An Open Source Java Rules Engine

Eduardo Araújo Oliveira

Drools

O que é?

Drools is a business rule management system (BRMS) with a forward chaining inference based rules engine, more correctly known as a production rule system, using an enhanced implementation of the Rete algorithm.

3

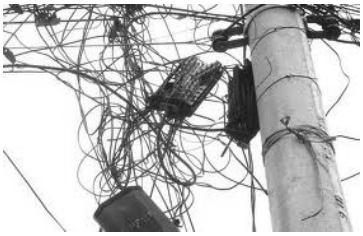
Drools

- The Business Logic integration Platform
- 5 Módulos integrados
- Desde 2001
- Semântica em 2011
- Jboss e Red Hat
- Open Source
- Plugin para Eclipse
- http://www.jboss.org/drools

4

Drools - motivação

- Implementa ferramentas para decisões complexas de negócio
- Problemas dos métodos tradicionais:
  - If-else (Código espaguete)
  - Alterações geralmente precisam de recompilação e redeploy
  - Não separa código de infraestrutura das regras de negócio



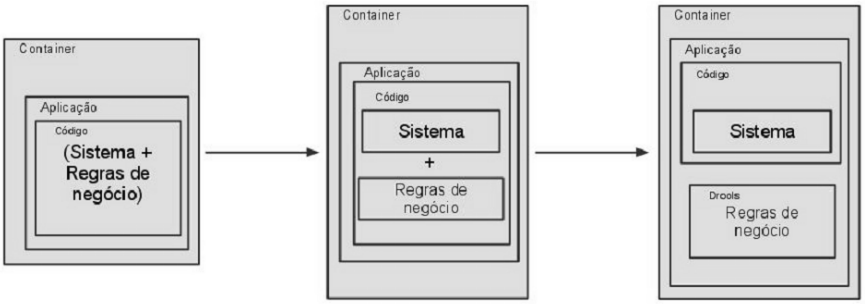
5

Drools - motivação

- Permite implementar a lógica de negócio de uma maneira mais declarativa
- Separa o conhecimento do código de infraestrutura
- Fornece diferentes ferramentas para cada tipo de lógica de negócio
  - Decisões
  - Processos de negócio
  - Eventos

6

Drools



7

Drools

Totalmente integrável com Java

- Integração transparente através de um plug-in do Eclipse;
- Pode ler classes Java nas condições das regras;
- Pode chamar métodos de Java na ação das regras;

8

Drools

Vantagens
- Fácil entendimento
- Maior facilidade de manutenção
- Desempenho razoável
- Requisitos traduzidos em regras
- Reutilização

9

Drools

java

```
public void checaMaiorDeIdade(Aluno aluno) {
   Date dataNascimento =
    aluno.getPessoaFisica().getPessoa().getDataNascimento();

   if (subtrairData(new Date(), dataNascimento) < IDADE_MINIMA) {
      aluno.getMatricula().invalidar();
   } else {
      aluno.getMatricula().validarIdade();
   }
}
```

drools

```
regra "Aluno deve ser maior de idade"
   quando
      O aluno em questão possui
         - idade menor que idade mínima
   então
      invalidar matrícula
fim
```

10

Drools

Vantagens das regras

- Regras rodam em uma camada acima do código nativo da aplicação
- Podem ser expressas em uma linguagem diferente, mais natural
- São compiladas para uma linguagem cujo o motor de regras as interpreta
- Linguagem mais próxima a linguagem natural
- Facilidade de manutenção e leitura
- Introduz a possibilidade do próprio analista de negócios realizar a autoria e manutenção das regras sabendo exatamente como serão interpretadas pelo sistema

11

Drools

Desvantagens

- Requer uma curva de aprendizado
  - Entender minimamente como funciona uma engine de regras (máquina de inferência)
  - As regras podem gerar recursão, que devem ser tratadas pelo desenvolvedor
  - Em casos de conflitos o desenvolvedor tem que escolher qual tratamento usar
- Consumo de memória

12

Drools License

- JBoss Rules - Apache License v.2.0

The Apache license is particularly business-friendly, and you can take the code and use it in pretty much any way you want, as long as you acknowledge that your product was 'built using Drools'. You don't have to publish your changes or additions (as another famous open source license, the GPL, requires you to do).

13

Drools

Módulos

14

Drools

- Módulos

Engine de regras

Linguagem para regras (DRL)

Tabelas de decisão (xls e cvs)

Linguagem específica do domínio (DSL)

Integrado ao Java

É o módulo principal, que compreende o motor de regras, linguagem de regras e a implementação do algoritmo de pattern matching.

http://www.jboss.org/drools/drools-expert.html

15

Drools

- Módulos

Workflows

BPMN

Editor gráfico do fluxograma

Extensível

Para criar, executar e monitorar processos de negócio

Adiciona as capacidades para lidar com BPM

http://www.jboss.org/drools/drools-flow.html

16

Drools

- Módulos

Processamento de Eventos Complexos (CEP)

- Eventos no tempo
- Para sistemas de:

– Detecção de fraudes

– Aprovação de crédito

Adiciona capacidades para realizar o CEP (Complex Event Processing).

http://www.jboss.org/drools/drools-fusion.html

17

Drools

- Módulos

BRMS (não só regras)

Repositório centralizado do conhecimento

Aplicação Web

Versionamento

Foco nas regras de negócio

Inclui a ferramenta para Gerência de Regras de negócio (BRMS).

http://www.jboss.org/drools/drools-guvnor.html

18

Drools

- Módulos

Planejamento automático

Problemas com restrições

Problemas como:

  – Escalas de empregados
  – Horário escolar
  – Caixeiro viajante

http://www.jboss.org/drools/drools-fusion.html

19

Drools

20

Conceitos Básicos

Fatos:

- São objetos que representam um determinado estado do domínio

Regras de negócio

- Toda regra é representada por dois elementos principais, seguindo a sintaxe:

quando

<condições (LHS)>

então

<ações (RHS)>

Conceitos Básicos

- Representam conhecimento com pares *condição-ação*
  - **Se** *condição* (ou *premissa* ou *antecedente*) ocorre **então** *ação* (*resultado*, *conclusão* ou *conseqüente*) deverá ocorrer.
- Regras de produção *produzem* novos fatos a partir dos fatos e regras da BC.
  - Esses novos fatos passam a fazer parte da BC

Conceitos Básicos

- Exemplo de regra:

```
package bank.model;
rule "basic rule"
when // condition
   Account( balance < 100 )
then // consequence
   System.out.println("Account balance is less than
   100");
end
```

Conceitos Básicos

O package funciona como um namespace

– Nomes de regras em um pacote tem que ser únicas

basic rule é o nome da regra

when indica a condição (premissa)

– LHS (Left Hand Side)

then indica a consequência da regra

– RHS (Right Hand Side)

// é usado para comentários

Conceitos Básicos

Várias condições

```
Account( balance == 200 )
Customer( name == "John" )
```

Variáveis nas regras

```
$account : Account( $type : type )
```

Tipos

– String

```
Customer( name matches "[A-Z][a-z]+" )
```

– Date

```
Account( dateCreated > "01-Jan-2008" )
```

– Boolean

```
Transaction( isApproved == true )
```

– Enum

```
Account( type == Account.Type.SAVINGS )
```

Comentários

```
#Comentário de única linha
//Comentário de única linha
/*Comentário de
várias linhas*/
```

Conceitos Básicos

Imports

```
import com.mycompany.mypackage.MyClass;
import com.mycompany.anotherPackage.*;
```

Variáveis Globais

Funções

```
function double calculateSquare(double value) {
return value * value;
}
```

Condição da regra

– And

```
Customer( name == "John", age < 26 )
```

– Or

```
Customer( name == "John" || age < 26 )
Customer( age < 26 || > 70 )
```

– Not

```
not Account( type == Account.Type.SAVINGS )
```

– Exists

```
exists Account( type == Account.Type.SAVINGS )
```

Conceitos Básicos

Quando todas as condições de uma regra são satisfeitas, a regra é ativada

Uma regra ativada é disparada, segundo a estratégia de resolução de conflito

A execução das regras podem ativar outras regras

O processo é repetido até que nenhuma regra seja ativada

Conceitos Básicos

Alguns comandos usados na consequência da regra

- update(objeto);
- insert(new Objeto());
- insertLogical(new Objeto());
- retract(objeto);
- drools.halt();
- drools.getRule().getName();
- kcontext.getKnowledgeRuntime().halt();

http://www.ibm.com/developerworks/java/library/j-drools/

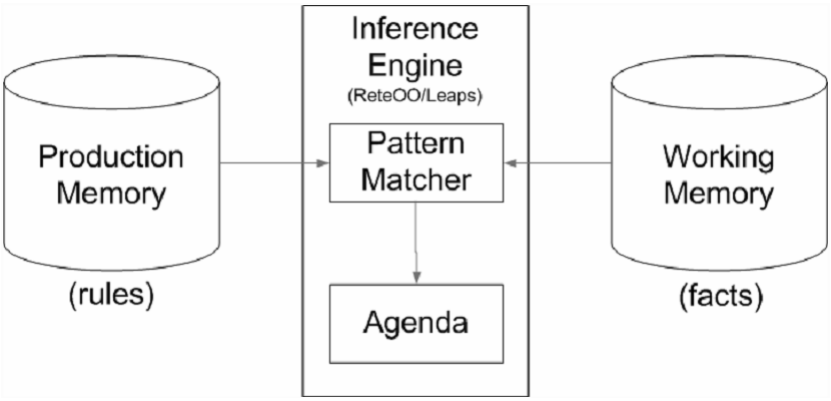
Conceitos Básicos

Alguns atributos das regras

– salience (prioridade)
  Default é 0
  salience 100
– no-loop
– date-effective
  date-effective "01-Jan-2011“
– date-expires
  date-expires "01-Jan-2011"
– duration
  duration 3000

http://www.ibm.com/developerworks/java/library/j-drools/

Como funciona?



30

Como funciona?

- Working Memory - Memória de Trabalho
  - É onde residem os fatos.
- Production Memory - Base de Conhecimento
  - É onde reside todo o conhecimento de negócio (regras).
- Pattern Matcher - Reconhecedor de padrões
  - Responsável por casar os fatos na memória de trabalho com as condições das regras e criar ativações a partir dos casamentos.
- Agenda
  - Responsável pela ordenação das ativações para execução.

Exemplos

http://imasters.com.br/artigo/12444/desenvolvimento/de_forca_ao_usuario_com_o_drools_parte_1/

http://imasters.com.br/artigo/12708/java/de_forca_ao_usuario_com_drools_parte_2/

http://imasters.com.br/artigo/15646/desenvolvimento/de-forca-ao-usuario-com-drools-parte-03/

códigos:
http://diegopacheco.svn.beanstalkapp.com/sandbox/trunk/JBossDrools-test/src/com/blogspot/diegopacheco/drools/

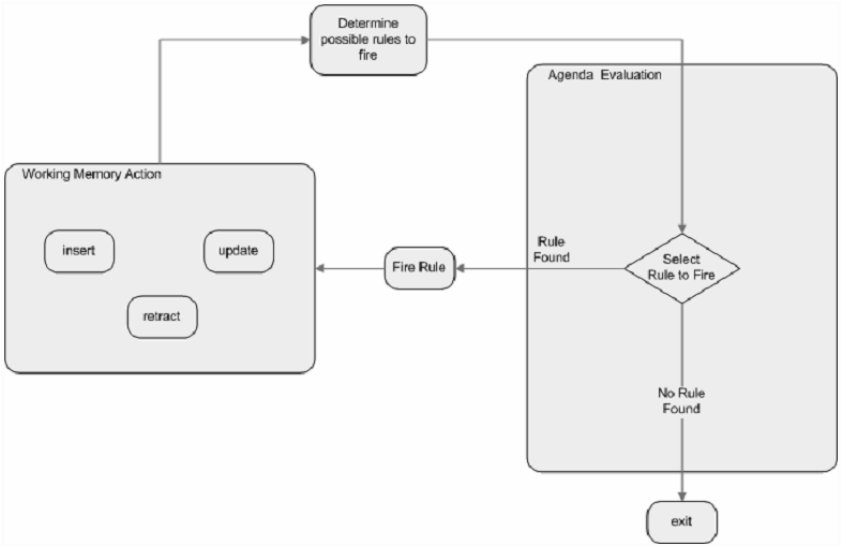
Sistemas de Produção

- Fatos: x, y
- Regras: x & y => p
- Encadeamento para a frente (*Forward chaining*)
  - *Dados x e y, derive então p*
- Encadeamento para trás (*Backward chaining*)
  - *p é verdade? Então verificarei x e y.*
  - Prolog

Módulos de Sistemas de Produção

- Base de Regras ou memória permanente
  - regras se-então e fatos conhecidos
- Memória de Trabalho ou temporária
  - percepções do agente, fatos iniciais e gerados a partir da BR pelo mecanismo de inferência
- Mecanismo (motor) de Inferência
  - determina o método de raciocínio utilizado (progressivo ou regressivo)
  - Executa a busca com casamento (unificação)
  - resolve conflitos e executa ações.

Basic RETE Network



35

Funcionamento de um sistema de regras de produção

- As regras seriam a memória duradoura
- Os fatos seriam a memória de trabalho
- Ao usarmos as regras sobre os fatos, são derivados novos fatos
- O ciclo se repete até que não haja mais fatos a derivar
- Conflito: várias regras podem ser disparadas ao mesmo tempo

Performance Characteristic

Sacrifices memory for speed

37

Exemplo

Working Memory...

```
Message message = new Message();
message.setMessage("Hello World");
message.setStatus(Message.HELLO);
ksession.insert(message);
```

38

Exemplo

Production Memory...

```
rule "Hello World"
  when
      m : Message( status == Message.HELLO )
  then
      m.setMessage( "Goodbye cruel world" );
      m.setStatus( Message.GOODBYE );
  update( m );
end
```

39

Jboss Rules

A business rule is any bit of knowledge that can be expressed in the following format:

When 'something' is true, **Then** do 'this'.

Nothing more complicated than that.

JSR-94.

40

Jboss Rules


```
rule "One"
   #include attributes such as "salience" here...
   when
       #conditions
   then
       #actions
end
```

41

Jboss Rules

Examples of these rules are:

- When a football team wins a game, jump up and down and shout loudly
- When a staff member gets promoted, give them a pay rise of 10%
- When a person's salary is less than 30,000 dollars, apply a tax rate of 20%
- When somebody leaves the office before 4 pm, make sarcastic comment about 'taking a half-day vacation‘

42

Rule Engine

When not to use a rule engine

Don't use a rule engine if your application doesn't have much complexity.

Don't use a rule engine for the first time on a project that has strict deadlines or is high-profile

43

Rule Engine

When to use a rule engine

The business logic changes often

There are people who understand the business problem in great detail, but may not have the technical IT skills

The problem may be too complex for other solutions

You need an agile solution—rule engines allow you to easily change the business logic in an iterative manner

44

Exemplos

45

regrasCasa.dslr

```
package benevides

expander regrasCasa.dsl

rule "Ligar ar-condicionado do Rafael quando temperatura maior que 26 gra
    when
        A casa é do "Rafael"
        e a temperatura for maior que 26
    then
        ligar o ar-condicionado
end

rule "Desligar ar-condicionado do Rafael quando temperatura menor que 20
    when
        A casa é do "Rafael"
        e a temperatura for maior que {temperatura
    then
        desligar o ar-condicionado
end
```

46

Natureza.java

```
package benevides;

public class Natureza {

    public int temperatura;


    public void setTemperatura(int temperatura) {
        this.temperatura = temperatura;
    }

    public int getTemperatura() {
        return temperatura;
    }
}
```

ArCondicionado.java

```
package benevides;

public class ArCondicionado {

    private boolean ligado = false;

    private Casa casaInstalada;

    public ArCondicionado(Casa casa) {
        this.casaInstalada = casa;
    }

    public void desligar() {
        if (ligado) {
            System.out.println("Desligando ar-condicionado " +
                    "da casa do " + casaInstalada.getDono() );
            ligado = false;
        }
    }

    public void ligar() {
        if (!ligado) {
            System.out.println("Ligando ar-condicionado " +
                    "da casa de " + casaInstalada.getDono());
            ligado = true;
        }
    }

}
```

Casa.java

```
package benevides;

public class Casa {

    private ArCondicionado arCondicionado = new ArCondicionado(this);

    private String dono;

    public Casa(String dono) {
        this.dono = dono;
    }

    public String getDono() {
        return dono;
    }

    public ArCondicionado getArCondicionado() {
        return arCondicionado;
    }
}
```

49

```
package benevides

    rule "Ligar ar-condicionado do Rafael quando temperatura maior que 26 graus"
       when
          Casa( dono == "Rafael", arCondicionado: ArCondicionado  )
          Natureza( temperatura  > 26 )
       then
          arCondicionado.ligar();
    end

    rule "Desligar ar-condicionado do Rafael quando temperatura menor que 20 graus"
       when
          Casa( dono == "Rafael", arCondicionado: ArCondicionado  )
          Natureza( temperatura  < 20 )
       then
          arCondicionado.desligar();
    end

    rule "Ligar ar-condicionado do Frederico quando temperatura maior que 21 graus"
       when
          Casa( dono == "Frederico", arCondicionado: ArCondicionado  )
          Natureza( temperatura  > 21 )
       then
          arCondicionado.ligar();
    end

    rule "Desligar ar-condicionado do Frederico quando temperatura menor que 18 graus"
       when
          Casa( dono == "Frederico", arCondicionado: ArCondicionado  )
          Natureza( temperatura  < 18 )
       then
          arCondicionado.desligar();
    end
```

50

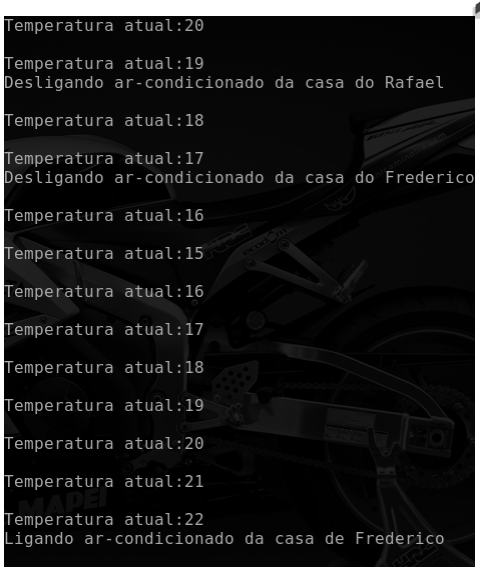
Para verificar o comportamento, vamos inserir na ***WorkingMemory*** do Drools as instâncias do objeto ***Casa*** (uma instância de cada cliente) e a cada alteração de temperatura, vamos atualizar a temperatura da instância da classe ***Natureza*** e também atualizar esta instância na ***WorkingMemory***.

51

Teste.java

```
public static void main(String[] args) throws Exception {
    Casa casaDoRafael = new Casa("Rafael");
    Casa casaDoFrederico = new Casa("Frederico");
    Casa casaDoBenevides = new Casa("João Gabriel");
    Natureza natureza = new Natureza();

    WorkingMemory wm = lerRegras();
    wm.insert(casaDoRafael);
    wm.insert(casaDoFrederico);
    wm.insert(casaDoBenevides);
    FactHandle handleNatureza = wm.insert(natureza);

    for (int temperatura = 30; temperatura > 15; temperatura--) {
        System.out.println("Temperatura atual:" + temperatura);

        natureza.setTemperatura(temperatura);
        wm.update(handleNatureza, natureza);
        wm.fireAllRules();

        System.out.println();
    }
    for (int temperatura = 15; temperatura <= 30; temperatura++) {
        System.out.println("Temperatura atual:" + temperatura);

        natureza.setTemperatura(temperatura);
        wm.update(handleNatureza, natureza);
        wm.fireAllRules();

        System.out.println();
    }
}
```

52



53

DSL

Domain Specific Languages

O motor de regras possui uma poderosa maneira de mapear sua linguagem nativa, expressa nos drl's, para uma linguagem mais natural, como mostrada nos primeiros exemplos.

Este mapeamento se dá através das Domain Specific Languages, ou dsl's.

dsl´s são expressas em arquivos .dsl, com uma sintaxe bastante simples, bem parecida com um arquivo .properties

54

DSL

Domain Specific Languages por exemplo:

```
[keyword]rule=regra
[keyword]when=quando
[keyword]then=então
[keyword]end=fim
[when]O Aluno em questão possui=a : Aluno($pf : pessoaFisica) pf : PessoaFisica($p :
pessoa) from $pf p : Pessoa($dataNasc: dataNascimento, $nome : nome) from $p
[when]- idade menor que {idadeMinima}=eval( subtraiData(dataAtual, $dataNasc) <
{idadeMinima} )
[then]Invalidar matrícula=mensagemMenorDeIdade($nome); a.getMatricula().invalidar();
```

55

DSL

A regra anterior poderia ser reescrita da seguinte maneira:

```
regra "deve ser maior de idade"
    salience 0
    ruleflow-group "validacao"
    quando
        O aluno em questão possui
            - idade menor que 18
    então
        Invalidar matrícula
fim
```

56

Drools + Protégé

Drools + Protégé

http://oogis.ru/component/option,com_remository/Itemid,34/func,fileinfo/id,15/lang,en/

DROOLS BASED SPATIAL SCENARIO SIMULATION PLUG-IN TO PROTEGE

DroolsTab is a tab plug-in to the open source ontology editor Protege (http://protege.stanford.edu). It uses the open source geo-information system Java library OpenMap (http://openmap.bbn.com) and the open source Java RETE rule engine Drools (http://labs.jboss.com/drools/).

57

Drools

An Open Source Java Rules Engine

Eduardo Araújo Oliveira