side)  
end
```

Drools Expert - sintaksa

- Naziv pravila može biti bilo kakav niz slova, samo mora da bude pod navodnicima
- Atributi pravila su opcioni i utiču na ponašanje pravila (prioritet, period važenja itd.)
- Uslov pravila (LHS) sadrži jedan ili više logičkih iskaza koji liče na logičke izraze u Javi
- Zaključak pravila se daje u formi običnih Java naredbi

Drools Expert - sintaksa

- Neki od atributa:
 - **salience** (int) - broj koji predstavlja prioritet pravila (podrazumavana vrednost je 0)
 - **no-loop** (boolean) - ako se izvršavanjem pravila izmeni neka činjenica ona može da izazove rekurzivno okidanje istog pravila; to se sprečava postavljanjem vrednosti ovog atributa na true (podrazumavana vrednost je false)
 - **agenda-group** (String) - naziv grupe pravila kojoj pravilo pripada - po potrebi, mogu se aktivirati samo određene grupe pravila (podrazumevana vrednost je MAIN)

Drools Expert - sintaksa

- Još neki od atributa:
 - **lock-on-active (boolean)** još jača verzija no-loop komande (ako neko drugo pravilo menja činjenicu)
 - **date-effective (String)** - datum napisan u formi String-a koji označava vremenski trenutak pre koga se pravilo ne smatra za aktivno
 - **date-expires (String)** - datum napisan u formi String-a koji označava vremenski trenutak posle koga se pravilo ne smatra za aktivno

Drools Expert - sintaksa

- Primer DRL fajla sa jednim pravilom - “svi muškarci oko 40. godine imaju krizu srednjeg doba”:

```
package psiholoska_pravila.krize;
```

```
import osnovni_podaci.Osoba;
```

```
rule "Kriza srednjeg doba"
```

```
when
```

```
o: Osoba (pol == 'M', godine == 40)
```

```
then
```

```
System.out.println(o.getIme() + " - nastupa kriza  
srednjeg doba");
```

```
end
```

Drools Expert - sintaksa

- Klasa Osoba (Java klasa) izgleda ovako:

```
package osnovni_podaci;  
public class Osoba {
```

```
    private String ime;  
    private int godine;  
    private char pol;
```

```
    public int getGodine() {return godine;}  
    public void setGodine(int godine) {this.godine =  
                                         godine;}  
    public String getIme() {return ime;}  
    public void setIme(String ime) {this.ime = ime;}  
    public char getPol() {return pol;}  
    public void setPol(char pol) {this.pol = pol;}  
}
```


Drools Expert - sintaksa

- Činjenice su obični Java objekti (klase)
- Ovi objekti **moraju da zadovoljavaju JavaBeans standard** tj. da imaju:
 - **get** metodu za preuzimanje vrednosti svakog atributa
 - **set** metodu za postavljanje vrednosti svakog atributa
- Na primer, klasa String ne može da bude činjenica (jer nema get i set metode)
- JBoss Rules automatski poziva ove get i set metode da bi preuzeo podatke iz objekata

Drools Expert - sintaksa

- U formiranju premise se mogu koristiti standardni logički Java izrazi (operatori “&&” i “||”, ali ne i “!”)
- Prethodno pravilo je moglo da se napiše i ovako:

```
rule "Kriza srednjeg doba"  
when  
    o: Osoba ((pol == 'M') && (godine == 40))  
then  
    System.out.println(o.getIme() + " - nastupa kriza  
        srednjeg doba");  
end
```

Drools Expert - sintaksa

- Pravilo sa složenom premisom:

```
rule "Resenje krize srednjeg doba"  
when  
    o: Osoba (pol == 'M', godine == 40)  
    a: Auto (marka == "Porsche", boja = "crvena")  
then  
    System.out.println(a.getMarka()+" moze da  
        pomogne da "+ o.getIme()+" prebrodi krizu");  
end
```

Drools Expert - sintaksa

- Primer pravila sa atributom za prioritet:

```
rule "Kriza srednjeg doba"
```

```
salience 10
```

```
when
```

```
o: Osoba (pol == 'M', godine == 40)
```

```
then
```

```
System.out.println(o.getIme()+" - nastupa kriza  
srednjeg doba");
```

```
end
```

```
rule "Resenje krize srednjeg doba"
```

```
when
```

```
o: Osoba (pol == 'M', godine == 40)
```

```
a: Auto (marka == "Porsche", boja = "crvena")
```

```
then
```

```
System.out.println(a.getMarka()+" moze da  
pomogne da "+ o.getIme()+" prebrodi krizu");
```

```
end
```

Drools Expert - sintaksa

- Pravila mogu u svom THEN delu da menjaju neke činjenice
- Posle promene bilo koje činjenice, mora se pozvati **update** metoda:

```
rule "Pera je musko"  
when  
    o: Osoba (ime == "Pera", pol == 'N')  
then  
    o.setPol('M');  
    update(o);  
end
```

Drools Expert - sintaksa

- U ovom shell-u postoje i posebni operatori
- Operator “postoji” - **exists**

```
rule "Da li postoji neki muskarac sa 40 godina"  
salience 11  
when  
    exists Osoba (pol == 'M', godine == 40)  
then  
    System.out.println("Postoje muskarci sa 40  
        godina");  
end
```

Drools Expert - sintaksa

- Operator “ako je svaki” - **forall**

```
rule "Da li su sve osobe muskarci"  
when  
    forall (Osoba (pol == 'M'))  
then  
    System.out.println("Sve osobe su muskarci");  
end
```

Drools Expert - sintaksa

- Puštanje ES u rad iz nekoliko koraka:
 - 1) Učitavanje baze znanja (iz DRL fajla)
 - 2) Kreiranje činjenica (Java objekata)
 - 3) Unošenje činjenica u radnu memoriju (dodavanje objekata u radnu memoriju)
 - 4) Pokretanje ES

Učitavanje baze znanja je **vremenski intenzivno**

Optimizacija - baza znanja može da se učitava iz DRL fajla i serijalizuje kao Java objekat

Drools Expert - sintaksa

- Klase koje se pri tom koriste su već gotove:
 - `KieContainer` (baza znanja Drools v 6.0 i noviji) -
`KnowledgeBase` (Drools 5.x)
 - `KieSession` (radna memorija v 6.0 i noviji) -
`StatefulKnowledgeSession` (v 5.x)
 - `KieServices` (učitava pravila iz fajla v6.0 i noviji) -
`KnowledgeBuilder` (v 5.x)

Drools Expert – primer 1

Ajnštajnova pitalica

- 4 igrača golfa stoje u liniji jedan pored drugog spremajući se da započnu igru;
- Svi igrači imaju pantalone različite boje; jedan od njih nosi crvene pantalone;
- Igrač desno od Freda nosi plave pantalone;
- Joe je drugi u redu;
- Bob ima karirane pantalone;
- Tom nije ni na 1. ni na 4. poziciji i ne nosi narandžaste pantalone;
- Pitanje je: u kom redosledu će igrači započeti igru i koje boje su pantalone svakog od njih?

Drools Expert

Ajnštajnova pitalica

- Ovaj problem može da se reši korišćenjem pravila
- Bitno je da se svaki uslov zadatka predstavi u vidu pravila
- Posle toga, dovoljno je pustiti program i Drools Expert će sam da reši pitalicu
- [Ovo nije ES](#), već samo ilustruje sintaksu i način funkcionisanja Drools Expert shell-a
- Sličan problem je i problem formiranja rasporeda sedenja (“seating problem”) koji se koristi za merenje brzine rada shell-ova

Drools Expert

Ajnštajnova pitalica

- Činjenice – klasa Golfer (Java klasa)

```
public class Golfer {  
  
    private int redosled = 0;  
    private String ime;  
    private String bojaPantalona = "nepoznata";  
  
    //Javne Get i Set metode za sva polja  
    //toString metoda  
  
}
```

Drools Expert

Ajnštajnova pitalica

- “Joe je na 2. poziciji”

```
rule "Joe je na poziciji 2"  
when  
    g : Golfer (ime == "Joe", redosled == 0)  
then  
    g.setRedosled(2);  
    update ( g );  
  
end
```

Drools Expert

Ajnštajnova pitalica

- “Bob ima karirane pantalone”

```
rule "Bob ima karirane pantalone"  
when  
    g:Golfer(ime=="Bob",bojaPantalona=="nepoznata")  
then  
    g.setBojaPantalona("karirane");  
    update ( g );  
end
```

Drools Expert

Ajnštajnova pitalica

- “Igrač sa Fredove desne strane nosi plave pantalone”

```
rule "Igrač sa Fredove desne strane nosi plave  
pantalone"
```

```
when
```

```
  g: Golfer (ime == "Fred", redosled > 0,  
             redosledFred : redosled)  
  g1 : Golfer (redosled == (redosledFred+1),  
              bojaPantalona == "nepoznata")
```

```
then
```

```
  g1.setBojaPantalona("plava");  
  update ( g1 );
```

```
end
```

Drools Expert

Ajnštajnova pitalica

- “Jedan od igrača nosi crvene pantalone”

```
rule "Jedan od golfera nosi crvene pantalone"  
when  
    exists Golfer (bojaPantalona == "plava")  
    exists Golfer (bojaPantalona == "karirane")  
    exists Golfer (bojaPantalona == "narandzasta")  
    g : Golfer (bojaPantalona == "nepoznata")  
then  
    g.setBojaPantalona("crvena");  
    update ( g );  
  
end
```


Drools Expert

Ajnštajnova pitalica

- “Igrač koji ima narandžaste pantalone nije Tom”

```
rule "Igrac koji ima narandzaste pantalone nije  
    Tom"  
when  
    exists Golfer (bojaPantalona == "plava")  
    exists Golfer (bojaPantalona == "karirane")  
    g : Golfer (ime != "Tom", bojaPantalona ==  
        "nepoznata")  
then  
    g.setBojaPantalona("narandzasta");  
    update ( g );  
end
```

Drools Expert

Ajnštajnova pitalica

- “Tom nije na četvrtoj poziciji” (onda je možda na trećoj)

```
rule "Tom nije na 1. niti na 4. poziciji (možda je
    na 3.)"
when
    exists Golfer (ime != "Tom", redosled == 2)
    g : Golfer (ime == "Tom", redosled == 0)
then
    g.setRedosled(3);
    update ( g );
end
```

Drools Expert

Ajnštajnova pitalica

- “Tom nije na četvrtoj poziciji” (onda je možda na drugoj)

```
rule "Tom nije na 1. niti na 4. poziciji (možda je
na 2.)"
when
    exists Golfer (ime != "Tom", redosled == 3)
    g : Golfer (ime == "Tom", redosled == 0)
then
    g.setRedosled(2);
    update ( g );
end
```

Drools Expert

Ajnštajnova pitalica

- “Fred ne može biti na četvrtoj poziciji jer ima nekoga sa desne strane”

```
rule "Fred ne moze biti na poziciji 4 jer mora da  
ima nekoga sa desne strane"
```

```
when
```

```
  exists Golfer (ime != "Fred", redosled == 3)
```

```
  exists Golfer (ime != "Fred", redosled == 2)
```

```
  g : Golfer (ime == "Fred", redosled == 0)
```

```
then
```

```
  g.setRedosled(1);
```

```
  update ( g );
```

```
end
```

Drools Expert

Ajnštajnova pitalica

- “Neko je onda i na četvrtoj poziciji”

```
rule "Neko je i na 4. poziciji"  
when  
    exists Golfer (redosled == 3)  
    exists Golfer (redosled == 2)  
    exists Golfer (redosled == 1)  
    g : Golfer (redosled == 0)  
then  
    g.setRedosled(4);  
    update ( g );  
end
```

Drools Expert – primer 2

ES za pomoć vozaču ako auto neće da upali

- **Ekspert:** majstor iz AMSS
- **Korisnik:** vozač
- ES zamenjuje eksperta kada su u pitanju najlakši problemi koje i korisnik sam može da otkloni
- **Moguća rešenja:**
 - Prazan akumulator
 - Neispravan anlaser
 - Neispravan električni sistem za paljenje
 - Prazan rezervoar
 - Neispravna pumpa za gorivo
 - Neispravna kontakt brava

Drools Expert

ES za pomoć vozaču ako auto neće da upali

- U razgovoru sa vozačem koji ima ovaj problem, majstor iz AMSS obično postavlja sledeća pitanja:
 - Da li auto hoće da vergla?
 - Da li rade farovi i sirena?
 - Da li rade lampice na komandnoj tabli?
 - Da li u autu ima goriva?
 - Kada pritiskate gas i verglate u isto vreme, da li se pojavljuje miris benzina u autu?

Drools Expert

ES za pomoč vozaču ako auto neće da upali

- Činjenice – klasa Automobil (Java klasa):

```
public class Automobil {  
  
    private String verglanje = "nepoznato";  
    private String mirisBenzina = "nepoznato";  
    private String radeFarovi = "nepoznato";  
    private String radeLampice = "nepoznato";  
    private String prazanRezervoar = "nepoznato";  
  
    private String uzrokProblema = "nepoznato";  
    private String resenje = "nepoznato";  
  
    //Javne Get i Set metode za sva polja  
    //toString metoda  
}
```


Drools Expert

ES za pomoć vozaču ako auto neće da upali

- “Ako auto vergla sporo, akumulator je prazan”

```
rule "Auto vergla sporo"  
lock-on-active true  
when  
    a: Automobil(verglanje == "Da, ali sporo")  
then  
    a.setUzrokProblema("Prazan akumulator");  
    a.setResenje("1. Izvaditi i napuniti  
    akumulator\n"+  
    "2. Upaliti auto uz pomoc kablova i jos jednog  
    auta\n"+  
    "3. Upaliti auto na 'gurku'");  
    update(a);  
end
```

Drools Expert

ES za pomoć vozaču ako auto neće da upali

- “Ako auto vergla, oseća se miris benzina i nije prazan rezervoar – loša elektriika”

```
rule "Auto vergla i oseca se miris benzina, nije prazan rezervoar"  
lock-on-active true  
when  
    a: Automobil(verglanje == "Da", mirisBenzina == "Da", prazanRezervoar == "Ne")  
then  
    a.setUzrokProblema("Los elektricni sistem (prekid u struji)");  
    a.setResenje("1. Pozvati auto-elektricara");  
    update(a);  
end
```

Drools Expert

ES za pomoč vozaču ako auto neće da upali

- “Auto vergla, ne oseća se benzin i rezervoar je prazan – napuniti rezervoar”

```
rule "Auto vergla, ne oseca se miris benzina i
    rezervoar je prazan"
lock-on-active true
when
    a: Automobil(verglanje == "Da", mirisBenzina ==
        "Ne", prazanRezervoar == "Da")
then
    a.setUzrokProblema("Prazan rezervoar");
    a.setResenje("1. Sipajte gorivo i pripazite da
        kod sledeceg paljenja auto stoji na ravnom");
    update(a);
end
```

Drools Expert

ES za pomoč vozaču ako auto neće da upali

- “Auto vergla, ne oseća se benzin i rezervoar je pun – neispravna pumpa za gorivo”

```
rule "Auto vergla, ne oseca se miris benzina i  
rezervoar nije prazan"
```

```
lock-on-active true
```

```
when
```

```
  a: Automobil(verglanje == "Da", mirisBenzina ==  
    "Ne", prazanRezervoar == "Ne")
```

```
then
```

```
  a.setUzrokProblema("Neispravna pumpa za  
gorivo");
```

```
  a.setResenje("1. Morate da odete kod auto-  
mehanicara da zameni pumpu za gorivo");
```

```
  update (a);
```

```
end
```

Drools Expert

ES za pomoč vozaču ako auto neće da upali

- “Auto ne vergla, rade farovi i lampice – neispravan anlaser”

```
rule "Auto ne vergla, rade farovi, rade lampice"  
lock-on-active true  
when  
    a: Automobil(verglanje == "Ne", radeFarovi ==  
        "Da", radeLampice == "Da")  
then  
    a.setUzrokProblema("Neispravan anlaser");  
    a.setResenje("1. Probajte da upalite auto na  
        gurku\n"+  
        "2. Idite kod auto-mehanicara ili auto-  
        elektricara da zameni anlaser");  
    update (a);  
end
```

Drools Expert

ES za pomoč vozaču ako auto neće da upali

- “Auto ne vergla, rade farovi, ali ne rade lampice – loša kontrakt brava”

```
rule "Auto ne vergla, rade farovi, ne rade  
    lampice"  
lock-on-active true  
when  
    a: Automobil(verglanje == "Ne", radeFarovi ==  
        "Da", radeLampice == "Ne")  
then  
    a.setUzrokProblema("Neispravna kontakt brava");  
    a.setResenje("1. Idite kod auto-mehanicara ili  
        auto-elektricara da zameni bravu");  
    update(a);  
end
```

Drools Expert

ES za pomoć vozaču ako auto neće da upali

- “Auto ne vergla, ne rade farovi, ne rade lampice – prazan akumulator”

```
rule "Auto ne vergla, ne rade farovi, ne rade  
lampice"
```

```
lock-on-active true
```

```
when
```

```
  a: Automobil(verglanje == "Ne", radeFarovi ==  
    "Ne", radeLampice == "Ne")
```

```
then
```

```
  a.setUzrokProblema("Prazan akumulator");  
  a.setResenje("1. Izvaditi i napuniti  
    akumulator\n"+  
    "2. Upaliti auto uz pomoc kablova i jos jednog  
    auta\n"+  
    "3. Upaliti auto na 'gurku'");
```

```
  update(a);
```

```
end
```

Mehanizam za objašnjavanje

- Objašnjenje ES
 - KAKO - kako je ES došao do zaključka, korak po korak
 - ZAŠTO - zašto ES postavlja korisniku neko pitanje
- Tehnike za objašnjavanje
 - Učaujeni tekst (canned text)
 - Trag izvršavanja pravila (rule trace)
- Povećavaju nivo poverenja korisnika u ES
- Objašnjenja moraju da budu prilagođena korisniku

JEFF mehanizam za objašnjavanje

- JEFF (Java Explanation Facility Framework)
- Besplatan, open-source Java framework
- <http://sourceforge.net/projects/jeff/>
- Pruža objašnjenje “KAKO” je ES došao do zaključka
- Tehnike **traga izvršavanja pravila i učeurenog teksta**
- Objašnjenje u PDF, XML ili TXT formatu
- Rečenice nalik na rečenice govornog jezika
Mogućnost unošenja slika i tabela u objašnjenje

JEFF mehanizam za objašnjavanje

- Kontekst poruke
 - WARNING, ERROR, POSITIVE, NEGATIVE, INFORMATIONAL...
- Tagovi za svaki deo objašnjenja