Anderson Neves

afrn@cin.ufpe.br

Roteiro

- **Sobre o Drools**
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- Infraestrutura
- Testes
- Exercício

Sobre o 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

Sobre o Drools

- Implementa ferramentas para decisões complexas de negócio
- Problemas dos métodos tradicionais:
  - If-else (Código espaguete)
  - Uma pequena alteração precisa de recompilação e redeploy
  - Não separa código de infraestrutura das regras de negócio
    - O que: requisitos de negócio
    - Como: algoritmo

Sobre o Drools

- 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

Sobre o Drools

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

Sobre o Drools

- Desvantagens
  - Não é a “bala de prata”
  - 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

Roteiro

- Sobre o Drools
- **Módulos**
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- Infraestrutura
- Testes
- Exercício

Módulos

- Engine de regras
- Linguagem para regras (DRL)
- Tabelas de decisão (xls e cvs)
- Linguagem específica do domínio (DSL)
- Integrado ao Java

Módulos

- Workflows
- BPMN
- Editor gráfico do fluxograma
- Extensível
- Para criar, executar e monitorar processos de negócio

Módulos

- Processamento de Eventos Complexos (CEP)
- Eventos no tempo
- Para sistemas de:
  - Detecção de fraudes
  - Aprovação de crédito

Módulos

- BRMS (não só regras)
- Repositório centralizado do conhecimento
- Aplicação Web
- Versionamento
- Foco nas regras de negócio

Módulos

- Planejamento automático
- Problemas com restrições
- Problemas como:
  - Escalas de empregados
  - Horário escolar
  - Caixeiro viajante

Roteiro

- Sobre o Drools
- Módulos
- **Drools Expert**
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- Infraestrutura
- Testes
- Exercício

Drools Expert

- 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
```

Drools Expert

- 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

Drools Expert

- 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*/
```

Drools Expert

- 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 )
```

Drools Expert

- Trabalhando com coleções
  - (Not) contains

```
$account : Account( )
Customer( accounts contains $account )
Customer( accounts not contains $account )
```

  -
  - (Not) memberOf

```
$customer : Customer($accounts : accounts)
Account( this memberOf $accounts )
Account(this memberOf $customer.accounts)
```

  -
  - From

```
$customer : Customer( )
Account( ) from $customer.accounts
```

Drools Expert

- 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

Drools Expert

- 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();

Drools Expert

- Alguns atributos das regras
  - salience (prioridade)
    - Default é 0

```
salience 100
salience ($account.balance * 5)
```

  - no-loop

```
no-loop
```

  - date-effective

```
date-effective "01-Jan-2011"
```

  - date-expires

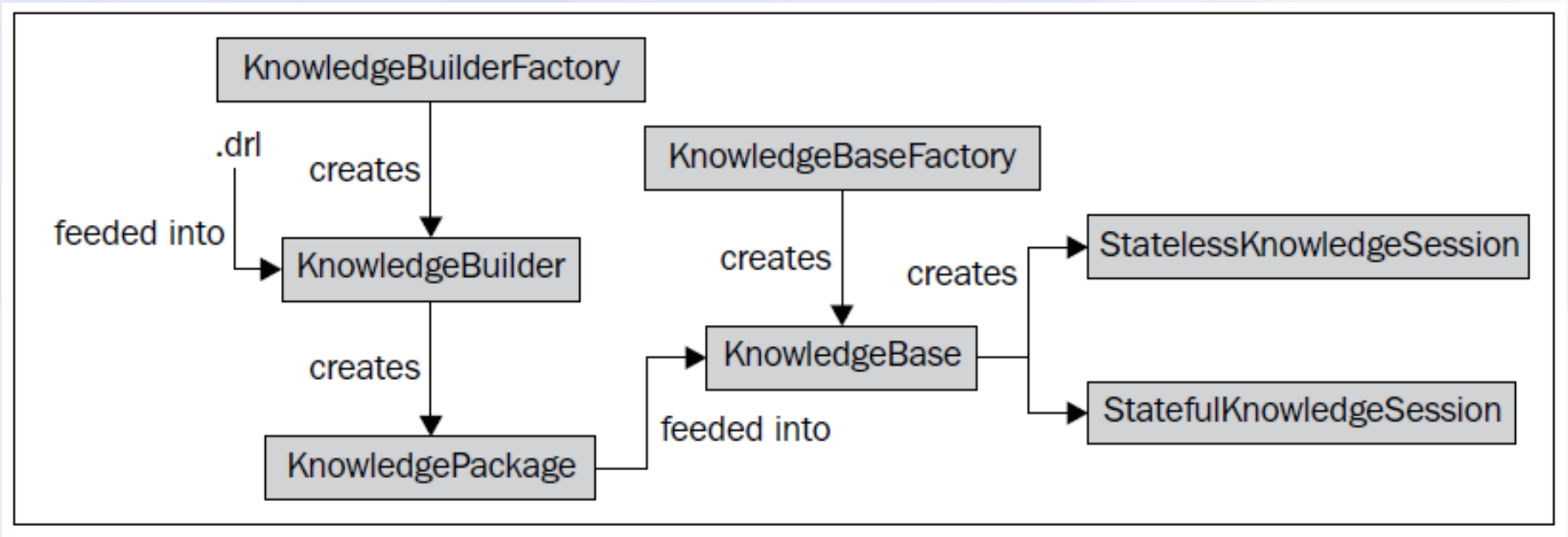
```
date-expires "01-Jan-2011"
```

  - duration

```
duration 3000
```

Drools Expert

- Parte da API Drools



Roteiro

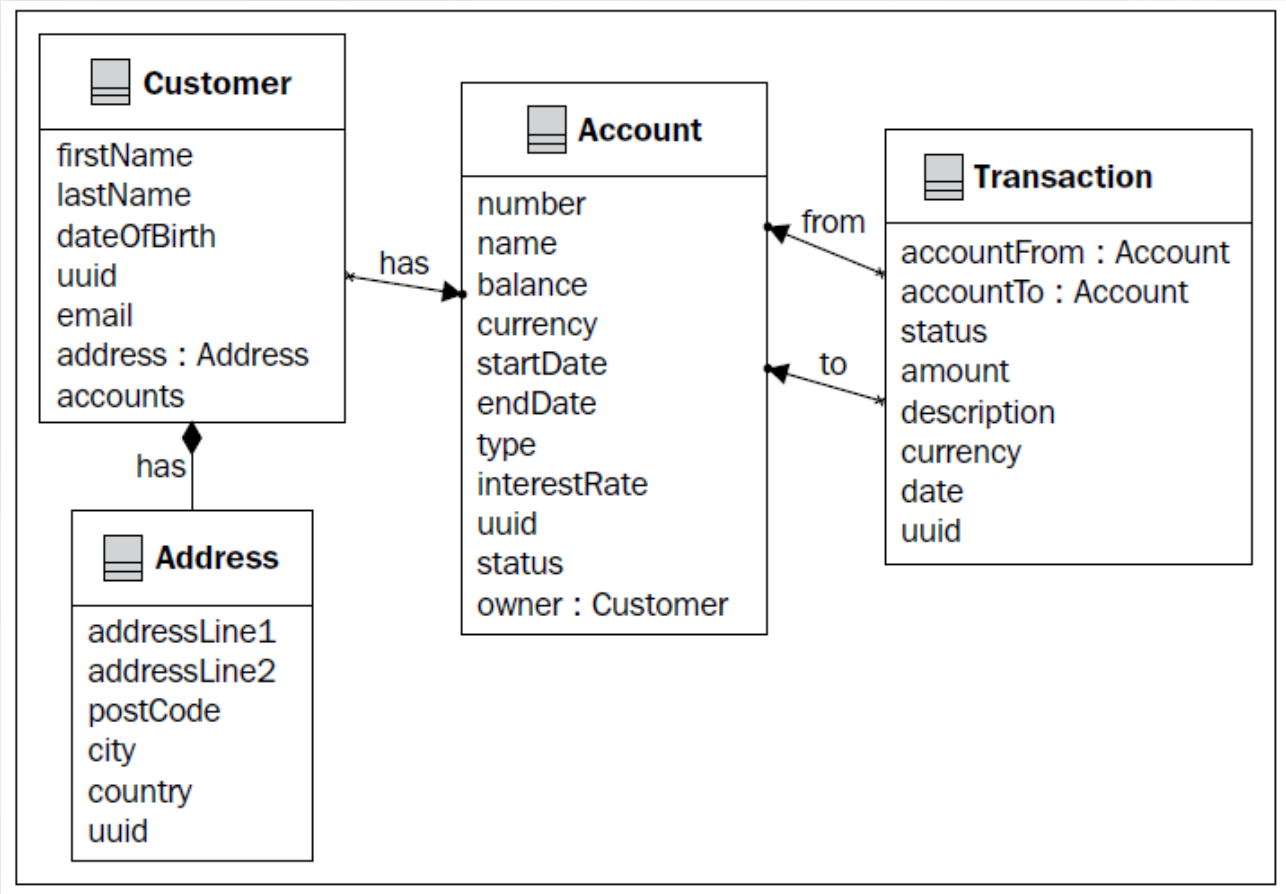
- Sobre o Drools
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - **Hello World**
- Aplicação bancária
- Infraestrutura
- Testes
- Exercício

Roteiro

- Sobre o Drools
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- **Aplicação bancária**
- Infraestrutura
- Testes
- Exercício

Aplicação Bancária

- Modelo



Roteiro

- Sobre o Drools
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- **Infraestrutura**
- Testes
- Exercício

Roteiro

- Sobre o Drools
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- Infraestrutura
- **Testes**
- Exercício

Roteiro

- Sobre o Drools
- Módulos
- Drools Expert
  - Regras DRL
  - API
  - Hello World
- Aplicação bancária
- Infraestrutura
- Testes
- **Exercício**

Exercício

- Criar regra suggestInvestment, onde ela verifica se o saldo de uma conta é maior que 100.
- Criar regra suggestLoan, onde ela verifica se o saldo de uma conta é menor que 0.
- Criar regra insertTransaction, que diz assim:
  - Sendo uma conta1 do tipo JOINT e uma conta2 do tipo SAVINGS, e as contas pertencem ao cliente Paulo Farias, então crie uma transação da conta1 para a conta2 cujo total é a metade do saldo da conta1 e coloque seu status como INIT. Insira a transação na memória de trabalho.
- Criar regra highTransaction, que diz assim:
  - Para uma transação com um total maior que 10000 e status INIT, então coloque seu status para PENDING e atualize o objeto.
  - No final desta regra coloque a seguinte linha:

```
report.addMessage("highTransaction" +
$transaction.getAccountFrom().getNumber() +
$transaction.getAccountTo().getNumber());
```