JBoss Rules（Drools）

1. JBoss Rules (Drools) とは

JBoss Rules は、jboss.org（http://www.jboss.org/）にて開発されている Drools（http://www.jboss.org/drools/）をベースにレッドハットがサポートを追加し製品化したプロダクトで SOA 基盤の JBoss Enterprise SOA Platform を構成している開発フレームワークです。

Drools は、ビジネスルールマネジメントシステム（BRMS）で、ReteOO と呼ばれる JVM に対応した Rete アルゴリズムをベースとするルールエンジンの実装です。また、JSR-94 の Java Rule Engine API も実装しています。Drools は、宣言型プログラミングを提供し、問題ドメインをドメイン特化言語（DSL:Domain Specific Language）によって解決します。

BRMS は、組織や企業で使用される決定ロジックであるビジネスルールを定義、デプロイ、実行、モニター、メンテナンスするために使用されるソフトウェアのことです。ビジネスルールには、戦術行動を決定するために使用されるポリシー、要件、および、条件文を含みます。BRMS によりビジネスルールをソースコードから外出しすることができます。また、システム開発者と業務の専門家の両者がビジネスルールを定義し、管理することができます。

JBoss Rules（Drools）を使用することによって、以下のようなメリットを得ることができます。

- ビジネスルールの変更が容易になる。
- ビジネスロジックをビジネスで使う用語や構文、図式（決定テーブル、木、スコアカード、フロー）を用いて表現できる。
- 業務の専門家がビジネスルールを設定できる。
- ビジネスルールの変更時のシステム開発者への負担が軽減される。

2. Drools のセットアップ

以下の環境における Drools のセットアップを説明します。

- Windows XP Professional SP2
- JDK 6 Update 13
- Eclipse 3.4（Eclipse IDE for Java EE Developers パッケージ）
- Drools 4.0.7

2.1. Eclipse プラグインのセットアップ

Eclipse を使用して開発する場合に便利なプラグインがあります。

以下の手順では、すでに Pleiades によって日本語化された Eclipse 3.4 の Eclipse IDE for Java EE Developers パッケージがインストールされていることを前提とします。

① Eclipse を起動し、[ヘルプ] – [ソフトウェア更新]メニューを押下します。

② [ソフトウェア更新およびアドオン]ダイアログで[サイトの追加]ボタンを押下します。

③ [ロケーション]テキストボックスに下記の URL を入力します。
http://download.jboss.org/jbosstools/updates/stable/

④ [ソフトウェア更新およびアドオン]ダイアログに追加されたロケーションからツリーをたどり、最新では、[JBoss Tools 3.0.0.GA-R200903141626-H5 (Development Release)]の以下の 3 つのプラグインをチェックします。

- JBoss Drools Core
- JBoss Drools Guvnor
- JBoss Drools Task

⑤ [インストール]ボタンを押下します。

⑥ [インストール]ダイアログの[完了]ボタンを押下します。

⑦　インストール完了後に、Eclipse を再起動し完了です。

2.2. Drools のダウンロード

http://www.jboss.org/drools/downloads.html から Drools 4.0.7 Binaries をダウンロードします。

2.3. サンプルのセットアップ

第 3 章以降の開発方法に関する説明にサンプルのソースコードを使用します。

①　http://www.jboss.org/drools/downloads.html から Drools 4.0.7 Examples をダウンロードします。

②　①で取得した drools-4.0.7-examples.zip を解凍します。

③　②で解凍されたフォルダにある[drools-examples-drl]フォルダを Eclipse のプロジェクトとしてインポートします。

④　Drools 4.0.7 Binaries で取得した drools-4.0.7-bin.zip を解凍します。

⑤　④で解凍されたフォルダおよび[lib]フォルダにあるすべての jar ファイルを③でインポートしたプロジェクトの[lib]フォルダ内にコピーします。

⑥　[lib]フォルダ内のすべての jar ファイルにビルド・パスを設定します。

3. ルール言語

ルールは、drl という拡張子を持つルールファイルにルール言語を使用して定義します。ルールファイルの構造は、**リスト 3-1** の通りです。

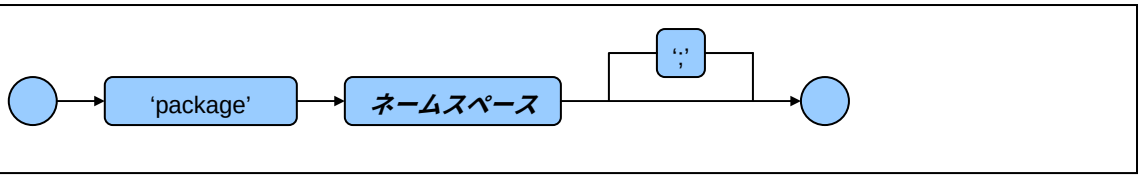
リスト 3-1：ルールファイルの構造

```
<パッケージ>
<インポート>
<エキスパンダ>
<グローバル>
<関数>
<クエリ>
<ルール>
```

3.1. パッケージ

パッケージは、ルールやその他の要素の集合でネームスペースを表します。構文は、**図 3-1** の通りです。

図 3-1：パッケージ



サンプルの HelloWorld.drl では、**リスト 3-2** のように定義しています。

リスト 3-2：HelloWorld.drl のパッケージ

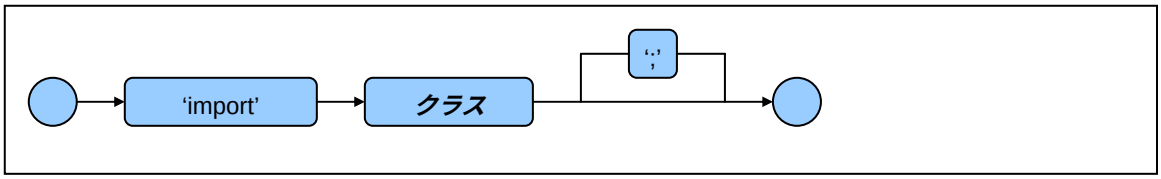
```
package org.drools.examples
```

3.2. インポート

インポートは、ルール内で使用したい Java クラスを定義します。構文は、**図**

3-2 の通りです。

図 3-2：インポート



サンプルの HelloWorld.drl では、**リスト 3-3** のように記述しています。

リスト 3-3：HelloWorld.drl のインポート

```
import org.drools.examples.HelloWorldExample.Message;
```

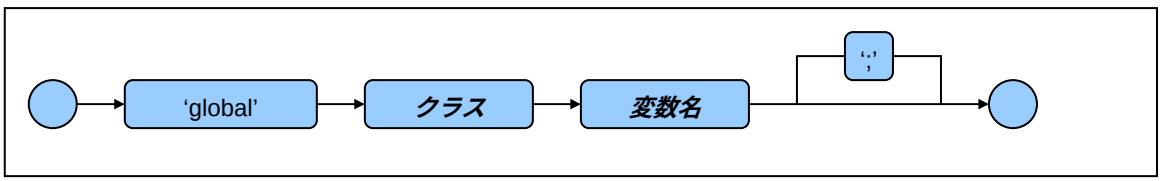
3.3. エキスパンダ

ドメイン特化言語（DSL）を利用する際に使用されます。詳細は、第 4 章で説明します。

3.4. グローバル

グローバルは、グローバル変数を表します。Java コードのオブジェクトをルールで使用できるようにするために設定します。構文は、**図 3-3** の通りです。

図 3-3：グローバル



サンプルの PetStore.drl では、**リスト 3-4** のように記述しています。

リスト 3-4：PetStore.drl のグローバル

```
global JFrame frame
```

また、グローバルを使用するには、ルールファイルを実行する Java コードにおいて使用したいオブジェクトを org.drools.WorkingMemory オブジェクトにセットする必要があります。サンプルの org.drools.examples.PetStore クラスでは、402～404 行目（**リスト 3-5**）のようにグローバルで使用するオブジェクトを org.drools.WorkingMemory クラスにセットしています。

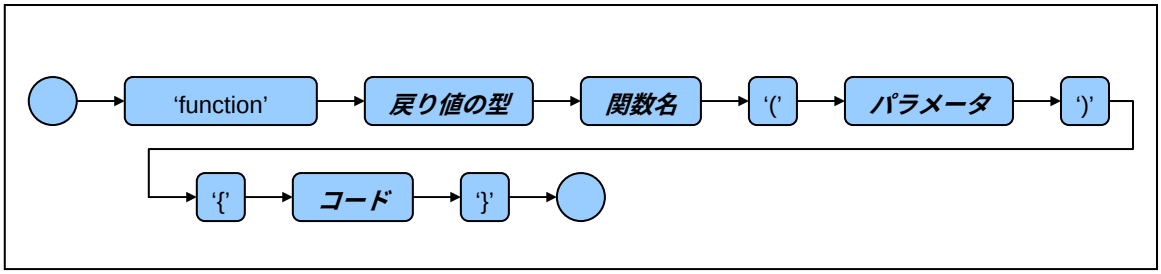
リスト 3-5：org.drools.examples.PetStore

```
WorkingMemory workingMemory = ruleBase.newStatefulSession();
workingMemory.setGlobal( "frame",
                         frame );
```

3.5. 関数

関数は、ルールで使用できる関数を定義することができます。特定のロジックを繰り返す場合に有用です。構文は、**図 3-4** の通りです。

図 3-4：関数



サンプルの PetStore.drl では、**リスト 3-6** のように記述しています。

リスト 3-6：PetStore.drl における関数

```
function void doCheckout(JFrame frame, WorkingMemory workingMemory) {
        Object[] options = {"Yes",
                            "No"};

        int n = JOptionPane.showOptionDialog(frame,
```

```
                                        "Would you like to checkout?",
                                        "",
                                        JOptionPane.YES_NO_OPTION,
                                        JOptionPane.QUESTION_MESSAGE,
                                        null,
                                        options,
                                        options[0]);

        if (n == 0) {
                workingMemory.setFocus( "checkout" );
        }
}
```

コード内は、Java で記述することができます。

3.6. ルール

ルールファイルには、複数のルールを定義することができます。ルールの構造は、**リスト 3-7** の通りです。

リスト 3-7：ルール

```
rule "<ルール名>"
    <属性>
    when
        <LHS>
    then
        <RHS>
end
```

LHS（Left Hand Side）は、条件を表します。また、RHS（Right Hand Side）は、ルールのアクションを表します。すなわち、ファクト（事実）が LHS の条件を満たすと RHS のアクションが実行されます。

ルールを実行するにはファクトを Java のオブジェクトとして表現し、Drools のワーキングメモリーに格納します。

3.6.1. 属性

属性は、ルールの動作を宣言的に設定することができます。

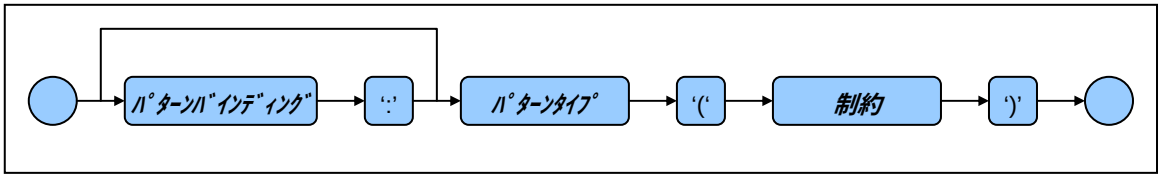
表 3-1：属性

属性名	タイプ	デフォルト値	説明
no-loop	boolean	false	ルールを実行した結果、ファクトが変更されるとルールが再実行される可能性があります。true にセットすると再実行しません。
lock-on-active	boolean	false	true にセットすると、ルールフローグループがアクティブでなくなるかアジェンダグループがフォーカスを失わない限り、指定したルールを実行できません。
salience	整数	0	ルールの優先度を現します。大きい値がより高い優先度になります。
agenda-group	文字列	MAIN	アジェンダグループ名を定義します。アジェンダグループ内のルールのみが実行されます。
auto-focus	boolean	false	true にセットされると、agenda-group がフォーカスを持たない場合は、自動的にフォーカスが得られ、ルールが実行される可能性を与えます。
activation-group	文字列	N/A	アクティベーショングループ名を定義します。アクティベーショングループ内の実行すべき最初のルールは、他のルールの実行を止めます。
dialect	文字列		LHS および RHS のコードブロック内のコードひょ間で使用する言語を指定します。”java”と”mvel”のどちらか一方を指定できます。
date-effective	文字列	N/A	現在の日時が date-effective で指定した日時の後である場合のみ実行されます。
date-expires	文字列	N/A	現在の日時が date- expires で指定した日時の後である場合は、実行されません。
duration	long		指定した期間の後に実行されます。
ruleflow-group	文字列	N/A	ルールフローグループ名を定義します。

3.6.2. LHS

0 以上の条件を設定することができます。0 の場合は、必ずルールが適用され RHS が実行されます。条件は、パターンとして記述します。パターンの構文は、**図 3-5** の通りです。

図 3-5：パターン



パターンバインディングは、「**:**」の右側で定義するパターンに当てはまるオブジェクトを格納する変数です。

パターンタイプは、パターンの対象となるオブジェクトを指定します。

制約は、パターンタイプに対する条件を記述します。**表 3-2** の連結詞を使用して複数指定することができます。優先度は、「**&& > || > ,**」です。優先度は、括弧（”()”）で囲むことで変更できます。しかし、「**,**」は、括弧内に使用することはできません。

表 3-2：連結詞

連結詞	説明
&&	AND 条件
||	OR 条件
,	AND 条件

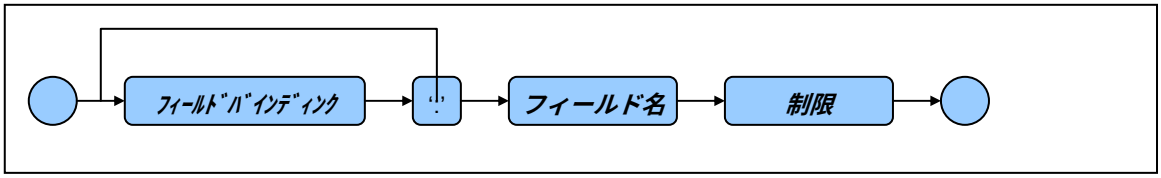
サンプルの HelloWorld.drl の[Hello World]ルールの LHS では、**リスト 3-8** のように記述しています。

リスト 3-8：HelloWorld.drl の[Hello World]ルールの LHS

```
m : Message( status == Message.HELLO, message : message )
```

サンプルの HelloWorld.drl の[Hello World]ルールの制約は、フィールドに対する条件であるフィールド制約を記述しています。フィールド制約の構文は、**図 3-6** の通りです。

図 3-6：フィールド制約



フィールドバインディングは、「**:**」の右側で定義する制限に当てはまるフィールドを格納する変数です。

サンプルの HelloWorld.drl の[Hello World]ルールの LHS（**リスト 3-8**）は、ワーキングメモリーに格納されたファクトである Message オブジェクトの status フィールドが Message.HELLO である場合、条件を満たすことを表しています。また、条件を満たした Message オブジェクトを m パターンバインディングに格納し、その Message オブジェクトの message フィールドを message フィールドバインディングに格納しています。

3.6.3. RHS

RHS は、LHS で定義したパターンに当てはまる場合に実行されます。

サンプルの HelloWorld.drl の[Hello World]ルールの RHS では、**リスト 3-9** のように記述しています。

リスト 3-9：HelloWorld.drl の[Hello World]ルールの RHS

```
System.out.println( message );
modify ( m ) { message = "Goodbyte cruel world",
               status = Message.GOODBYE };
System.out.println( message );
```

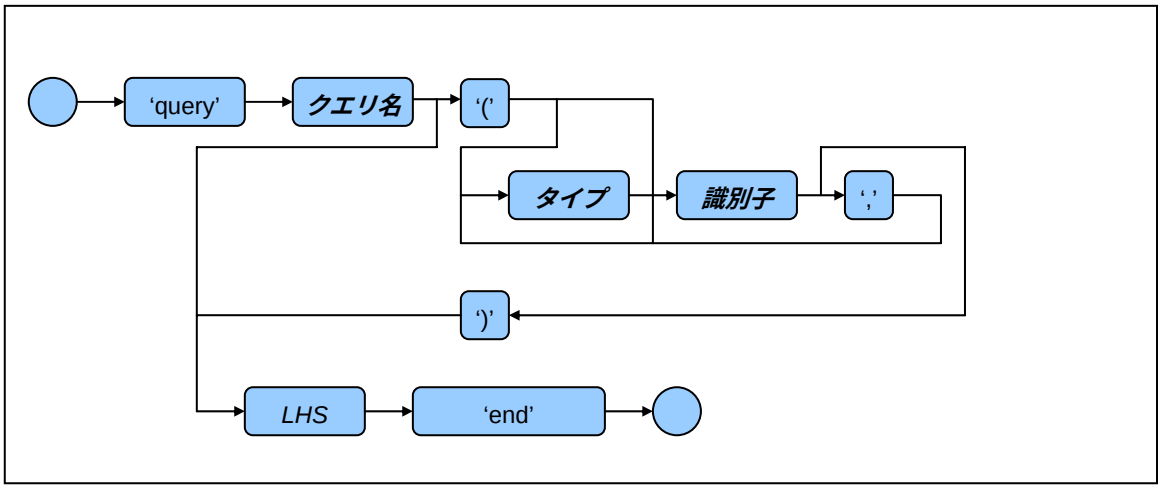
Java と似た文法で記述します。また、Java コードを実行することがで

きます。

3.7. クエリ

クエリは、ルールの LHS に当たる部分を名前付きで定義したものです。構文は、**図 3-7** の通りです。

図 3-7：クエリ



結果は、org.drools.WorkingMemory クラスの getQueryResults メソッドで取得することができます。

4. ドメイン特化言語（DSL）

ルール言語は、システム開発者には分かりやすい文法ですが、業務の専門家には分かりづらい文法になっています。DSL は、業務の専門家が分かる言葉でビジネスルールを記述することができます。

4.1. DSL ファイル

DSL は、dsl という拡張子を持つ DSL ファイルに定義します。

サンプルの ticketing.dsl をテキストエディタで表示すると**リスト 4-1** のようになります。

リスト 4-1：ticketing.dsl のテキストエディタ表示

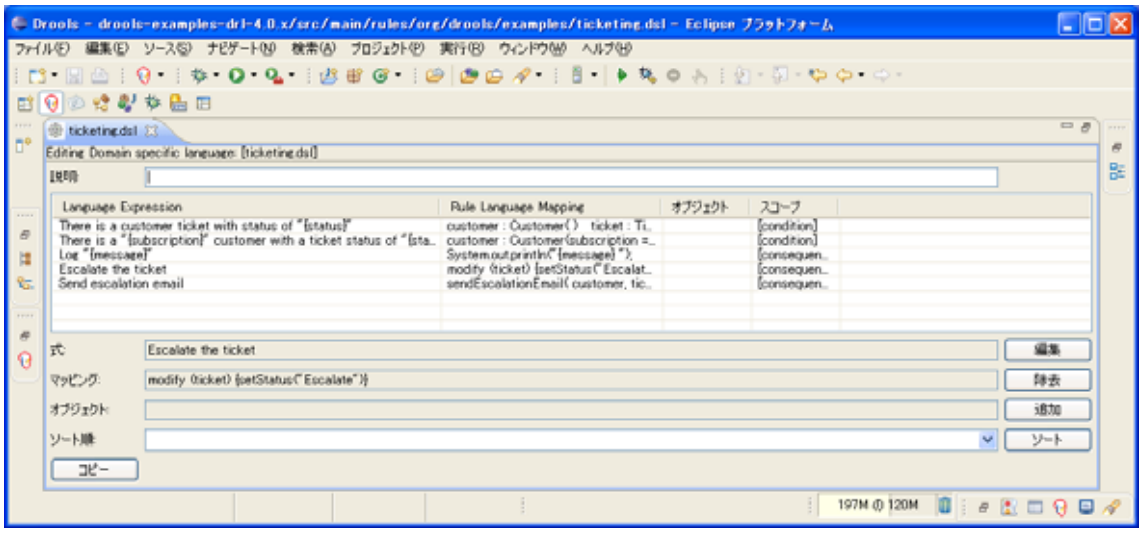
```
[condition][]There is a customer ticket with status of "{status}"=customer : _
        Customer( )   ticket : Ticket( customer == customer, status == "{status}" )
[condition][]There is a "{subscription}" customer with a ticket status of _
        "{status}"=customer : Customer(subscription == "{subscription}") _
        ticket : Ticket( customer == customer, status == "{status}")
[consequence][]Log "{message}"=System.out.println("{message} ");
[consequence][]Escalate the ticket=modify (ticket) {setStatus("Escalate")}
[consequence][]Send escalation email=sendEscalationEmail( customer, ticket );
```

[condition]の行は、LHS に記述できる DSL を定義している行です。また、**[consequence]**の行は、RHS に記述できる DSL を定義している行です。

「=」の左側は、DSL を定義しています。また、右側は、左側で定義した DSL に対応するルール言語での記述となります。

DSL ファイルは、テキストエディタだけでなく、Eclipse プラグインの DSL Editor で編集することができます（**図 4-1**）。

図 4-1：ticketing.dsl の DSL Editor 表示



4.2. ルールファイル

定義した DSL を使用するには、サンプルの TroubleTicketWithDSL.dslr（**リスト 4-2)**のように expander で使用する DSL ファイルを宣言する必要があります。

リスト 4-2：TroubleTicketWithDSL.dslr

```
package org.drools.examples
expander ticketing.dsl
...
```

4.3. DSL の使用例

DSL の使用例をサンプルの TroubleTicketWithDSL.dslr の[New Ticket]ルール（**リスト 4-3**）で確認したいと思います。

リスト 4-3：TroubleTicketWithDSL.dslr の[New Ticket]ルール

```
rule "New Ticket"
        salience 10
        when
       There is a customer ticket with status of "New"
        then
           Log "New ticket..."
```

```
end
```

LHS および RHS を見ると「There is a customer ticket with status of "New"」、「Log "New ticket..."」となっており、業務の専門家にとっても分かりやすい表現になっています。

これは、Drools によって DSL ファイルである ticketing.dsl（**リスト 4-4**）で定義された DSL をルール言語にマッピングしているためです。

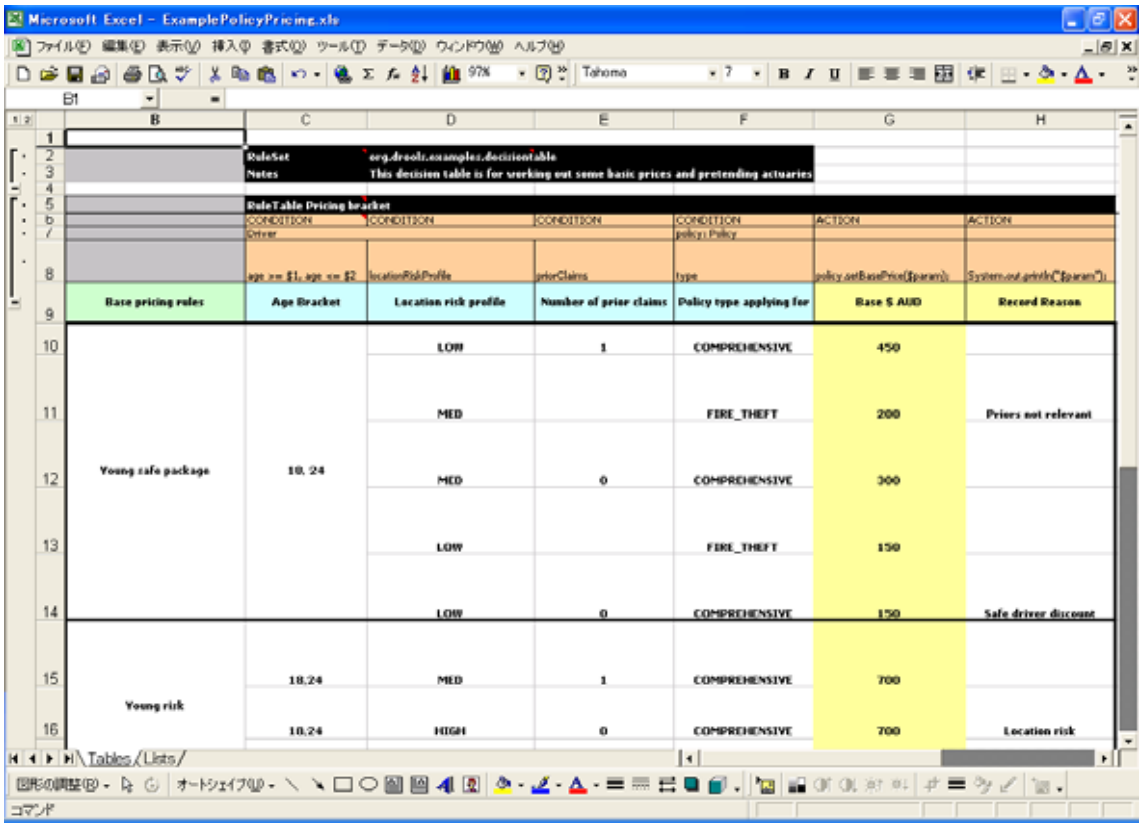
リスト 4-4：ticketing.dsl

```
[condition][]There is a customer ticket with status of "{status}"=customer : _
        Customer( )    ticket : Ticket( customer == customer, status == "{status}" )
...
[consequence][]Log "{message}"=System.out.println("{message} ");
...
```

5. 決定テーブル

Drools では、ルールファイルだけでなく、Excel のワークシートを使った決定テーブルでもルールを定義することができます。決定テーブルは、ルールの数が多い場合に強力なツールになります。**図 5-1** は、サンプルの ExamplePolicyPricing.xls です。

図 5-1：ExamplePolicyPricing.xls



5.1. キーワードとルールテーブル

決定テーブルは、Excel のセルにキーワードとルールテーブルを記述することによってルールやルールの振る舞いを定義します。

5.1.1. キーワード

キーワードは、1 つのセルに 1 つ記述します。使用できるキーワードは、**表 5-1** の通りです。

表 5-1：キーワード

キーワード	説明	備考
RuleSet	右側のセルにパッケージ名を記述します。	1 つのみ
Sequential	右側のセルに true または false を記述します。true の場合は、ルールテーブルを上から順番に実行します。	オプション
Import	右側のセルにインポートする Java クラスをカンマ区切りで記述します。	オプション
RuleTable	「RuleTable <ルール名>」という構文で記述します。ルールテーブルは、下の行から始まります。ルールテーブルは、1 つの空白行が現れるまで左から右、上から下の順に読み取られます。	少なくとも 1 つ
CONDITION	この列がルール条件を記述する列であることを表します。	ルールテーブルに対して少なくとも 1 つ
ACTION	この列がルール条件に当てはまるときに実行されるアクションを記述する列であることを表します。	ルールテーブルに対して少なくとも 1 つ
PRIORITY	この列の値をルール行に対して salience 属性として設定します。Sequential キーワードよりも優先されます。	オプション
DURATION	この列の値をルール行に対して duration 属性として設定します。	オプション
NAME	この列の値がその行から生成されたルールに対する名前を表します。	オプション
Functions	右側のセルにルールで使用できる関数を記述します。	オプション
Variables	右側のセルにルールで使用できるグローバルを記述します。	オプション
NO-LOOP	この列の値をルール行に対して no-loop 属性として設定します。	オプション
ACTIVATION-GROUP	この列の値をルール行に対して activation-group 属性として設定します。	オプション
RULEFLOW-GROUP	この列の値をルール行に対して ruleflow-group 属性として設定します。	オプション

5.1.2. ルールテーブル

ルールテーブルは、RuleTable キーワードが記述されたセルから始まり、ルールを定義するために記述します。

ルールテーブルの記述方法をサンプルの ExamplePolicyPricing.xls の Discounts ルールテーブル（**図 5-2**）を例に説明します。

図 5-2：ExamplePolicyPricing.xls の Discounts ルールテーブル

	RuleTable Discounts			
	CONDITION	CONDITION	CONDITION	ACTION
	Driver		policy: Policy	policy
	age >= $1, age <= $2	priorClaims	type	applyDiscount($param)
Promotional discount rules	**Age Bracket**	**Number of prior claims**	**Policy type applying for**	**Discount %**
Rewards for safe drivers	18,24	0	COMPREHENSIVE	1
	18,24	0	FIRE_THEFT	2
	25,30	1	COMPREHENSIVE	5
	25,30	2	COMPREHENSIVE	1
	25,30	0	COMPREHENSIVE	20

RuleTable キーワードは、これからルールテーブルを記述することを宣言し、ルール名を定義します。サンプルではルール名は「Discounts」となります。

RuleTable キーワードの 1 つ下の行は、CONDITION キーワードおよび ACTION キーワードを使用し、この列のルールがルールの LHS を定義しているのか RHS を定義しているのかを宣言します。CONDITION キーワードは LHS を表し、ACTION キーワードは RHS を表します。

RuleTable キーワードから 2 つ下の行は、パターンタイプを定義します。RuleSet キーワードで記述したパッケージにあるクラスを示しています。

パターンタイプの 1 つ下の行は、制約を定義します。

制約の 1 つ下の行は、パラメータ名を記述します。

パラメータ名の 1 つ下の行以降は、パラメータ値を定義します。

サンプルの 1 つ目の LHS の制約では、「**age >= $1, age <= $2**」と記述されていますが、「**$1**」、「**$2**」は、パラメータ値に記述された値（「**18,24**」等）をカンマ区切りにし、「1 つ目のパラメータ」、「2 つ目のパラメータ」として制約で使用することができます。パラメータ値が 1 つしかない場合は、サンプルの RHS のように「**$param**」として使用することができます。

6. まとめ

JBoss Rules（Drools）は、ビジネスルールをソースコードから外出しすることにより、変更容易性を高めることができます。また、DSL と決定テーブルという 2 つの強力なツールにより生産性を高めることができます。

DSL により業務の専門家が分かる言葉でビジネスルールを定義できるので、システム開発者と業務の専門家との認識のずれを起こりにくくします。また、業務の専門家自らがルールのパラメータを変更することも可能になります。

決定テーブルは、**Excel** という一般的なソフトウェアを使用することによって、分かりやすい形でルールを定義することができます。しかも、変更は簡単です。

JBoss Rules（Drools）は、変更が発生する可能性が高いビジネスルールが存在する場合には検討に値するプロダクトであると思います。

7. 参考資料

- 『jboss.org』
  http://www.jboss.org/drools/

- 『JBoss Rules リファレンスマニュアル』
  http://www.redhat.com/docs/ja-JP/JBoss_SOA_Platform/4.2/html/JBoss_Rules_Reference_Manual/index.html

- 『JBoss Rules 入門』 日本 JBoss ユーザグループ 山本祐介
  http://www.slideshare.net/yusukey/jboss-rules

開発部
大森　洋行