JBoss Enterprise SOA Platform 5

Drools Tools リファレンスガイド

このガイドは開発者向けです。

Last Updated: 2023-03-28

JBoss Enterprise SOA Platform 5 Drools Tools リファレンスガイド

このガイドは開発者向けです。

Enter your first name here. Enter your surname here.
Enter your organisation's name here. Enter your organisational division here.
Enter your email address here.

法律上の通知

概要

このガイドでは、開発者が JBDS の Drools Tools プラグインを使用する方法を説明します。

目次

はじめに

1. ドキュメント規則

本ガイドでは、いくつかの規則を使用して特定の単語やフレーズを強調表示し、特定の情報への注意を促しています。

1.1. 表記規則

特定の単語や句への注意を促すために 4 つの表記慣習を使用しています。これらの規則や、これらが適用される状況は以下のとおりです。

等幅ボールド

シェルコマンド、ファイル名、パスなど、システム入力を強調表示するために使用されます。キーとキーの組み合わせを強調表示するためにも使用されます。以下に例を示します。

> 現在の作業ディレクトリーのファイル **my_next_bestselling_novel** の内容を表示するには、シェルプロンプトで **cat my_next_bestselling_novel** コマンドを入力し、**Enter** を押してコマンドを実行します。

上記には、ファイル名、シェルコマンドおよびキーが含まれます。これはすべて等幅ボールドで表示され、コンテキストにより区別可能なものになります。

キーの組み合わせは、各部分をつなぐプラス記号によって、個別のキーと区別できます。以下に例を示します。

> **Enter** を押してコマンドを実行します。
>
> **Ctrl**+**Alt**+**F2** を押して、仮想ターミナルに切り替えます。

最初の例では、押す特定のキーを強調表示しています。2 つ目の例は、同時に押す 3 つのキーのセットというキーの組み合わせを強調表示しています。

ソースコードの場合、段落内で記述されるクラス名、メソッド、関数、変数名、および戻り値は、上記のように **等幅ボールド** で示されます。以下に例を示します。

> ファイル関連のクラスには、ファイルシステムの **filesystem**、ファイルの **file**、ディレクトリーの **dir** が含まれます。各クラスには、独自の関連付けられたパーミッションセットがあります。

プロポーショナルボールド

これは、アプリケーション名、ダイアログボックステキスト、ラベルが付いたボタン、チェックボックスおよびラジオボタン、メニュータイトルおよびサブメニュータイトルなど、システムで発生した単語またはフレーズを示します。以下に例を示します。

> メインメニューバーから System → Preferences → Mouse を選択し、Mouse Preferences を起動します。**Buttons** タブで、**Left-handed mouse** チェックボックスを選択し、**Close** をクリックしてメインのマウスボタンを左から右に切り替えます (マウスを左手で使い易くします)。
>
> 特殊文字を gedit ファイルに挿入するには、メインメニューバーから Applications → Accessories → Character Map を選択します。次に、Character Map メニューバーから Search → Find... を選択し、**Search** フィールドに文字の名前を入力して **Next** をクリックします。目的の文字が **Character Table** で強調表示されます。この強調表示し

た文字をダブルクリックして **Text to copy** フィールドに配置し、**Copy** ボタンをクリックします。ここでドキュメントに戻り、**gedit** メニューバーから **Edit → Paste** を選択します。

上記のテキストにはアプリケーション名、システム全体のメニュー名および項目、アプリケーション固有のメニュー名、GUI インターフェイス内のボタンおよびテキストなどがあります。すべては proportional bold で示され、コンテキストと区別できます。

等幅ボールドイタリック または ***プロポーショナルボールドイタリック***

等幅ボールドまたはプロポーショナルボールドのいずれでも、イタリックが追加されると、置換または変数テキストを意味します。イタリックは、状況に応じて変化するテキストや、文字を入力しないテキストを表します。以下に例を示します。

ssh を使用してリモートマシンに接続するには、シェルプロンプトで **ssh *username@domain.name*** を入力します。リモートマシンが **example.com** で、そのマシン上でのユーザー名が john の場合は、**ssh john@example.com** と入力します。

mount -o remount *file-system* コマンドにより、指定したファイルシステムが再マウントされます。たとえば、**/home** ファイルシステムを再マウントする場合、コマンドは **mount -o remount /home** となります。

現在インストールされているパッケージのバージョンを表示するには、**rpm -q *package*** コマンドを使用します。これにより、***package-version-release*** のような結果が返されます。

上記の太字のイタリック体の用語、username、domain.name、file-system、package、version、および release に注意してください。各単語はプレースホルダーで、コマンドの発行時に入力するテキストまたはシステムによって表示されるテキストのどちらかになります。

作業のタイトルを示す標準的な使用法のほかに、イタリックは新用語と重要な用語の最初の使用を示します。以下に例を示します。

Publican は *DocBook* 公開システムです。

1.2. 引用規則

端末の出力およびソースコードの一覧は、周りのテキストから視覚的に表示されます。

ターミナルに送信される出力は **mono-spaced roman** に設定され、以下のように表示されます。

```
books       Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

ソースコードの一覧も **mono-spaced roman** に設定されますが、以下のように構文の強調表示が追加されます。

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                 struct kvm_assigned_pci_dev *assigned_dev)
{
        int r = 0;
        struct kvm_assigned_dev_kernel *match;

        mutex_lock(&kvm->lock);

        match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
```

```
                                assigned_dev->assigned_dev_id);
        if (!match) {
                printk(KERN_INFO "%s: device hasn't been assigned before, "
                  "so cannot be deassigned\n", __func__);
                r = -EINVAL;
                goto out;
        }

        kvm_deassign_device(kvm, match);

        kvm_free_assigned_device(kvm, match);

out:
        mutex_unlock(&kvm->lock);
        return r;
}
```

1.3. 注記および警告

最後に、3 つの視覚的スタイルを使用して、見落とす可能性のある情報に注意を促します。

注記

注記とは、タスクへのヒント、ショートカット、または代替アプローチです。注意を無視しても悪い結果を招くことはありませんが、便利なヒントを見逃してしまう可能性があります。

重要

見落としやすい詳細のある重要なボックス: 現行セッションにのみ適用される設定変更や、更新を適用する前に再起動が必要なサービスなどです。Important というラベルが付いたボックスを無視しても、データが失われることはありませんが、スムーズな操作が行えないことがあります。

警告

警告は無視すべきではありません。警告を無視すると、データが失われる可能性があります。

2. ヘルプの利用とフィードバック提供

2.1. ヘルプが必要ですか？

本ガイドで説明されている手順で問題が発生した場合は、Red Hat カスタマーポータル http://access.redhat.com にアクセスしてください。カスタマーポータルでは、以下を行うことができます。

- Red Hat 製品に関する技術サポート記事の知識ベースの検索または閲覧。

- Red Hat グローバルサポートサービス (GSS) へのサポートケースの送信。
- その他の製品ドキュメントへのアクセス。

Red Hat は、Red Hat のソフトウェアおよびテクノロジーについて、多くの電子メーリングリストも提供しています。一般に公開されているメーリングリストの一覧は、https://www.redhat.com/mailman/listinfoを参照してください。メーリングリストの名前をクリックして、その一覧をサブスクライブするか、またはメーリングリストのアーカイブにアクセスします。

2.2. ご意見をお寄せください

本ガイドで誤字脱字を発見されたり、このマニュアルを改善するための提案をお持ちの場合は、弊社までご連絡ください。JBoss Enterprise SOA Platform の製品に対して、Bugzilla: http://bugzilla.redhat.com/ レポートを送信してください。

バグレポートを送信する際には、マニュアル識別子である 『Drools_Tools_Reference_Guide』 を指定してください。

本ガイドを改善するためのご意見やご提案をお寄せいただく場合は、できるだけ具体的にご説明ください。エラーが見つかった場合は、簡単に確認できるように、セクション番号と前後のテキストを含めてください。

第1章 はじめに

1.1. ビジネス統合

動的かつアジャイルなビジネスインフラストラクチャーを提供するためには、異なるアプリケーションとデータソースが最小限のオーバーヘッドで相互に通信できるように、サービス指向のアーキテクチャーを用意することが重要です。

JBoss Enterprise SOA Platform は、ビジネスプロセスの変更に対応するために、継続的に再利用することなく、ビジネスサービスをオーケストレーションできるフレームワークです。JBoss Enterprise SOA Platform では、ビジネスルールとメッセージの変換およびルーティング機能を使用することで、複数のソースからビジネスデータを操作できます。

バグの報告

1.2. サービス指向アーキテクチャーとは

はじめに

SOA (*Service Oriented Architecture*) は単一のプログラムまたはテクノロジーではありません。ソフトウェア設計パラダイムと見なします。

すでに分かっているように、*ハードウェアバス* は、複数のシステムとサブシステムを結び付ける物理コネクターです。これを使用する場合は、システムのペア間で多数のポイントツーポイントコネクターを使用する代わりに、各システムを中央バスに接続するだけです。*エンタープライズサービスバス* (ESB) は、ソフトウェアとまったく同じことを行います。

アーキテクトは、メッセージングシステムの上記のアーキテクチャー層にあります。このメッセージングシステムは、このメッセージングシステムを使用することでサービス間の *非同期通信* を容易にします。実際、アーキテクトを使用している場合、すべてを概念的に、*サービス*（このコンテキストではアプリケーションソフトウェア）またはサービス間で送信される *メッセージ* のいずれかです。サービスは接続アドレスとして一覧表示されます (*エンドポイント参照* と呼ばれています)。

このコンテキストでは、サービスは必ずしも Web サービスであるとは限らないことに注意することが重要です。File Transfer Protocol や Java Message Service などのトランスポートを使用する他のタイプのアプリケーションもサービスになる可能性があります。

注記

この時点で、エンタープライズサービスバスがサービス指向のアーキテクチャーと同じ場合は、妨げられる場合があります。回答は Not exactly です。 アーキテクトは、サービス指向のアーキテクチャーを形成しません。代わりに、ツールを多数使用して構築できます。特に、SOA が必要とする *loose-coupling* および *非同期メッセージを渡すこと* が容易になります。SOA は常にソフトウェア以外のものと考えてください。これは一連の原則、パターン、およびベストプラクティスです。

バグの報告

1.3. サービス指向アーキテクチャーの重要なポイント

以下は、サービス指向のアーキテクチャーの主要なコンポーネントです。

1. 交換される *メッセージ*
2. サービスリクエスターおよびプロバイダーとして動作する *エージェント*
3. メッセージを送受信できるようにする *共有トランスポートメカニズム*。

バグの報告

1.4. JBOSS ENTERPRISE SOA PLATFORM とは

JBoss Enterprise SOA Platform は、エンタープライズアプリケーションインテグレーション (EAI) およびサービス指向アーキテクチャー (SOA) ソリューションを開発するためのフレームワークです。これは、エンタープライズサービスバス (JBoss Warehouse) およびビジネスプロセス自動化インフラストラクチャーで設定されます。これにより、ビジネスサービスの構築、デプロイ、統合、オーケストレーションを行うことができます。

バグの報告

1.5. SERVICE-ORIENTED ARCHITECTURE PARADIGM

SOA (Service-ient Architecture)は、リクエスター、プロバイダー、およびブローカーの 3 つのロールで設定されます。

サービスプロバイダー

サービスプロバイダーはサービスへのアクセスを許可し、サービスの説明を作成し、サービスブローカーに公開します。

サービスリクエスター

サービスリクエスターは、サービスブローカーが提供するサービスの説明を検索して、サービスを検出します。リクエスターはサービスプロバイダーが提供するサービスにバインドするロールも果たします。

サービスブローカー

サービスブローカーは、サービスの説明のレジストリーをホストします。リクエスターをサービスプロバイダーにリンクします。

バグの報告

1.6. コアおよびコンポーネント

JBoss Enterprise SOA Platform は、データ統合のニーズに対応する包括的なサーバーを提供します。基本的なレベルでは、Enterprise Service Bus を介してビジネスルールを更新し、メッセージをルーティングすることができます。

JBoss Enterprise SOA Platform の中心となるのは、Enterprise Service Bus です。JBoss (ESB) はメッセージの送受信を行う環境を作成します。メッセージにアクションを適用して変換し、サービス間でルーティングすることができます。

JBoss Enterprise SOA Platform を設定するコンポーネントは複数あります。ESB とともに、レジストリ (jUDDI)、変換エンジン (Smooks)、メッセージキュー (HornetQ)、BPEL エンジン (Riftsaw) が用意されています。

バグの報告

1.7. JBOSS ENTERPRISE SOA PLATFORM のコンポーネント

- 完全な Java EE 準拠のアプリケーションサーバー (JBoss Enterprise Application Platform)
- Enterprise Service Bus (JBoss ESB)
- ビジネスプロセス管理システム (jBPM)
- ビジネスルールエンジン (JBoss ルール)
- オプションの JBoss Enterprise Data Services (EDS) 製品のサポート。

バグの報告

1.8. JBOSS ENTERPRISE SOA PLATFORM の機能

JBoss Enterprise Service Bus (ESB)

ドメインはサービス間でメッセージを送信し、それらを変換して、異なるタイプのシステムで処理できるようにします。

Business Process Execution Language (BPEL)

Web サービスを使用して、この言語を使用してビジネスルールをオーケストレーションできます。これは、ビジネスプロセス命令を簡単に実行するために SOA に含まれています。

Java Universal Description、Discovery and Integration (jUDDI)

これは SOA のデフォルトサービスレジストリーです。ここで説明する内容は、インストラクター上のサービスに関する情報をすべて保管する場所です。

Smooks

この変換エンジンは SOA と組み合わせて使用してメッセージを処理できます。また、メッセージを分割して正しい宛先に送信するためにも使用できます。

JBoss ルール

これは、SOA にパッケージ化されたルールエンジンです。受信するメッセージからデータを推測して、実行する必要があるアクションを判別できます。

バグの報告

1.9. JBOSS ENTERPRISE SOA PLATFORM の JBOSSESB コンポーネントの機能

JBoss Enterprise SOA Platform の JBossESB コンポーネントは以下をサポートします。

- 複数のトランスポートおよびプロトコル
- リスナーアクションモデル (これにより、サービスを相互に選択可能)
- コンテンツベースのルーティング (JBoss Rules エンジン、XPath、Regex、および Smooks 経由)
- サービスオーケストレーション機能を提供するために JBoss Business Process Manager (jBPM) との統合
- ビジネスルールの開発機能を提供するために、JBoss ルールとの統合
- BPEL エンジンとの統合。

さらに、新しいデプロイメントにレガシーシステムを統合し、同期または非同期で通信できるようにします。

さらに、エンタープライズサービスバスは、以下を可能にするインフラストラクチャーおよびツールセットを提供します。

- さまざまなトランスポートメカニズム (電子メールや JMS など) と連携するよう設定されます。
- 汎用オブジェクトリポジトリーとして使用します。
- プラグ可能なデータ変換メカニズムを実装できるようにします。
- 対話のロギングをサポートします。

重要

ソースコードには、**org.jboss.internal.soa.esb** と **org.jboss.soa.esb** の 2 つのツリーがあります。**org.jboss.internal.soa.esb** パッケージの内容をそのまま使用します。これは、通知なしに変更される可能性があるためです。これとは対照的に、**org.jboss.soa.esb** パッケージ内のすべての内容は、Red Hat の非推奨ポリシーでカバーされます。

バグの報告

1.10. タスク管理

JBoss SOA は、影響を受けるすべてのシステムで汎用的に実行するタスクを指定することにより、タスクを簡素化します。つまり、ユーザーは各ターミナルで個別に実行するようにタスクを設定する必要はありません。ユーザーは、Web サービスを使用してシステムを簡単に接続できます。

JBoss SOA を使用して各マシンに複数回ではなく、ネットワーク全体でトランザクションを 1 回委譲することで、時間を節約できます。これにより、エラーが発生する可能性も低くなります。

バグの報告

1.11. 統合のユースケース

ACME Equity は、大規模な経理サービスです。会社には多くのデータベースやシステムがあります。旧式の COBOL ベースのレガシーシステムや、近年で小規模な企業で取得されるデータベースもあります。ビジネスルールが頻繁に変化するため、これらのデータベースを統合することは困難でコストがかかります。会社は、クライアント向け e-commerce の Web サイトを新たに開発することを希望していますが、現在使用している既存のシステムとは同期しない場合があります。

会社は、安価なソリューションを希望していますが、企業セクターの厳密な規制およびセキュリティー要件に準拠するソリューションを検討しています。会社が望ましくないことは、レガシーデータベースおよびシステムを接続するために glob コードを作成および維持する必要があることです。

JBoss Enterprise SOA Platform は、これらのレガシーシステムを新しいお客様の Web サイトに統合するためにミドルウェアレイヤーとして選択されました。フロントエンドシステムとバックエンドシステムの間のブリッジを提供します。JBoss Enterprise SOA Platform で実装されたビジネスルールは、すぐに簡単に更新できます。

その結果、SOA の統合方法により、古いシステムは新しいシステムと同期できるようになりました。1 カ月あたり数万のトランザクションであっても、ボトルネックはありません。XML、JMS、FTP などのさまざまな統合タイプは、システム間でデータを移動するために使用されます。数多くのエンタープライズ標準のメッセージングシステムの 1 つを JBoss Enterprise SOA Platform にプラグインすることで、柔軟性を高めることができます。

さらに利点は、既存のインフラストラクチャーにより多くのサーバーやデータベースが追加されると、システムを簡単にスケールアップできることです。

バグの報告

1.12. ビジネス環境での JBOSS ENTERPRISE SOA PLATFORM の使用

エラーメッセージが発生する可能性が低いサービスの実装により、コストを削減できます。生産性とソーシングオプションの強化により、継続的なコストを削減できます。

情報およびビジネスプロセスは、接続が増加するため、迅速に共有できます。これは Web サービスによって強化され、クライアントを簡単に接続するために使用できます。

レガシーシステムは Web サービスと組み合わせて使用して、異なるシステムが同じ言語にピークにすることができます。これにより、システムの同期に必要なアップグレードおよびカスタムコードの量が減ります。

バグの報告

第2章 はじめに

2.1. 本ガイドの対象者

本書は、セットアップで Drools Rules を利用する方法を学びたいシステム管理者を対象としています。Drools Tools を使用して、Drools のプロセスとルールを作成、実行、およびデバッグする方法について説明します。

バグの報告

2.2. ガイドの目的

このガイドは、Drools Rules パッケージを使用して基本的なタスクと複雑なタスクの両方を実行する方法の概要をユーザーに提供することを目的としています。Drools プロジェクトの作成方法とルールのデバッグ方法について詳しく説明します。エディター、その使用方法、実行するタスクに関する情報もあります。

バグの報告

第3章 新しい DROOLS プロジェクトの作成

3.1. サンプル DROOLS プロジェクトの作成

1. 新しい Drools プロジェクトを作成するには、**File → New → Project → Drools → Drools Project** をクリックします。これにより、**New Drools Project wizard** が開きます。

   関連するフィールドにプロジェクト名を入力し、**Next** ボタンをクリックします。

2. サンプルルール、デシジョンテーブル、ルールフロー、およびそれらの Java クラスなど、デフォルトのアーティファクトのリストが表示されます。プロジェクトに関連するものを追加し、**Next** をクリックします。

3. Drools ランタイムを指定するように求められます。まだ設定していない場合は、**Configure Workspace Settings** リンクをクリックして、今すぐ設定してください。

   Drools ランタイムのワークスペース設定を行うことができる Preferences window が表示されるはずです。新しいランタイムを作成するには、**Add** ボタンをクリックします。新しいランタイムの名前と、ファイルシステム上の Drools ランタイムへのパスを入力するように求められます。

   注記

   Drools ランタイムは、ファイルシステム上の .jar のコレクションであり、Drools プロジェクトの .jar の特定の 1 つのリリースを表します。新しいランタイムを作成するときは、選択したリリースを指定するか、Drools Eclipse プラグインに含まれる .jar からファイルシステムに新しいランタイムを作成する必要があります。

4. Drools Eclipse プラグインに埋め込まれた .jar から新しい Drools 5 ランタイムを作成できます。**Create a new Drools 5 runtime** ボタンをクリックし、ランタイムを作成するフォルダーを選択して **OK** をクリックします。

5. 新規作成されたランタイムが Drools ランタイムの一覧に表示されます。これを確認して、**OK** ボタンをクリックします。

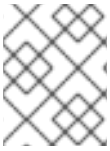
6. **Finish** をクリックしてプロジェクトの作成を完了します。

7. これで、開始するための基本的な構造、クラスパス、サンプルルール、およびテストケースが用意されました。

バグの報告

3.2. DROOLS プロジェクト構造の概要

新規作成されたプロジェクトには、**src/main/rules** ディレクトリーのサンプルルールファイル **Sample.drl** と、サンプル java ファイル **DroolsTest.java** が含まれています。これらは、**com.sample** パッケージ内の **src/main/java** フォルダーの Drools エンジンでルールを実行するために使用できます。実行中に必要な他のすべての .jar も、**Drools Library** と呼ばれるカスタムクラスパスコンテナー内のクラスパスに追加されます。



ヒント:

ルールを Java プロジェクトに保持する必要はまったくありません。これは、すでに Eclipse を Java IDE として使用している人々にとって便利なだけです。

バグの報告

3.3. 新しいルールの作成

1. 新しいプロジェクトを作成するには、空のテキスト **.drl** ファイルを作成するか、**新しいルールパッケージ** ウィザードを使用します。

2. ウィザードを開くには、**File → New → Rule Resource** を選択するか、ツールバーの JBoss Drools アイコンのメニューを使用します。

3. ウィザードページで、ルールを保存する最上位ディレクトリーとして **rules** を選択し、ルール名を入力します。次に、必須のルールパッケージ名を指定します。これにより、ルールをグループ化する名前空間が定義されます。

4. ウィザードは、開始するためのルールスケルトンを生成します。

バグの報告

第4章 デバッグルール

4.1. ブレークポイント

ルールの実行がブレークポイントに到達するたびに、実行が停止します。実行が停止したら、その時点で既知の変数を検査し、デフォルトのデバッグアクションのいずれかを使用して、次に何が起こるか (ステップオーバー、続行など) を決定できます。Debug ビューを使用して、ワーキングメモリーとアジェンダの内容を調べることができます。

バグの報告

4.2. ブレークポイントの作成

1. Drools パースペクティブの Package Explorer ビューまたは Navigator ビューでブレークポイントを作成するには、選択した **.drl** ファイルをダブルクリックしてエディターで開きます。

2. **.drl** ファイルでルールブレークポイントを追加および削除するには、次の 2 つの方法があります。

   - ブレークポイントを追加する行で **Rule editor** のルールをダブルクリックします。ルールをもう一度ダブルクリックすると、ブレークポイントを削除できます。

     注記

     ルールブレークポイントは、ルールの結果でのみ作成できることに注意してください。ブレークポイントが許可されていない行をダブルクリックしても何も起こりません。

   - ルーラーを右クリックします。コンテキストメニューで **Toggle Breakpoint** アクションを選択します。このアクションを選択すると、選択した行にブレークポイントが追加されるか、すでに存在する場合は削除されます。

3. **Debug perspective** には、定義されたすべてのブレークポイントの表示、プロパティーの取得、有効化/無効化、および削除に使用できる **Breakpoints view** が含まれています。**Window → Perspective → Others → Debug** をクリックして切り替えることができます。

バグの報告

4.3. デバッグ

1. Drools ブレークポイントは、アプリケーションを Drools アプリケーションとしてデバッグする場合にのみ有効になります。これを行うには、次の 2 つのアクションのいずれかを実行する必要があります。

   - アプリケーションのメインクラスを選択します。それを右クリックし、**Debug As → Drools Application** を選択します。

   - または、**Debug As → Debug Configuration** を選択して、デバッグ設定を作成、管理、および実行するための新しいダイアログウィンドウを開きます。

     左側のツリーで **Drools Application** 項目を選択し、**New launch configuration** ボタン (ツ

リーの上のツールバーの一番左のアイコン) をクリックします。これにより、最初に選択したメインクラスに基づいて、複数のプロパティーがすでに入力されている新しい設定が作成されます。ここに示されているすべてのプロパティーは、標準の Java プログラムと同じです。

注記

デバッグ設定の名前を意味のあるものに変更してください。

2. 下部の **Debug** ボタンをクリックして、アプリケーションのデバッグを開始します。

3. デバッグを有効にすると、アプリケーションは実行を開始し、ブレークポイントが発生すると停止します。これは、Drools ルールブレークポイントまたはその他の標準の Java ブレークポイントでもかまいません。Drools ルールブレークポイントが発生するたびに、対応する **.drl** ファイルが開き、アクティブな行が強調表示されます。**Variables** ビューには、すべてのルールパラメーターとその値も含まれます。その後、デフォルトの Java デバッグアクションを使用して、次に何をすべきか (再開、終了、ステップオーバーなど) を決定できます。デバッグビューは、その時点での作業メモリーとアジェンダの内容を決定するためにも使用できます (現在実行中の作業メモリーが自動的に表示されます)。

バグの報告

第5章 エディター

5.1. DSL

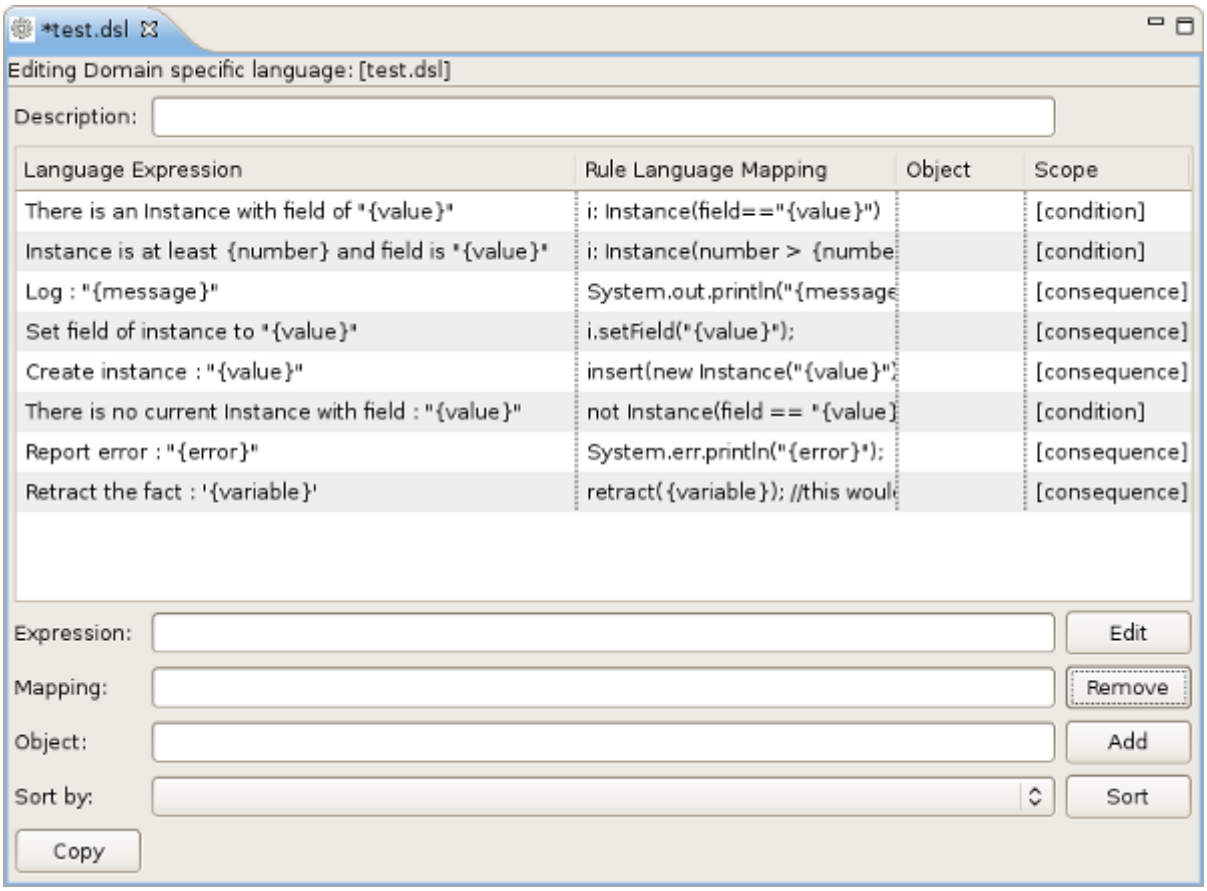
ドメイン固有言語 (DSL) は、特定のドメイン内の問題を解決するために特別に作成された一連のカスタムルールであり、ドメイン外の問題を解決できるようには意図されていません。DSL の設定はプレーンテキストで保存されます。

バグの報告

5.2. DSL エディター

- Drools では、DSL 設定は **.dsl** ファイルに保存されます。これらを作成するには、プロジェクトコンテキストメニューから **File → New → Other → Drools → Domain Specific Language** を選択します。

図5.1 DSL エディター



バグの報告

5.3. DSL エディターコンポーネント

表5.1 DSL エディターコンポーネント

コンポーネント	Description
Description	特定の言語メッセージマッピングに関するユーザーのコメント。
言語メッセージマッピングのテーブル	このテーブルは、以下の 4 つの行に分割されています。 ● **Language Expression**: ルールとして使用する式。 ● **Rule Language Mapping**: ルールの実装この言語式のルールは、ルールエンジンコンパイラーによってコンパイルされます。 ● **Object**: オブジェクトの名前 ● **Scope**: ルールの条件部分、結果部分などの場合、式が対象となる場所を示します。 行のヘッダーをクリックすると、クリックした行に応じてテーブルの行をソートできます。行をダブルクリックすると、言語マッピングの編集ウィザードが開きます。
式	選択したテーブル行 (言語メッセージマッピング) の言語式を表示します。
マッピング	選択したテーブル行 (言語メッセージマッピング) の言語マッピングのルールを表示します。
オブジェクト	選択したテーブル行 (言語メッセージマッピング) のオブジェクトを表示します。
Sort By	このオプションを使用すると、言語メッセージマッピングのソート順序を変更できます。これを行うには、ドロップダウンリストからソート方法を選択し、**Sort** ボタンをクリックします。

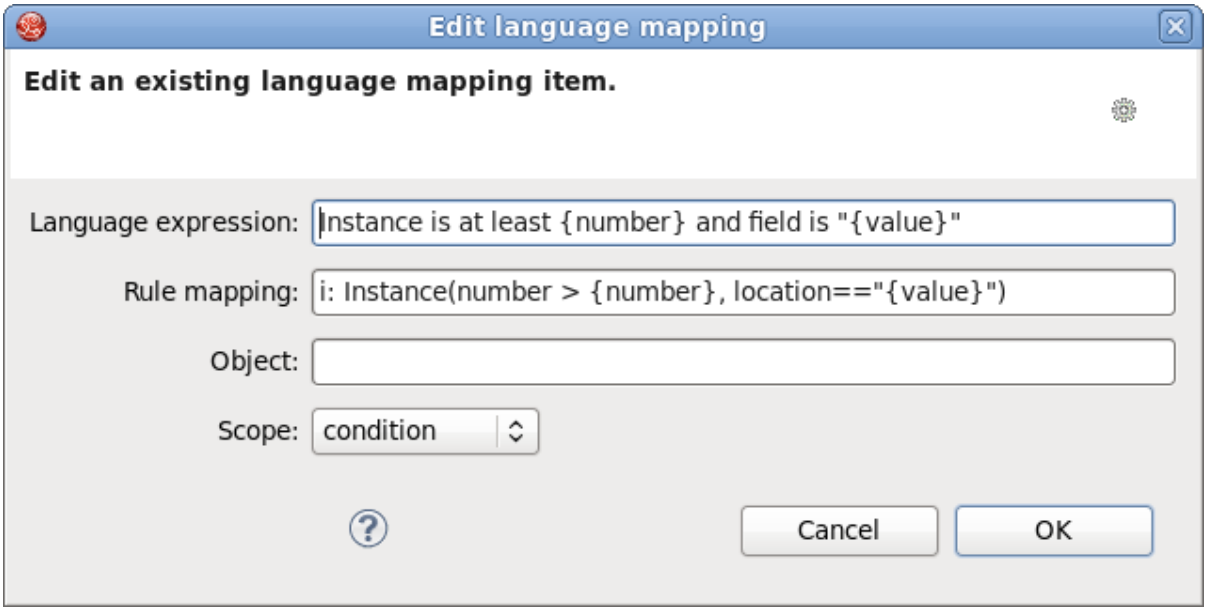
コンポーネント	Description
Buttons	• **Edit**: このボタンをクリックすると、言語メッセージマッピングテーブルで選択した行を編集できます。 • **Remove**: このボタンをクリックすると、選択したマッピング行が削除されます。 • **Add**: このボタンを使用すると、新しいマッピング行をテーブルに追加できます。 • **Sort**: 詳細については、上の **Sort By** の行を参照してください。 • **Copy**: このボタンを使用すると、選択したマッピング行からすべての情報がコピーされる新しいマッピング行をテーブルに追加できます。

バグの報告

5.4. 言語マッピングウィザードの編集

1. このウィザードを開くには、言語メッセージマッピングのテーブルの行をダブルクリックするか、**Edit** ボタンをクリックします。インスタンスの言語式、ルールマッピング、オブジェクト、およびスコープを編集できます。

図5.2 編集オプション



2. マッピングを変更するには、ユーザーは適切なオプションを編集し、最後に **OK** ボタンをクリックする必要があります。

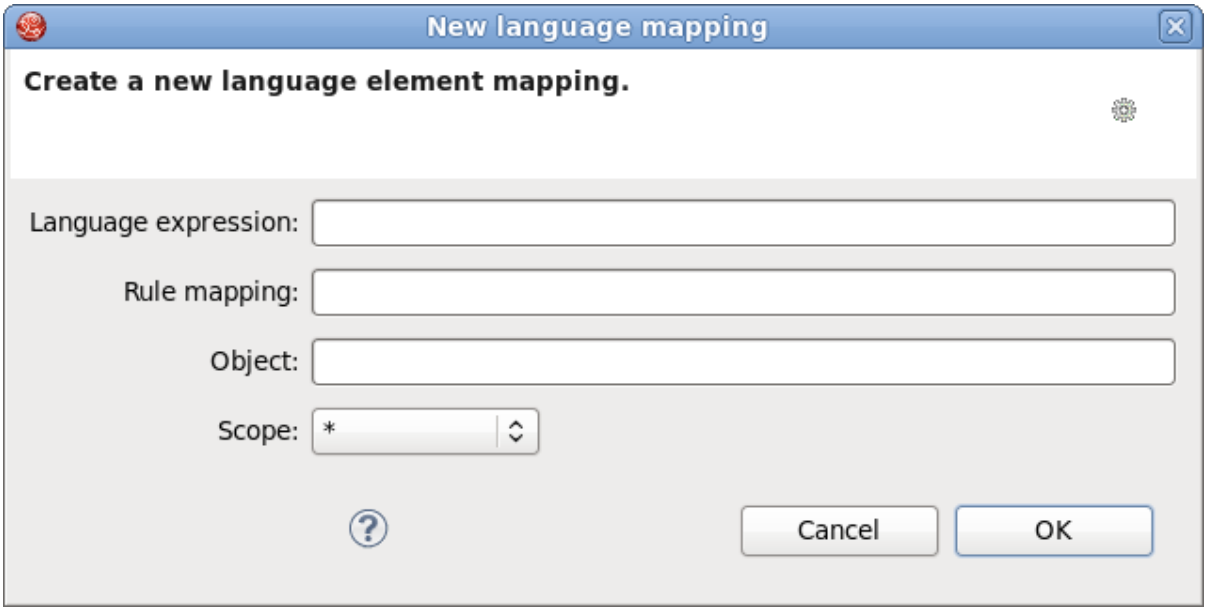
バグの報告

5.5. 言語マッピングウィザードの追加

手順5.1 タスク

1. **Add** ボタンをクリックすると、ダイアログボックスが表示されます。
2. 言語マッピングウィザードの詳細を適切なフィールドに入力します。

図5.3 ダイアログボックス



バグの報告

5.6. DROOLS FLOW EDITOR

1. ルールフローは、Eclipse 用の Drools プラグインの一部であるグラフィカルなフローエディターを使用してのみ設定できます。Drools プロジェクトを設定したら、ルールフローの追加を開始できます。プロジェクトをクリックし、**File → New → Other... → Flow File** を選択して、ルールフローファイル (.rf) を追加します。

   デフォルトでは、これらのルールフローファイル (.rf) は、グラフィカルなフローエディターで開かれます。

2. Flow エディターは **palette**、**canvas**、および **outline** ビューで構成されます。パレットで作成する要素を選択し、希望の場所をクリックしてキャンバスに追加します。

3. パレットの Select オプションをクリックしてから、ルールフローの要素をクリックすると、**Properties** ビューでその要素のプロパティーを表示および設定できます。

4. **Outline** ビューは、一度にすべてのノードが表示されない複雑なスキーマに役立ちます。**Outline** ビューを使用すると、スキーマのパーツ間を簡単に移動できます。

バグの報告

5.7. フローパレットのさまざまなタイプのコントロール要素

表5.2 フローパレットのさまざまなタイプのコントロール要素

アイコン	名前	Description
 [D]	Select	キャンバスでノードを選択します。
 [D]	マーキー	要素のグループを選択します。
 [D]	シーケンスフロー	キャンバスで 2 つの要素を結合します。

バグの報告

5.8. フローパレットのさまざまなタイプのノード

表5.3 フローパレットのさまざまなタイプのノード

アイコン	名前	Description
 [D]	開始イベント	ルールフローの開始。開始ノードは1つだけ指定できます。開始イベントは着信接続を持つことができず、1つの発信接続を持つ必要があります。ルールフロープロセスが開始されるたびに、ここで実行が開始され、この開始イベントにリンクされた最初のノードに自動的に転送されます。
 [D]	終了イベント	ルールフローファイルには、複数の終了イベントを含めることができます。このノードには、1つの着信接続があり、発信接続はありません。ルールフローで終了ノードに到達すると、ルールフローは終了します (並列処理が使用されている場合は、他の残りのアクティブなノードを含みます)。

アイコン	名前	Description
[D]	ルールタスク	一連のルールを表します。ルールタスクノードには、1つの着信接続と1つの発信接続が必要です。RuleFlowGroup プロパティーは、一連のルールを表すルールフローグループの名前を指定するために使用されます。ルールフローでルールタスクノードに到達すると、エンジンは、対応するルールフローグループの一部であるルールの実行を開始します。このルールフローグループにアクティブなルールがなくなると、実行は自動的に次のノードに進みます。
[D]	Gateway[diverge]	ルールフローにブランチを作成できます。Gateway[diverge] ノードには、1つの着信接続と 2 つ以上の発信接続が必要です。
[D]	Gateway[converge]	複数のブランチを同期できます。Gateway[converge] ノードには、2 つ以上の着信接続と1つの発信接続が必要です。
[D]	再利用可能なサブプロセス	このルールフローからの別のルールフローの呼び出しを表します。サブフローノードには、1つの着信接続と1つの発信接続が必要です。実行する必要があるプロセスの ID を指定するプロパティー processId が含まれています。ルールフローで再利用可能なサブプロセスノードに到達すると、エンジンは指定された ID でプロセスを開始します。サブフローノードは、他のサブフロープロセスがその実行を終了した場合にのみ続行されます。サブフロープロセスは独立したプロセスとして開始されます。つまり、このプロセスが終了ノードに到達しても、サブフロープロセスは終了しません。

アイコン	名前	Description
[D]	スクリプトタスク	このルールフローで実行するアクションを表します。スクリプトタスクノードには、1つの着信接続と1つの発信接続が必要です。実行する必要があるアクションを指定するプロパティー action が含まれています。ルールフローでスクリプトタスクノードに到達すると、アクションが実行され、次のノードに進みます。アクションは、(有効な) MVEL コードの一部として指定する必要があります。

バグの報告

5.9. RULE EDITOR

Rule editor は、拡張子が **.drl** (または、複数のルールファイルにわたってルールを分散する場合は、**.rule**) のファイルに対して機能します。エディターは、Eclipse の通常のテキストエディターのパターンに従い、テキストエディターのすべての通常の機能を備えています。

バグの報告

5.10. CONTENT ASSIST

Rule editor で作業しているときに、**Ctrl+Space** を押すと、通常の方法でコンテンツのサポートを得ることができます。Content Assist は、現在のカーソル位置に使用できるキーワードをすべて表示します。また、メッセージで使用可能なすべてのフィールドを提案します。

バグの報告

5.11. コードの折りたたみ

コードの折りたたみは、**Rule editor** で使用されます。エディターの左側の垂直線にあるマイナスとプラスのアイコンを使用して、ファイルのセクションを非表示および表示できます。

バグの報告

5.12. OUTLINE VIEW との同期

Rule editor は、ルールの構造を表示する **Outline view** と同期して動作し、オブジェクトをファイルにインポートし、ファイル機能を実行します。ビューは保存時に更新されます。数百のルールが存在するようなファイルで名前によるルールのナビゲーションを迅速に行うことができます。デフォルトでは、アイテムはアルファベット順にソートされます。

バグの報告

5.13. RETE TREE ビュー

Rete Tree ビューには、**.drl** ファイルの現在の **Rete Network** が表示されます。**Rule editor** の下部にある Rete Tree タブをクリックするだけです。その後、現在の Rete Network 可視化を生成できます。ノードをプッシュおよびプルして、最適なネットワーク概要を調整できます。

多数のノードがある場合は、フレームでそのいくつかを選択して、まとめることができます。現在のビューにすべてのノードが表示されていない場合は、Rete Tree をズームインおよびズームアウトできます。これには、ツールバーのコンボボックスまたは + および - アイコンを使用します。

注記

Rete Tree ビューは、Drools Builder がプロジェクトプロパティーに設定されている Drools Rule Projects でのみ機能します。

バグの報告

付録A 更新履歴

改訂 5.3.1-68.400 2013-10-31 **Rüdiger Landmann**
Rebuild with publican 4.0.0

改訂 5.3.1-68 Wed Feb 20 2013 **CS Builder Robot**
コンテンツ仕様からビルド: 7094、リビジョン: 375347