Expert systems – design and development part 2

Bojan Tomić

E-mail: bojan.tomic@fon.rs

Drools

- Link: http://www.jboss.org/drools
- Group of tools - a unique platform rule-based systems development
  - Drools Expert (desktop IDE - rule engine)
  - Drools Guvnor (web-based IDE - Business Rules Manager)
  - Drools Flow (business processes)
  - Drools Fusion (event processing)
  - Drools Planner (automated planner)

Drools Expert

- Written in Java, free, open-source
- Uses rules and Java objects for knowledge representation
- Forward chaining
- Development of all types of ES
- Conflict resolution strategies:
  - Rule importance (salience)
  - Each rule can execute only once (no-loop, lock-on-active)

Drools Expert

- Knowledge base - DRL files
- Development tools as Eclipse plug-in
- No generic end-user GUI
- Special language for writing rules (DRL)
- Core is a Java framework
- Facts in the form of Java objects
- It easily integrates into any Java application (direct access to Java classes)

Drools Expert - advantages

- Advantages
  - Short learning curve if familiar with Java and Eclipse
  - The ability to input facts from any source via Java objects
  - Easily connect/integrate with Java applications
  - Fast execution (Rete algorithm for forward chaining [Charles Forgy, 1982])
  - It can also be used for large knowledge base
  - Development of all types of ES (desktop and distributed)

Drools Expert - disadvantages

- Disadvantages
  - Special language for writing rules (DRL - much like Java)
  - No predefined end-user GUI (either desktop or web)
  - No ready-made modules to connect to databases or files – one has to use other Java frameworks or libraries

Drools Expert - syntax

- Rules stored in textual DRL files (DRL - Drools Rule Language)
- One file can contain multiple rules, the ES can consist of multiple DRL files.
- DRL file structure:

```
package rule_package_name;

import java_class_1;
import java_class_2;
...

//rules
```

Drools Expert - syntax

package

- Packages are used to divide the rules into groups if the domain is complex
- Each DRL file must belong to exactly one package (package command is required)
- They are not Java packages ie. not necessarily correspond to the folder name
- Example:

```
package financial_rules;
```

Drools Expert - syntax

import

- Import command is the same as in Java and is used for importing all the Java classes that will be used within the rules
- Example:

```
import finances.Report;
```

Drools Expert - syntax

- Comments are written as in Java with the addition of another special comment comment - #
- Examples:

```
//One line comment

# One line comment

/*
Multiple line
comment
*/
```

Drools Expert - syntax

- Rules - syntax:

```
rule "name"
     //rule attributes
when
     //condition(LHS-Left hand side)
then
     //conclusion(RHS-Right hand side)

end
```

Drools Expert - syntax

- Rule name can be any string, just has to be in quotes
- Rule attributes are optional and influence the behavior of rules (priority, period of validity etc.).
- Rule condition (LHS) contains one or more logical statements that resemble the logical expressions in Java
- Rule conclusion is in the form of ordinary Java commands

Drools Expert - syntax

- Important rule attributes:
  - salience (int) - number that represents rule importance (default value is 0)
  - no-loop (boolean) - if the execution of rules amend a fact it can cause recursive execution of the same rule; this is prevented by setting the value of this attribute to true (default value is false)
  - agenda-group (String) - name of a rule group to which the rule belongs - if necessary, only certain rule groups can be activated (the default is MAIN)

Drools Expert - syntax

- Important rule attributes:
  - lock-on-active (boolean) more effective version of the no-loop command (if another rule changes some important fact)
  - date-effective (String) - the date written in the form of a String denoting the time before which the rule is not considered to be active
  - date-expires (String) - the date written in the form of a String denoting the time after which the rule is not considered to be active

Drools Expert - syntax

- Example DRL file with one rule - "all men of the age of 40 have a mid-life crisis":

```
package psychology.crisis;

import data.Person;

rule "Mid-life crisis"
when
   p: Person (gender == 'M', age == 40)
then
   System.out.println(p.getName()+" - mid-life
     crisis starts now");
end
```

Drools Expert - syntax

- Class Person (Java class):

```
package data;
public class Person {

private String name;
private int age;
private char gender;

public int getAge() {return age;}
public void setAge(int age) {this.age = age;}
public String getName() {return name;}
public void setName(String name) {this.name =
  name;}
public char getGender() {return gender;}
public void setGender(char gender) {this.gender =
  gender;}
}
```

Drools Expert - syntax

- The facts are plain Java objects (classes)
- They must meet the JavaBean standard, ie. have get/set methods for each attribute
- For example, the String class cannot be used by itself as a fact (as there are no get and set methods)
- Drools Expert automatically calls the get and set methods to retrieve attribute values

Drools Expert - syntax

- In the conditional part of a rule you can use standard Java logical expressions (operators "&&" and "||", but not "!")
- The previous rule could be written like this:

```
rule "Mid-life crisis"
when
   p: Person ((gender == 'M')&&(age == 40))
then
   System.out.println(p.getName()+" - mid-life
     crisis starts now");
end
```

Drools Expert - syntax

- Rule with a complex condition:

```
rule "Mid-life crisis - solution"
when
   p: Person (gender == 'M',age == 40)
   p2: Person (gender == 'F',age > 25,age < 30)
then
   System.out.println(p2.getName()+" can help "+
     p.getName()+" overcome the mid-life crisis");
end
```

Drools Expert - syntax

- Rules with salience:

```
rule "Mid-life crisis"
salience 10
when
   p: Person (gender == 'M', age == 40)
then
   System.out.println(p.getName()+" - mid-life
     crisis starts now");
end

rule "Mid-life crisis - solution"
when
   p: Person (gender == 'M',age == 40)
   p2: Person (gender == 'F',age > 25,age < 30)
then
   System.out.println(p2.getName()+" can help "+
     p.getName()+" overcome the mid-life crisis");
end
```

Drools Expert - syntax

- Rules can also change some facts (in the THEN part)
- After changing any facts, the update method must be called:

```
rule "John is male"
when
   p: Person (name == "John", gender == 'N')
then
   p.setGender('M');
   update(p);
end
```

Drools Expert - syntax

- Drools Expert also has special operators
- Operator “exists”

```
rule "Are there any men age 40"
salience 11
when
   exists Person (gender == 'M', age == 40)
then
   System.out.println("There is at least one man
     of the age 40");
end
```

Drools Expert - syntax

- The “forall” special operator

```
rule "Is every person male"
when
   forall (Person(gender == 'M'))
then
   System.out.println("All persons are male");
end
```

Drools Expert - syntax

- Starting the ES in several steps:

1. Loading the knowledge base (from DRL files)
2. Creating facts (Java objects)
3. Entering the facts into the working memory (add objects to working memory)
4. Execute ES inference process

- Knowledge base loading is time-intensive. Optimization - a knowledge base can be loaded from DRL file and serialized as a Java object

Drools Expert - syntax

- Drools Expert predefined core classes:
  - RuleBase (Drools 4.x) - KnowledgeBase (Drools 5.x)
  - WorkingMemory (Drools 4.x) - StatefulKnowledgeSession (Drools 5.x)
  - PackageBuilder (Drools 4.x) - KnowledgeBuilder (Drools 5.x)
  - Package (Drools 4.x) - KnowledgePackage (Drools 5.x)

Drools Expert – Einstein's riddle

- 4 golfers standing in line next to each other getting ready to start the game
- All players have the pants of different colors, one of them is wearing red pants
- The player to the right of Fred has blue pants
- Joe is the second in line
- Bob has plaid pants
- Tom is not the first or the fourth in line and is not wearing orange pants;
- The question is in what order the players will start the game and the color of the pants of each of them?

Drools Expert – Einstein's riddle

- This problem can be solved by using rules
- It is important that each condition is presented in the form of rules
- After that, just let Drools Expert solve the guessing game
- This is not an ES
- A similar problem is the problem of forming the seating arrangements ("seating problem"), which is used to measure an ES shell's execution speeds

Drools Expert – Car malfunction diagnosis

- Expert: Car mechanic
- User: Car driver
- ES replaces an expert when it comes to the easiest problems that the user can solve by him/herself.
- Possible solution:
- Dead battery
- Faulty starter
- Faulty electrical ignition system
- Empty fuel tank
- Faulty fuel pump
- Faulty ignition

Drools Expert – Car malfunction diagnosis

- In a conversation with a driver who has this problem, the mechanic usually asks the following questions:
- Does the car crank?
- Are the lights and siren working?
- Do the lights on the dashboard ignite?
- Does the car have fuel?
- When you press the gas and crank at the same time, does a smell of gas appear in the car?

Explanation facility

- ES explanation
  - HOW - how the ES inferred some conclusion, step by step
  - WHY - why the ES asks the user some question
- Explanation techniques
  - Canned text
  - Rule trace
- Increase the users level of trust in ES conclusions
- Explanations must be adapted to the user knowledge level

JEFF

- JEFF (Java Explanation Facility Framework)
- Free, open-source Java framework
- http://sourceforge.net/projects/jeff/
- The “HOW” explanation
- Rule trace and canned text
- Explanation in PDF, XML and TXT formats
- Explanations with natural-language like sentences, images, data tables...