



Migration Toolkit for Applications 5.0

ルール開発ガイド

移行範囲を強化するためのカスタムルールを作成

Migration Toolkit for Applications 5.0 ルール開発ガイド

移行範囲を強化するためのカスタムルールを作成

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

Copyright © 2022 | You need to change the HOLDER entity in the en-US/Rules_Development_Guide.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux[®] is the registered trademark of Linus Torvalds in the United States and other countries.

Java[®] is a registered trademark of Oracle and/or its affiliates.

XFS[®] is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL[®] is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js[®] is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack[®] Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

概要

本ガイドでは、Migration Toolkit for Applications にカスタム XML ルールを作成する方法を説明します。

目次

多様性を受け入れるオープンソースの強化	5
第1章 はじめに	6
1.1. ルール開発ガイドについて	6
1.1.1. 本ガイドの <MTA_HOME> の使用	6
1.2. MTA ルールについて	6
第2章 ルールを使い始める	7
2.1. 最初の XML ルールの作成	7
ルールのディレクトリ構造の作成	7
ルールをテストするためのデータの作成	7
ルールの作成	7
ルールのインストール	10
ルールのテスト	10
レポートの確認	10
2.2. MIGRATION TOOLKIT FOR APPLICATIONS クイックスタートの確認	12
最新のクイックスタートのダウンロード	12
クイックスタートの GitHub プロジェクトのフォークおよびクローン作成	12
第3章 XML ルールの作成	14
3.1. XML ルール構造	14
3.1.1. ルールセット	14
3.1.2. 事前定義されたルール	14
3.2. 基本的な XML ルールの作成	15
3.2.1. 基本的な XML ルールテンプレートの作成	15
3.2.2. ruleset メタデータの作成	16
3.2.3. ルールの作成	17
3.2.3.1. <when> 条件の作成	17
3.2.3.2. <perform> アクションの作成	18
3.3. XML ルール構文	18
3.3.1. <when> 構文	18
3.3.1.1. <javaclass> syntax	19
3.3.1.1.1. 概要	19
3.3.1.1.2. <javaclass> 要素の作成	19
3.3.1.2. <xmlfile> 構文	21
3.3.1.2.1. 概要	21
3.3.1.2.2. <xmlfile> 要素の作成	22
3.3.1.3. <project> 構文	23
3.3.1.3.1. 概要	23
3.3.1.3.2. <project> 要素の作成	24
3.3.1.4. <filecontent> 構文	24
3.3.1.4.1. 概要	24
3.3.1.4.2. <filecontent> の作成	25
3.3.1.5. <file> 構文	25
3.3.1.5.1. 概要	25
3.3.1.5.2. <file> 要素の作成	25
3.3.1.6. <has-hint> 構文	26
3.3.1.6.1. 概要	26
3.3.1.6.2. <has-hint> の作成	27
3.3.1.7. <has-classification> 構文	27
3.3.1.7.1. 概要	27
3.3.1.7.2. <has-classification> の作成	27

3.3.1.8. <graph-query> 構文	27
3.3.1.8.1. 概要	27
3.3.1.8.2. <graph-query> の作成	28
3.3.1.9. <dependency> 構文	29
3.3.1.9.1. 概要	29
3.3.2. <perform> 構文	29
3.3.2.1. <classification> 構文	29
3.3.2.1.1. 概要	29
3.3.2.1.2. <classification> 要素属性	30
3.3.2.1.3. <classification> 子要素	30
3.3.2.2. <link> 構文	31
3.3.2.2.1. 概要	31
3.3.2.2.2. <link> 要素属性	32
3.3.2.3. <hint> 構文	32
3.3.2.3.1. 概要	32
3.3.2.3.2. <hint> 要素属性	32
3.3.2.3.3. <hint> 子要素	33
3.3.2.4. <xslt> 構文	34
3.3.2.4.1. 概要	34
3.3.2.4.2. <xslt> 要素属性	34
3.3.2.4.3. <xslt> 子要素	35
3.3.2.5. <lineitem> 構文	35
3.3.2.5.1. 概要	35
3.3.2.5.2. <lineitem> 要素属性	36
3.3.2.6. <iteration> 構文	36
3.3.2.6.1. 概要	36
3.3.2.6.2. <iteration> 要素属性	36
3.3.2.6.3. <iteration> 子要素	36
3.3.3. <where> 構文	37
3.4. MIGRATION TOOLKIT FOR APPLICATIONS へのルールの追加	37
第4章 XML ルールのテスト	39
4.1. テストルールの作成	39
4.1.1. XML ルール構造のテスト	39
4.1.2. XML ルール構文のテスト	39
4.1.2.1. <not> 構文	40
概要	40
4.1.2.2. <iterable-filter> 構文	40
概要	41
<iterable-filter> 要素属性	41
4.1.2.3. <classification-exists> 構文	41
<classification-exists> 要素属性	42
4.1.2.4. <hint-exists> 構文	42
<hint-exists> 要素属性	43
4.1.2.5. <fail> 構文	44
<fail> 要素属性	44
4.2. XML ルールの手動テスト	44
4.3. JUNIT を使用したルールのテスト	44
4.4. 検証レポートについて	46
4.4.1. 検証レポートの作成	46
4.4.2. 検証レポートのエラーメッセージ	47
第5章 ルールの上書き	49

5.1. ルールの上書き	49
5.2. ルールの無効化	50
第6章 カスタムルールカテゴリーの使用	51
カスタムカテゴリーの追加	51
カスタムカテゴリーへのルールの割り当て	51
付録A 参考資料	53
A.1. ルールのストーリーポイントについて	53
A.1.1. ストーリーポイントとは	53
A.1.2. ルールにおけるストーリーポイントの見積方法	53
A.1.3. タスクカテゴリー	53
A.2. 関連情報	54
A.2.1. 既存の MTA XML ルールの確認	54
A.2.1.1. Migration Toolkit for Applications XML ルールのフォークおよびクローン作成	54
A.2.2. リソース	55

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、[Red Hat CTO である Chris Wright のメッセージ](#) をご覧ください。

第1章 はじめに

1.1. ルール開発ガイドについて

本ガイドは、Migration Toolkit for Applications (MTA) ツールのカスタム XML ベースのルールを作成するエンジニア、コンサルタント、およびその他のユーザーを対象としています。

MTA を初めて使用する場合は、最初に [Migration Toolkit for Applications の概要](#)で、Migration Toolkit for Applications の機能とシステム要件の概要を確認することが推奨されます。また、CLI のインストール方法および実行方法が記載されている [CLI ガイド](#)を確認することが推奨されます。

MTA ソースコードベースに貢献したり、Java ベースのルールアドオンを提供する場合は、[Core Development Guide](#)を参照してください。

1.1.1. 本ガイドの <MTA_HOME> の使用

本ガイドでは、置き換え可能な変数 <MTA_HOME> を使用して、MTA インストールへのパスを示します。インストールディレクトリーは、**.zip** ファイルを抽出した **mta-cli-5.0.Final** ディレクトリーです。



注記

Windows オペレーティングシステムの場合は、**Path too long** エラーを防ぐために、**mta** という名前のディレクトリーに、MTA の **.zip** ファイルを展開する必要があります。

本ガイドで <MTA_HOME> が発生した場合は、これを MTA インストールへの実際のパスに置き換えます。

1.2. MTA ルールについて

Migration Toolkit for Applications (MTA) には、移行予定のアプリケーションが使用する API、テクノロジー、アーキテクチャーを分析するルールベースの移行ツールが含まれています。実際、MTA 分析プロセスは MTA ルールを使用して実装されます。MTA は内部でルールを使用して、アーカイブからのファイルの抽出、ファイルの逆コンパイル、ファイルタイプのスキャンと分類、XML およびその他のファイルコンテンツの分析、アプリケーションコードの分析、レポートの作成を行います。

MTA は、ルール実行結果に基づいてデータモデルを構築し、グラフデータベースにコンポーネントデータと関係を格納します。これにより、移行ルールやレポート目的に応じてクエリーおよび更新が可能になります。

MTA ルールでは、以下のルールパターンを使用します。

```
when(condition)
  perform(action)
otherwise(action)
```

MTA は、標準の包括的な移行ルールの包括的なセットをそのまま使用できます。アプリケーションにはカスタムライブラリーまたはコンポーネントが含まれる可能性があるため、MTA では独自のルールを作成して、既存のルールセットで対応していない可能性のあるコンポーネントやソフトウェアの使用を特定することができます。

第2章 ルールを使い始める

ルールを作成するか、クイックスタートを確認して、カスタム MTA ルールの作成を開始できます。

2.1. 最初の XML ルールの作成

本セクションでは、最初の MTA XML ベースのルールを作成してテストするプロセスを説明します。これは、すでに MTA がインストールされていることを前提としています。インストール手順は、[CLI ガイド](#) を参照してください。

この例では、アプリケーションが **<class-loading>** 要素を含む **jboss-web.xml** ファイルの定義を行うインスタンスを検出するルールを記述します。また、コードの移行方法を説明するドキュメントへのリンクを提供します。

ルールのディレクトリー構造の作成

最初のルールと、テストに使用するデータファイルを含むディレクトリー構造を作成します。

```
$ mkdir -p /home/<USER_NAME>/migration-rules/rules
$ mkdir -p /home/<USER_NAME>/migration-rules/data
```

このディレクトリー構造は、生成された MTA レポートを保持するためにも使用されます。

ルールをテストするためのデータの作成

1. **/home/<USER_NAME>/migration-rules/data/** サブディレクトリーに **jboss-web.xml** ファイルを作成します。
2. 以下の内容にコピーします。

```
<!DOCTYPE jboss-web PUBLIC "-//JBoss//DTD Web Application 4.2//EN"
"http://www.jboss.org/j2ee/dtd/jboss-web_4_2.dtd">
<jboss-web>
  <class-loading java2ClassLoadingCompliance="false">
    <loader-repository>
      seam.jboss.org:loader=@projectName@
      <loader-repository-config>java2ParentDelegation=false</loader-repository-config>
    </loader-repository>
  </class-loading>
</jboss-web>
```

ルールの作成

MTA XML ベースのルールは、以下のルールパターンを使用します。

```
when(condition)
  perform(action)
otherwise(action)
```

手順

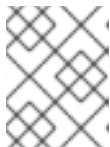
1. **/home/<USER_NAME>/migration-rules/rules/** ディレクトリーで、以下の内容が含まれる **JBoss5-web-class-loading.windup.xml** という名前のファイルを作成します。

```
<?xml version="1.0"?>
<ruleset id="<UNIQUE_RULESET_ID>"
```

```

xmlns="http://windup.jboss.org/schema/jboss-ruleset"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
<metadata>
  <description>
    <!-- Ruleset Description -->
  </description>
  <dependencies>
    <!-- Ruleset Dependencies -->
  </dependencies>
  <sourceTechnology id="<SOURCE_ID>" versionRange="
<SOURCE_VERSION_RANGE>"/>
  <targetTechnology id="<TARGET_ID>" versionRange="
<TARGET_VERSION_RANGE>"/>
  <tag>Reviewed-2015-05-01</tag>
</metadata>
<rules>
  <rule id="<UNIQUE_RULE_ID>">
    <when>
      <!-- Test for a condition here -->
    </when>
    <perform>
      <!-- Perform an action -->
    </perform>
  </rule>
</rules>
</ruleset>

```



注記

XML ファイル名には、**.windup.xml** または **.mta.xml** 拡張子が含まれている必要があります。それ以外の場合には、MTA は新しいルールを評価しません。

2. ruleset および rule に一意の識別子を追加します。

- **<UNIQUE_RULESET_ID>** を、適切なルールセット ID (例: **JBoss5-web-class-loading**) に置き換えます。
- **<UNIQUE_RULE_ID>** を、適切なルール ID (例: **JBoss5-web-class-loading_001**) に置き換えます。

3. 次のルールセットアドオン依存関係を追加します。

```

<dependencies>
  <addon id="org.jboss.windup.rules,windup-rules-javaee,3.0.0.Final"/>
  <addon id="org.jboss.windup.rules,windup-rules-java,3.0.0.Final"/>
</dependencies>

```

4. ソースおよびターゲットのテクノロジーを追加します。

- **<SOURCE_ID>** を **eap** に置き換えます。
- **<TARGET_ID>** を **eap** に置き換えます。

5. ソースおよびターゲットのテクノロジーバージョンを設定します。

- `<SOURCE_VERSION_RANGE>` を `(4,5)` に置き換えます。
- `<TARGET_VERSION_RANGE>` を `(6,)` に置き換えます。

詳細は、Apache Maven [version range specification](#) を参照してください。

6. **when** 条件を完了します。このルールは XML ファイルの一致をテストするため、ファイルの評価には `xmlfile` が使用されます。

`jboss-web` の子である `class-loading` 要素と一致するには、xpath 式 `jboss-web/class-loading` を使用します。

```
<when>
  <xmlfile matches="jboss-web/class-loading" />
</when>
```

7. このルールの **perform** アクションを完了します。

- 説明的なタイトルと作業レベル **1** を使用して分類を追加します。
- ヒントに情報メッセージと、移行の詳細を説明するドキュメントへのリンクを提供します。

```
<perform>
  <iteration>
    <classification title="JBoss Web Application Descriptor" effort="1"/>
    <hint title="JBoss Web XML class-loading element is no longer valid">
      <message>
        The class-loading element is no longer valid in the jboss-web.xml file.
      </message>
      <link href="https://access.redhat.com/documentation/ja-
JP/JBoss_Enterprise_Application_Platform/6.4/html-
single/Migration_Guide/index.html#Create_or_Modify_Files_That_Control_Class_Loading_i
n_JBoss_Enterprise_Application_Platform_6" title="Create or Modify Files That Control
Class Loading in JBoss EAP 6"/>
    </hint>
  </iteration>
</perform>
```

ルールは完了し、以下の例のようになるはずです。

```
<?xml version="1.0"?>
<ruleset id="JBoss5-web-class-loading"
  xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <metadata>
    <description>
      This ruleset looks for the class-loading element in a jboss-web.xml file, which is no longer
valid in JBoss EAP 6
    </description>
    <dependencies>
      <addon id="org.jboss.windup.rules,windup-rules-javaee,3.0.0.Final"/>
    </dependencies>
  </metadata>
</ruleset>
```

```

    <addon id="org.jboss.windup.rules,windup-rules-java,3.0.0.Final"/>
  </dependencies>
  <sourceTechnology id="eap" versionRange="(4,5)"/>
  <targetTechnology id="eap" versionRange="[6,)" />
</metadata>
<rules>
  <rule id="JBoss5-web-class-loading_001">
    <when>
      <xmlfile matches="jboss-web/class-loading" />
    </when>
    <perform>
      <iteration>
        <classification title="JBoss Web Application Descriptor" effort="1"/>
        <hint title="JBoss Web XML class-loading element is no longer valid">
          <message>
            The class-loading element is no longer valid in the jboss-web.xml file.
          </message>
          <link href="https://access.redhat.com/documentation/ja-
JP/JBoss_Enterprise_Application_Platform/6.4/html-
single/Migration_Guide/index.html#Create_or_Modify_Files_That_Control_Class_Loading_in_JBoss_Er
terprise_Application_Platform_6" title="Create or modify files that control class loading in JBoss EAP
6"/>
        </hint>
      </iteration>
    </perform>
  </rule>
</rules>
</ruleset>

```

ルールのインストール

MTA ルールは、ルールを適切なディレクトリーに配置してインストールされます。

JBoss5-web-class-loading.windup.xml ファイルを **<MTA_HOME>/rules/** ディレクトリーにコピーします。

```
$ cp /home/<USER_NAME>/migration-rules/rules/JBoss5-web-class-loading.windup.xml
<MTA_HOME>/rules/
```

ルールのテスト

端末を開き、以下のコマンドを実行して、テストファイルを入力引数として、出力レポートのディレクトリーとして渡します。

```
$ <MTA_HOME>/bin/mta-cli --sourceMode --input /home/<USER_NAME>/migration-rules/data --
output /home/<USER_NAME>/migration-rules/reports --target eap:6
```

以下の結果が表示されるはずです。

```
Report created: /home/<USER_NAME>/migration-rules/reports/index.html
Access it at this URL: file:///home/<USER_NAME>/migration-rules/reports/index.html
```

レポートの確認

レポートを確認して、予想される結果が表示されることを確認します。MTA レポートの詳細は、MTA のCLI ガイドの [レポートの確認](#) セクションを参照してください。

1. Web ブラウザーで **/home/<USER_NAME>/migration-rules/reports/index.html** を開きます。

2. ルールが実行されていることを確認します。

- a. 主な編集ページから、**Rule providers execution overview** リンクをクリックし、Rule Providers Execution Overview を開きます。
- b. **JBoss5-web-class-loading_001** ルールを探して、**Status?** が **Condition met** で、**Result?** が **success** であることを確認します。

図2.1 テストルール実行

JBoss5-web-class-loading Phase: MigrationRulesPhase					
Rule-ID	Rule	Statistics	Status?	Result?	Failure Cause
JBoss5-web-class-loading_001	<pre><rule id="JBoss5-web-class-loading_001" xmlns="http://windup.jboss.org/schema/jboss-ruleset"> <when> <xmlfile matches="jboss-web/class-loading"/> </when> <perform> <iteration> <classification effort="1" title="JBoss Web Application Descriptor"/> <hint title="JBoss Web XML class-loading element is no longer valid"> <message> The class-loading element is no longer valid in the jboss-web.xml file. </message> <link href="https://access.redhat.com/documentation/en-US /JBoss_Enterprise_Application_Platform/6.4/html-single/Migration_Guide /index.html#Create_or_Modify_Files_That_Control_Class_Loading_in_JBoss_Enterprise_Application_Platform_6" title="Create or Modify Files That Control Class Loading in JBoss EAP 6"/> </hint> </iteration> </perform> </rule></pre>	Vertices Created: 6 Edges Created: 11 Vertices Removed: 0 Edges Removed: 0	Condition met.	success	

3. ルールがテストデータと一致していることを確認します。

- a. メインの発行ページから、この例の **データ** であるアプリケーションまたは入力フォルダーの名前をクリックします。
- b. **Application Details** レポートリンクをクリックします。
- c. **jboss-web.xml** リンクをクリックして、**ソースレポート** を表示します。
<class-loading> 行が強調表示されており、カスタムルールからのヒントがインラインで表示されます。

図2.2 ルール一致

The screenshot displays the 'Source Report' for 'data/jboss-web.xml' in the Migration Toolkit for Applications. The report is organized into sections: 'Technologies' (listing JBoss Web XML, JBoss Web, JBoss-AS, JBoss-AS-Descriptor, and JBoss-AS-Deployment) and 'Story Points'. The first story point, 'JBoss Web XML class-loading element is no longer valid', is highlighted in green. The report also includes a hint about the class-loading element being no longer valid in the jboss-web.xml file, with a link to the JBoss EAP 6 Migration Guide.

ファイルの上部には、一致するルールの分類が表示されます。リンクアイコンを使用する

と、そのルールの詳細を表示できます。この例では、**jboss-web.xml** ファイルが 1 つのストーリーポイントを生成した別のルール (**JBoss web application descriptor (jboss-web.xml)**) と一致することに注意してください。これは、カスタムルールから 1 つのストーリーポイントと組み合わせて、このファイルの合計部分を 2 にまとめます。

2.2. MIGRATION TOOLKIT FOR APPLICATIONS クイックスタートの確認

Migration Toolkit for Applications クイックスタートは、カスタム Java ベースのルールアドオンおよび XML ルールの作成方法に関する作業例を提供します。カスタムルールを作成する際の開始点として使用することができます。

各クイックスタートには、そのクイックスタートの指示が含まれる **README.adoc** ファイルがあります。

クイックスタートの最新バージョンの Zip アーカイブファイルをダウンロードできます。ソースコードを使用する場合は、**windup-quickstarts** プロジェクトリポジトリをフォークし、クローンできます。

最新のクイックスタートのダウンロード

クイックスタートの最新リリースをダウンロードできます。

手順

1. ブラウザーを起動し、<https://github.com/windup/windup-quickstarts/releases> に移動します。
2. 最新のリリースをクリックして、Zip アーカイブファイルをローカルファイルシステムにダウンロードします。
3. アーカイブファイルをローカルディレクトリーに展開します。
クイックスタートの **README.adoc** ファイルを確認できます。

クイックスタートの GitHub プロジェクトのフォークおよびクローン作成

ローカルマシンでクイックスタートの Github プロジェクトをフォークし、クローン作成することができます。

前提条件

- **git** クライアントがインストールされていること。

手順

1. GitHub ページ [Migration Toolkit for Applications quickstart](#) で **Fork** をクリックし、自分用の Git にプロジェクトを作成します。フォークされた GitHub リポジトリの URL は **https://github.com/<YOUR_USER_NAME>/windup-quickstarts.git** のようになります。
2. Migration Toolkit for Applications クイックスタートリポジトリをローカルのファイルシステムにクローンします。

```
$ git clone https://github.com/<YOUR_USER_NAME>/windup-quickstarts.git
```

これにより、ローカルファイルシステムに **windup-quickstarts** ディレクトリーが作成されます。

3. 新規作成されたディレクトリーに移動します。


```
$ cd windup-quickstarts/
```

4. 最新のコード更新を取得するには、元のフォークしたリポジトリに変更を取得できるように、リモートの **upstream** リポジトリを追加します。

```
$ git remote add upstream https://github.com/windup/windup-quickstarts.git
```

5. **upstream** リポジトリから最新のファイルをダウンロードします。

```
$ git fetch upstream
```

第3章 XML ルールの作成

3.1. XML ルール構造

本セクションでは、XML ルールの基本構造を説明します。すべての XML ルールは、rulesets 内の要素として定義されます。詳細は、[MTA XML rule schema](#) を参照してください。

3.1.1. ルールセット

ルールセットとは、特定の移行領域をターゲットとする1つ以上のルールのグループです。これは **<ruleset>** 要素の基本構造です。

- **<ruleset id="<UNIQUE_RULESET_ID>">**: これを MTA ルールセットとして定義し、一意のルールセット ID を指定します。
 - **<metadata>**: ルールセットに関するメタデータ。
 - **<description>**: ルールセットの説明。
 - **<dependencies/>**: このルールセットに必要なルールアドオンです。
 - **<sourceTechnology/>**: ソーステクノロジー。
 - **<targetTechnology/>**: ターゲットテクノロジー。
 - **<overrideRules/>**: **true** に設定した場合は、MTA で配布されるコアルールセットと同じ ID で、このルールセットのルールを上書きします。ルールセット ID とルール id の両方がコアルールセット内のルールと一致する必要があり、ルールは無視されます。デフォルトは **false** です。
 - **<rules>**: 個々のルールのセット。
 - **<rule id="<UNIQUE_RULE_ID>">** ルールを定義して、一意の ID を指定します。ルール ID の一部としてルールセット ID を含めることが推奨されます (例: **<UNIQUE_RULESET_ID_UNIQUE_RULE_ID>**)。ルールセットには1つ以上のルールを定義できます。
 - **<when>**: 照合する条件。
 - **<perform>**: ルール条件が一致したときに実行されるアクション。
 - **<otherwise>**: ルール条件が一致しない場合に実行されるアクション。この要素は、**<perform>** 要素と同じ子要素を取ります。
 - **<where>**: パラメーターとして定義される文字列パターン。このパターンは、ルール定義の他の場所で使用できます。
 - **<file-mapping/>**: エクステンションをグラフタイプにマップします。
 - **<package-mapping/>**: パッケージパターン (正規表現) から組織名にマッピングします。

3.1.2. 事前定義されたルール

MTA は、一般的な移行要件に事前定義されたルールを提供します。これらのコア MTA ルールは、MTA インストールの **<MTA_HOME>/rules/migration-core/** にあります。

以下は、プロプライエタリーユーティリティークラスで一致するコア MTA ルールの例になります。

```
<?xml version="1.0"?>
<ruleset xmlns="http://windup.jboss.org/schema/jboss-ruleset" id="weblogic"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">

  <metadata>
    <description>
      This ruleset provides analysis of WebLogic proprietary classes and constructs that may
      require individual attention when migrating to JBoss EAP 6+.
    </description>
    <dependencies>
      <addon id="org.jboss.windup.rules.windup-rules-javaee,2.0.1.Final" />
      <addon id="org.jboss.windup.rules.windup-rules-java,2.0.0.Final" />
    </dependencies>
    <sourceTechnology id="weblogic" />
    <targetTechnology id="eap" versionRange="[6,)" />
    <tag>reviewed-2015-06-02</tag>
    <tag>weblogic</tag>
  </metadata>
  <rules>
    ...
    <rule id="weblogic-02000">
      <when>
        <javaClass references="weblogic.utils.StringUtils.{*}" />
      </when>
      <perform>
        <hint title="WebLogic StringUtils usage" effort="1" category-id="mandatory">
          <message>Replace with the `StringUtils` class from Apache Commons.</message>
          <link href="https://commons.apache.org/proper/commons-lang/" title="Apache Commons
Lang" />
        </hint>
      </perform>
    </rule>
    ...
  </rules>
</ruleset>
```

3.2. 基本的な XML ルールの作成

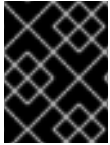
本セクションでは、MTA XML ルールを作成する方法を説明します。これは、すでに MTA がインストールされていることを前提としています。インストール手順は、MTA の [CLI ガイド](#) を参照してください。

3.2.1. 基本的な XML ルールテンプレートの作成

MTA XML ルールは **conditions** および **actions** で設定され、以下のルールパターンを使用します。

```
when(condition)
  perform(action)
otherwise(action)
```

以下の内容で、XML ルールの基本的な構文であるファイルを作成します。



重要

XML ファイル名には、**.windup.xml** または **.mta.xml** 拡張子が含まれている必要があります。それ以外の場合には、MTA は新しいルールを評価しません。

```
<?xml version="1.0"?>
<ruleset id="unique-ruleset-id"
  xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <metadata>
    <!-- Metadata about the rule including a description,
      source technology, target technology, and any
      add-on dependencies -->
  </metadata>
  <rules>
    <rule id="unique-ruleset-id-01000">
      <when>
        <!-- Test a condition... -->
      </when>
      <perform>
        <!-- Perform this action when condition is satisfied -->
      </perform>
      <otherwise>
        <!-- Perform this action when condition is not satisfied -->
      </otherwise>
    </rule>
  </rules>
</ruleset>
```

3.2.2. ruleset メタデータの作成

XML ルールセットの **metadata** 要素は、説明、ソーステクノロジーおよびターゲットテクノロジー、アドオンの依存関係などのルールセットに関する追加情報を提供します。メタデータによりタグの指定が可能になり、ルールセットに関する追加情報を提供できます。

<metadata> の例

```
<ruleset id="unique-ruleset-id"
  xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <metadata>
    <description>
      This is the description.
    </description>
    <dependencies>
      <addon id="org.jboss.windup.rules,windup-rules-javaee,2.0.1.Final"/>
      <addon id="org.jboss.windup.rules,windup-rules-java,2.0.0.Final"/>
    </dependencies>
```

```

<sourceTechnology id="weblogic" versionRange="(10,12]"/>
<sourceTechnology id="ejb" versionRange="(2,3]"/>
<targetTechnology id="eap" versionRange="(5,6]"/>
<targetTechnology id="ejb" versionRange="(2,3]"/>
<tag>require-stateless</tag>
<tag>require-nofilesystem-io</tag>
<executeAfter>AfterRulesetId</executeAfter>
<executeBefore>BeforeRulesetId</executeBefore>
</metadata>
<rules>
...
</rules>
</ruleset>

```

3.2.3. ルールの作成

個々のルールは **<rules>** 要素に含まれます。これには、1つ以上の **when** 条件と実行アクションが含まれます。

有効なルール構文は、[XML rule schema](#) を参照してください。

3.2.3.1. <when> 条件の作成

XML ルール **<when>** 要素は条件をテストします。以下は、有効な **<when>** 条件の一覧です。

要素	説明
<and>	標準の論理 and 演算子。
<filecontent>	ファイル内の文字列またはテキスト (プロパティファイルなど) を検索します。
<file-mapping>	内部に保存されているファイルモデルにファイル名を定義します。
<javaclass>	Java クラスで一致のテスト。
<javaclass-ignore>	検出の処理で無視する javaclass を除外します。
<not>	標準の論理 not 演算子。
<or>	標準の論理 or 演算子。
<package-mapping>	組織またはライブラリーにパッケージ名を定義します。
<project>	依存関係などのプロジェクトの特性をテストします。
<true>	常に一致します。
<xmlfile>	XML ファイルで一致をテストします。

特定の構文は、Java クラス、XML ファイル、プロジェクト、またはファイルコンテンツを評価するルールを作成するかどうかによって異なります。

3.2.3.2. <perform> アクションの作成

XML ルール **<perform>** 要素は、条件が満たされるとアクションを実行します。このルールで許可される操作には、アプリケーションリソースの分類、移行手順のインラインヒント、移行情報へのリンク、およびプロジェクトライン項目の報告が含まれます。以下は、有効な **<perform>** アクションの一覧です。

要素	説明
<classification>	この操作により、ファイル全体に適用するメタデータが追加されます。たとえば、Java クラスが JMS メッセージリスナーの場合、ファイル全体に適用される情報が含まれる JMS Message Listener というタイトルで分類を追加できます。ファイル全体の作業レベルを設定することもできます。
<hint>	この操作により、ファイル内の行にメタデータが追加されます。これにより、コードのセクションを移行するヒントまたはインライン情報が提供されます。
<iteration>	これは、ルール内で定義される暗黙的な変数または明示的な変数を繰り返し処理することを指定します。
<lineitem>	これは、アプリケーションの概要ページに表示される高レベルのメッセージを提供します。
<link>	これにより、追加の情報や移行タスクに関するドキュメントへの HTML リンクが提供されます。
<xslt>	これは、XML ファイルの変換方法を指定します。

3.3. XML ルール構文

3.3.1. <when> 構文

ルールの **when** で許可される条件は、[GraphOperation](#) を拡張する必要があり、現在、Java クラス、XML ファイル、プロジェクト、およびファイルコンテンツの評価が含まれます。XML ルールは Java ベースのルールアドオンの後にモデル化されるため、関連する Java クラスの JavaDocs へのリンクは、その動作をよりよく理解するために提供されます。

完全な XML ルールスキーマは <http://windup.jboss.org/schema/windup-jboss-ruleset.xsd> にあります。

次のセクションでは、より一般的な XML **when** ルール条件を説明します。

- <javaclass> 条件構文
- <xmlfile> 条件構文
- <project> 条件構文
- <filecontent> 条件構文

- <file> 条件構文
- <has-hint> 条件構文
- <has-classification> 条件構文
- <graph-query> 条件構文
- <dependency> 条件構文

デフォルトでは、**when** ルール条件が複数指定された場合に、ルールが一致するには、すべての条件が満たされる必要があります。

3.3.1.1. <javaclass> syntax

3.3.1.1.1. 概要

<javaclass> 要素を使用して、インポート、メソッド、変数宣言、アノテーション、クラス実装、および Java クラスに関連する他の項目を検索します。<javaclass> 条件のより詳細な理解は、[JavaClass](#) クラスの JavaDoc を参照してください。

以下は、WebLogic 固有の Apache XML パッケージをテストするルールの例です。

```
<rule id="weblogic-03000">
  <when>
    <javaclass references="weblogic.apache.xml.*" />
  </when>
  <perform>
    <hint title="WebLogic Specific Apache XML Package" effort="1" category-id="mandatory">
      <message>
        Code using this package should be replaced with code using the org.apache.xml package
        from [Apache
          Xerces](http://xerces.apache.org/).
      </message>
    </hint>
  </perform>
</rule>
```

3.3.1.1.2. <javaclass> 要素の作成

3.3.1.1.2.1. <javaclass> 要素属性

属性名	タイプ	説明
-----	-----	----

属性名	タイプ	説明
references	CLASS_NAME	<p>一致するパッケージまたはクラス名。ワイルドカード文字を使用できます。この属性は必須です。</p> <div>  <p>注記</p> <p>パフォーマンス上の理由から、ワイルドカード文字を使用して参照を開始しないでください。たとえば、<code>{web}.apache.xml.*</code> の代わりに <code>weblogic.apache.xml.*</code> を使用します。</p> </div> <pre>references="weblogic.apache.xml.*"</pre>
matchesSource	STRING	<p>一致する正確な正規表現。これは、ハードコーディングされた文字列を区別する場合に便利です。この属性は必須です。</p> <pre>matchesSource="log4j.logger"</pre>
as	VARIABLE_NAME	<p>ルールに割り当てられた変数名。これにより、後で処理した参照として使用できるようになります。以下の from 属性を参照してください。</p> <pre>as="MyEjbRule"</pre>
from	VARIABLE_NAME	<p>as VARIABLE_NAME として識別される前の検索からのフィルターリングされた結果から検索クエリーを開始します。</p> <pre>from="MyEjbRule"</pre>
in	PATH_FILTER	<p>この正規表現 (regex) の命名パターンに一致する入力ファイルをフィルターします。ワイルドカード文字を使用できます。</p> <pre>in="*File1"</pre>

3.3.1.1.2.2. <javaclass> 子要素

子要素	説明
<location>	<p>参照が Java クラスで見つかった場所。場所は、アノテーション、フィールドおよび変数宣言、インポート、およびメソッドを参照できます。有効な値の完全なリストは、TypeReferenceLocation の JavaDoc を参照してください。</p> <pre><location>IMPORT</location></pre>

子要素	説明
<annotation-literal>	<p>アノテーション内のリテラル値に一致します。</p> <p>以下の例は @MyAnnotation(myvalue="test") で一致します。</p> <pre data-bbox="427 347 1142 517"><javaclass references="org.package.MyAnnotation"> <location>ANNOTATION</location> <annotation-literal name="myvalue" pattern="test"/> </javaclass></pre> <p>この場合、<javaclass> はアノテーション (@MyAnnotation) を参照します。そのため、最上位アノテーションフィルター <annotation-literal> は name 属性を指定する必要があります。アノテーションが付いたクラスを参照する <javaclass> が参照される場合、使用される最上位アノテーションフィルターは <annotation-type> になります。</p>
<annotation-type>	<p>特定のアノテーションタイプで一致します。アノテーション要素に対して照合されるサブ条件を指定できます。</p> <p>以下の例は、@MyAnnotation(myvalue="test") アノテーションが付けられた Calendar フィールド宣言で一致します。</p> <pre data-bbox="427 963 1193 1193"><javaclass references="java.util.Calendar"> <location>FIELD_DECLARATION</location> <annotation-type pattern="org.package.MyAnnotation"> <annotation-literal name="myvalue" pattern="test"/> </annotation-type> </javaclass></pre>
<annotation-list>	<p>アノテーション内の配列の項目で一致します。配列インデックスが指定されていない場合は、配列内の任意のアイテムに適用されると条件が一致します。この要素と照合するサブ条件を指定できます。</p> <p>以下の例は @MyAnnotation(mylist={"one","two"}) で一致しています。</p> <pre data-bbox="427 1456 1137 1686"><javaclass references="org.package.MyAnnotation" > <location>ANNOTATION</location> <annotation-list name="mylist"> <annotation-literal pattern="two"/> </annotation-list> </javaclass></pre> <p>この場合、<javaclass> はアノテーション (@MyAnnotation) を参照します。そのため、最上位アノテーションフィルター <annotation-list> は name 属性を指定する必要があります。アノテーションが付いたクラスを参照する <javaclass> が参照される場合、使用される最上位アノテーションフィルターは <annotation-type> になります。</p>

3.3.1.2. <xmlfile> 構文

3.3.1.2.1. 概要

<xmlfile> 要素を使用して XML ファイルで情報を検索します。**<xmlfile>** 条件の理解を深めるには、[XmlFile](#) クラスの JavaDoc を参照してください。

以下は、XML ファイルをテストするルールの例です。

```
<rule id="<UNIQUE_RULE_ID>">
  <when>
    <xmlfile matches="/w:web-app/w:resource-ref/w:res-auth[text() = 'Container']">
      <namespace prefix="w" uri="http://java.sun.com/xml/ns/javaee"/>
    </xmlfile>
  </when>
  <perform>
    <hint title="Title for Hint from XML">
      <message>Container Auth</message>
    </hint>
    <xslt description="Example XSLT Conversion" extension="-converted-example.xml"
      template="/exampleconversion.xsl"/>
  </perform>
</rule>
```

3.3.1.2.2. <xmlfile> 要素の作成

3.3.1.2.2.1. <xmlfile> 要素属性

属性名	タイプ	説明
matches	XPATH	XML ファイル条件で一致します。 <div> <pre>matches="/w:web-app/w:resource-ref/w:res-auth[text() = 'Container']"</pre> </div>
xpathResultMatch	XPATH_RESULT_STRING	指定の正規表現に一致する結果を返します。 <div> <pre><xmlfile matches="//foo/text()" xpathResultMatch="Text from foo."/></pre> </div>
as	VARIABLE_NAME	ルールに割り当てられた変数名。これにより、後で処理した参照として使用できるようになります。以下の from 属性を参照してください。 <div> <pre>as="MyEjbRule"</pre> </div>
in	PATH_FILTER	この正規表現 (regex) の命名パターンに一致する入力ファイルをフィルターを設定するのに使用されます。ワイルドカード文字を使用できます。 <div> <pre>in="{*}File1"</pre> </div>

属性名	タイプ	説明
from	VARIABLE_NAME	as VARIABLE_NAME として識別される前の検索からのフィルターリングされた結果から検索クエリーを開始します。 from="MyEjbRule"
public-id	PUBLIC_ID	DTD public-id 正規表現。 public-id="public"

3.3.1.2.2.2. <xmlfile> がカスタム関数と一致 します。

matches 属性は、ルール変数スタックに一致した値の設定など、 有用な影響を及ぼす可能性のある組み込みのカスタム XPath 関数をいくつか使用できます。

機能	説明
windup:matches()	XPath 式を文字列と照合します。MTA パラメーター化プレースホルダーが含まれている可能性があります。 matches="windup:matches(//foo/@class, '{javaclassname}')" これは、すべての <foo/> 要素と class 属性に一致し、それぞれの反復についてその値を javaclassname パラメーターに保存します。

3.3.1.2.2.3. <xmlfile> 子要素

子要素	説明
<namespace>	XML ファイルで参照される名前空間。この要素には、 prefix と uri の 2 つの任意の属性が含まれます。 <namespace prefix="abc" uri="http://maven.apache.org/POM/4.0.0"/>

3.3.1.3. <project> 構文

3.3.1.3.1. 概要

<project> 要素を使用して、プロジェクトの特性について Maven POM ファイルをクエリーします。<project> 条件をよりよく理解するには、 [Project](#) クラスの JavaDoc を参照してください。

以下は、2.0.0.Final と 2.2.0.Final の間で JUnit 依存関係バージョンを確認するルールの例になります。

```

<rule id="UNIQUE_RULE_ID">
  <when>
    <project>
      <artifact groupId="junit" artifactId="junit" fromVersion="2.0.0.Final" toVersion="2.2.0.Final"/>
    </project>
  </when>
  <perform>
    <lineitem message="The project uses junit with the version between 2.0.0.Final and
2.2.0.Final"/>
  </perform>
</rule>

```

3.3.1.3.2. <project> 要素の作成

3.3.1.3.2.1. <project> 要素属性

<project> 要素は、プロジェクトの Maven POM ファイルと照合するために使用されます。この条件を使用して、プロジェクトの依存関係をクエリーできます。属性自体はありません。

3.3.1.3.2.2. <project> 子要素

子要素	説明
<artifact>	プロジェクトの依存関係に対してクエリーするために <project> 内で使用されるサブ条件。 <artifact> 要素属性は以下で説明されています。

3.3.1.3.2.3. <artifact> 要素属性

属性名	タイプ	説明
groupId	PROJECT_GROUP_ID	依存関係の <groupId> プロジェクトで一致します。
artifactId	PROJECT_ARTIFACT_ID	依存関係の <artifactId> プロジェクトで一致します。
fromVersion	FROM_VERSION	アーティファクトの下位バージョンバインドを指定します。例: 2.0.0.Final
toVersion	TO_VERSION	アーティファクトの上位バージョンバインドを指定します。例: 2.2.0.Final

3.3.1.4. <filecontent> 構文

3.3.1.4.1. 概要

<filecontent> 要素を使用して、ファイル内の文字列またはテキストを検索します (例: Properties ファイルの行)。 **<filecontent>** 条件のより詳細な理解は、 [FileContent](#) クラスの JavaDoc を参照してください。

3.3.1.4.2. <filecontent> の作成

3.3.1.4.2.1. <filecontent> 要素属性

属性名	タイプ	説明
pattern	String	指定されたパラメーター化された文字列に対してファイルの内容を一致させます。この属性は必須です。
filename	String	提供されるパラメーター化された文字列に対してファイル名を一致させます。
as	VARIABLE_NAME	ルールに割り当てられた変数名。これにより、後で処理した参照として使用できるようになります。以下の from 属性を参照してください。 <div> <div></div> as="MyEjbRule" </div>
from	VARIABLE_NAME	as VARIABLE_NAME として識別される前の検索からのフィルタリングされた結果から検索クエリーを開始します。 <div> <div></div> from="MyEjbRule" </div>

3.3.1.5. <file> 構文

3.3.1.5.1. 概要

<file> 要素を使用して、特定の名前のファイル (例: **ibm-webservices-ext.xmi**) ファイルが存在するかどうかを検索します。<file> 条件のより詳細な理解は、[File](#) クラスの JavaDoc を参照してください。

3.3.1.5.2. <file> 要素の作成

3.3.1.5.2.1. <file> 要素属性

属性名	タイプ	説明
filename	String	提供されるパラメーター化された文字列に対してファイル名を一致させます。この属性は必須です。
as	VARIABLE_NAME	ルールに割り当てられた変数名。これにより、後で処理した参照として使用できるようになります。以下の from 属性を参照してください。 <div> <div></div> as="MyEjbRule" </div>

属性名	タイプ	説明
from	VARIABLE_NAME	<p>as VARIABLE_NAME として識別される前の検索からのフィルタリングされた結果から検索クエリーを開始します。</p> <p>以下に例を示します。</p> <pre>from="MyEjbRule"</pre>

3.3.1.6. <has-hint> 構文

3.3.1.6.1. 概要

<has-hint> 要素を使用して、ファイルまたは行にすでに関連付けられているヒントがあるかどうかをテストします。これは主に、ヒントがすでに存在する場合に起動を防止するため、また他の条件が適用されない場合のデフォルト実行のルールを実装するために使用されます。<has-hint> 条件のより詳しい情報は、[HasHint](#) クラスの JavaDoc を参照してください。

以下は、IBM JMS 宛先メッセージのヒントが存在するかどうかを確認し、含まれていない場合にそれを含めるルールの例です。

```
<rule id="websphere-jms-eap7-03000">
  <when>
    <javaclass references="{package}.{prefix}{type}Message" />
  </when>
  <perform>
    <iteration>
      <when>
        <not>
          <has-hint />
        </not>
      </when>
      <perform>
        <hint title="IBM JMS destination message" effort="1" category-id="mandatory">
          <message>
            JMS `{package}.{prefix}{type}Message` messages represent the actual data passed through
            JMS destinations. This reference should be
            replaced with the Java EE standard API `javax.jms.{type}Message`.
          </message>
          <link href="https://docs.oracle.com/javaee/7/tutorial/jms-concepts003.htm#sthref2271"
            title="Java EE 7 JMS Tutorial - Message API" />
          <tag>jms</tag>
          <tag>websphere</tag>
        </hint>
      </perform>
    </iteration>
  </perform>
  <where param="type">
    <matches pattern="(Text|Stream|Object|Map|Bytes)?" />
  </where>
  <where param="prefix">
    <matches pattern="(JMS|MQe|MQ)" />
  </where>
</rule>
```

```

<where param="package">
  <matches pattern="com.ibm(\..*)?\.\jms" />
</where>
</rule>

```

3.3.1.6.2. <has-hint> の作成

<has-hint> 要素は、ファイルまたは行のヒントが存在するかどうかを判断するために使用されます。子要素はありません。

3.3.1.6.2.1. <has-hint> 要素属性

属性名	タイプ	説明
message	String	ヒントを提供されたメッセージ文字列と照合できるようにする任意の引数。

3.3.1.7. <has-classification> 構文

3.3.1.7.1. 概要

<has-classification> 要素を使用して、ファイルまたは行に分類があるかどうかをテストします。これは主に、分類がすでに存在する場合に起動を防止するため、また他の条件が適用されない場合のデフォルト実行のルールを実装するために使用されます。**<has-classification>** 条件のより詳細な理解は、[HasClassification](#) クラスの JavaDoc を参照してください。

3.3.1.7.2. <has-classification> の作成

has-classification 要素は、指定された分類が存在するかどうかを判断するために使用されます。子要素はありません。

3.3.1.7.2.1. <has-classification> 要素属性

属性名	タイプ	説明
title	String	分類と照合する任意のタイトル。

3.3.1.8. <graph-query> 構文

3.3.1.8.1. 概要

<graph-query> 要素を使用して、生成されたグラフで任意の要素を検索します。この要素は、主に特定のアーカイブを検索するために使用されます。**<graph-query>** 条件のより詳細な理解は、[QueryHandler](#) クラスの JavaDoc を参照してください。

以下は、**ehcache** パッケージが見つかったかどうかを判断するためにテストするルールの例です。

```

<rule id="embedded-cache-libraries-01000">
  <when>
    <graph-query discriminator="JarArchiveModel">

```

```

    <property name="fileName" searchType="regex">.*ehcache.*\jar$</property>
  </graph-query>
</when>
<perform>
  <classification title="Caching - Ehcache embedded library" category-id="cloud-mandatory"
effort="5">
    <description>
      The application embeds an Ehcache library.

      Cloud readiness issue as potential state information that is not persisted to a backing
service.
    </description>
  </classification>
  <technology-tag level="INFORMATIONAL">Ehcache (embedded)</technology-tag>
</perform>
</rule>

```

3.3.1.8.2. <graph-query> の作成

3.3.1.8.2.1. <graph-query> 要素属性

属性名	タイプ	説明
discriminator	MODEL_TYPE	検索に使用するモデルのタイプ。これは任意の有効なモデルにすることができます。ただし、アーカイブの調査には JarArchiveModel を使用することが推奨されます。この属性は必須です。
as	VARIABLE_NAME	ルールに割り当てられた変数名。これにより、後で処理した参照として使用できるようになります。以下の from 属性を参照してください。 <pre>as="MyEjbRule"</pre>
from	VARIABLE_NAME	as VARIABLE_NAME として識別される前の検索からのフィルターリングされた結果から検索クエリーを開始します。 <pre>from="MyEjbRule"</pre>

3.3.1.8.2.2. <graph-query> プロパティ

プロパティ名	タイプ	説明
name	String	選択したモデル内で照合する属性の名前。ファイルベースのモデルを使用する場合は、 fileName と一致することが推奨されます。この属性は必須です。
type	property-type	予想されるプロパティの型 (STRING または BOOLEAN のいずれか) を定義します。

プロパティ名	タイプ	説明
searchType	property-search-type	条件の一致方法を定義します。 equals に設定された場合は、完全に一致する必要があります。 regex を使用すると、正規表現を使用できます。

3.3.1.9. <dependency> 構文

3.3.1.9.1. 概要

<dependency> 要素を使用して、アプリケーションの POM ファイル内で定義された依存関係を検索し、ターゲットランタイムでサポートされるかどうかを判別します。

以下は、1.6.0 までのバージョンがある **org.springframework.boot** グループに属するすべてのアーティファクトを確認するルールの例です。

```
<rule id="springboot-00001">
  <!-- rule condition, when it could be fired -->
  <when>
    <dependency groupId="org.springframework.boot" artifactId="{*}" toVersion="1.6.0" />
  </when>
  <!-- rule operation, what to do if it is fired -->
  <perform>
    <hint title="Unsupported version of Spring Boot" effort="3" category-id="mandatory">
      <message>Spring Boot has to be updated to Spring Boot 2.0 before being able to be
migrated to a version supported by Red Hat Runtimes</message>
      <link href="https://access.redhat.com/articles/3349341" title="RHOAR Spring Boot Supported
Configurations" />
      <link href="https://access.redhat.com/articles/3348731" title="RHOAR Component Details
Overview" />
      <link href="https://github.com/spring-projects/spring-boot/wiki/Spring-Boot-2.0-Migration-
Guide" title="Spring Boot 2.0 Migration Guide" />
    </hint>
  </perform>
</rule>
```

3.3.2. <perform> 構文

ルールの **perform** セクションで利用可能な操作には、アプリケーションリソースの分類、移行手順のインラインヒント、移行情報へのリンク、およびプロジェクトラインレポートレポートが含まれます。XML ルールは Java ベースのルールアドオンの後にモデル化されるため、関連する Java クラスの JavaDocs へのリンクは、その動作をよりよく理解するために提供されます。

[完全な XML ルールスキーマ](#) を表示できます。

次のセクションでは、より一般的な XML ルール実行アクションを説明します。

3.3.2.1. <classification> 構文

3.3.2.1.1. 概要

<classification> 要素は、ルールに一致するアプリケーションリソースを識別または分類するのに使用されます。これは、レポートに表示されるタイトル、作業レベル、およびリソースの分類の移行方法に関する追加情報へのリンクも提供します。**<classification>** アクションの詳細は、[Classification](#) クラスの JavaDoc を参照してください。

以下は、リソースを WebLogic EAR アプリケーションデプロイメント記述子ファイルとして分類するルールの例です。

```
<rule id="XmlWebLogicRules_10vvyf">
  <when>
    <xmlfile as="default" matches="/*[local-name()='weblogic-application']"></xmlfile>
  </when>
  <perform>
    <iteration>
      <classification title="Weblogic EAR Application Descriptor" effort="3"/>
    </iteration>
  </perform>
</rule>
```

3.3.2.1.2. <classification> 要素属性

属性名	タイプ	説明
title	STRING	このリソースに指定されたタイトル。この属性は必須です。 <pre>title="JBoss Seam Components"</pre>
effort	BYTE	このリソースに割り当てられた作業量レベル。 <pre>effort="2"</pre>
category-id	STRING	MTA_HOME/rules/migration-core/core.windup.categories.xml で定義されているカテゴリへの参照。デフォルトのカテゴリは mandatory 、 optional 、 potential 、 information です。 <pre>category-id="mandatory"</pre>
/	VARIABLE_NAME	指定の参照用に新しい分類を作成します。 <pre>of="MySeamRule"</pre>

3.3.2.1.3. <classification> 子要素

子要素	説明
-----	----

子要素	説明
<link>	<p>詳細情報のリンク URI およびテキストタイトルを提供します。</p> <pre><classification title="Websphere Startup Service" effort="4"> <link href="http://docs.oracle.com/javaee/6/api/javax/ejb/Singleton.html" title="EJB3.1 Singleton Bean"/> <link href="http://docs.oracle.com/javaee/6/api/javax/ejb/Startup.html" title="EJB3.1 Startup Bean"/> </classification></pre>
<tag>	<p>分類に関する追加のカスタム情報を提供します。</p> <pre><tag>Seam3</tag></pre>
<description>	<p>このリソースの説明</p> <pre><description>JBoss Seam components must be replaced</description></pre>

3.3.2.2. <link> 構文

3.3.2.2.1. 概要

<link> 要素は、分類またはヒントで使用され、情報コンテンツへのリンクが提供されます。<link> アクションのより詳細な理解は、[Link](#) クラスの JavaDoc を参照してください。

以下は、追加情報へのリンクを作成するルールの例です。

```
<rule id="SeamToCDIRules_2fmb">
  <when>
    <javaclass references="org.jboss.seam.*" as="default"/>
  </when>
  <perform>
    <iteration>
      <classification title="SEAM Component" effort="1">
        <link href="http://www.seamframework.org/Seam3/Seam2ToSeam3MigrationNotes"
title="Seam 2 to Seam 3 Migration Notes"/>
        <link href="http://docs.jboss.org/weld/reference/latest/en-US/html/example.html" title="JSF
Web Application Example"/>
        <link href="http://docs.jboss.org/weld/reference/latest/en-US/html/contexts.html"
title="JBoss Context Documentation"/>
        <link href="http://www.andygibson.net/blog/tutorial/cdi-conversations-part-2/" title="CDI
Conversations Blog Post"/>
      </classification>
    </iteration>
  </perform>
</rule>
```

3.3.2.2.2. <link> 要素属性

属性名	タイプ	説明
href	URI	参照リンクの URI。 <pre>href="https://access.redhat.com/articles/1249423"</pre>
title	STRING	リンクのタイトル。 <pre>title="Migrate WebLogic Proprietary Servlet Annotations"</pre>

3.3.2.3. <hint> 構文

3.3.2.3.1. 概要

<hint> 要素は、コードのセクションの移行方法に関するヒントまたはインライン情報を提供するために使用されます。<hint> アクションについてはよく理解するには、[Hint](#) クラスの JavaDoc を参照してください。

以下は、ヒントを作成するルールの例です。

```
<rule id="WebLogicWebServiceRules_8jyqn">
  <when>
    <javaclass
      references="weblogic.wsee.connection.transport.http.HttpTransportInfo.setUsername({*})"
      as="default">
      <location>METHOD</location>
    </javaclass>
  </when>
  <perform>
    <iteration>
      <hint title="Proprietary web-service" category-id="mandatory" effort="3">
        <message>Replace proprietary web-service authentication with JAX-WS
standards.</message>
        <link href="http://java-x.blogspot.com/2009/03/invoking-web-services-through-proxy.html"
title="JAX-WS Proxy Password Example"/>
      </hint>
    </iteration>
  </perform>
</rule>
```

3.3.2.3.2. <hint> 要素属性

属性名	タイプ	説明
-----	-----	----

属性名	タイプ	説明
title	STRING	指定された文字列を使用してこのヒントを入力します。この属性は必須です。 <pre>title="JBoss Seam Component Hint"</pre>
category-id	STRING	MTA_HOME/rules/migration-core/core.windup.categories.xml で定義されているカテゴリへの参照。デフォルトのカテゴリは mandatory 、 optional 、 potential 、 information です。 <pre>category-id="mandatory"</pre>
in	VARIABLE_NAME	指定された変数によって解決される FileLocationModel に新しいヒントを作成します。 <pre>in="Foo"</pre>
effort	BYTE	このリソースに割り当てられた作業量レベル。 <pre>effort="2"</pre>

3.3.2.3.3. <hint> 子要素

子要素	説明
<message>	移行ヒントを記述するメッセージ。 <pre><message>EJB 2.0 is deprecated</message></pre>
<link>	情報コンテンツへのリンクを特定または分類します。 <pre><link href="http://docs.oracle.com/javaee/6/api/" title="Java Platform, Enterprise Edition 6 API Specification" /></pre>
<tag>	この hint のカスタムタグを定義します。 <pre><tag>Needs review</tag></pre>

子要素	説明
<quickfix>	<p>ルール条件を満たす際に迅速な修正を実行するために IDE プラグインによって使用される情報が含まれています。</p> <pre><quickfix name="slink-qf" type="REPLACE"> <replacement>h:link</replacement> <search>s:link</search> </quickfix></pre>

3.3.2.4. <xslt> 構文

3.3.2.4.1. 概要

<xslt> 要素は XML ファイルの変換方法を指定します。<xslt> アクションのより詳しい情報は、[XSLTTransformation](#) クラスの JavaDoc を参照してください。

以下は、XSLT アクションを定義するルールの例です。

```
<rule id="XmlWebLogicRules_6bcvk">
  <when>
    <xmlfile as="default" matches="/weblogic-ejb-jar"/>
  </when>
  <perform>
    <iteration>
      <classification title="Weblogic EJB XML" effort="3"/>
      <xslt title="JBoss EJB Descriptor (Windup-Generated)"
        template="transformations/xslt/weblogic-ejb-to-jboss.xml" extension="-jboss.xml"/>
    </iteration>
  </perform>
</rule>
```

3.3.2.4.2. <xslt> 要素属性

属性名	タイプ	説明
title	STRING	<p>レポート内のこの XSLTTransformation のタイトルを設定します。この属性は必須です。</p> <pre>title="XSLT Transformed Output"</pre>
/	STRING	<p>指定の参照の変換を新たに作成します。</p> <pre>of="testVariable_instance"</pre>

属性名	タイプ	説明
extension	STRING	この XSLTTransformation のエクステンションを設定します。 この属性は必須です。 <div>extension="-result.html"</div>
template	STRING	XML テンプレートを設定します。この属性は必須です。 <div>template="simpleXSLT.xsl"</div>
effort	BYTE	変換に必要な作業量レベル。

3.3.2.4.3. <xslt> 子要素

子要素	説明
<xslt-parameter>	XSLTTransformation パラメーターをプロパティ値のペアとして指定 <div><xslt-parameter property="title" value="EJB Transformation"/></div>

3.3.2.5. <lineitem> 構文

3.3.2.5.1. 概要

<lineitem> 要素は、非推奨のライブラリーを置き換える必要や、潜在的なクラスの読み込み問題を解決する必要がある場合など、アプリケーションの一般的な移行要件を提供するために使用されます。この情報は、プロジェクトまたはアプリケーションの概要ページに表示されます。<lineitem> アクションについてよく理解するには、[LineItem](#) クラスの JavaDoc を参照してください。

以下は、行項目メッセージを作成するルールの例です。

```
<rule id="weblogic_servlet_annotation_1000">
  <when>
    <javaclass references="weblogic.servlet.annotation.WLServlet" as="default">
      <location>ANNOTATION</location>
    </javaclass>
  </when>
  <perform>
    <hint effort="1">
      <message>Replace the proprietary WebLogic @WLServlet annotation with the Java EE 6
standard @WebServlet annotation.</message>
      <link href="https://access.redhat.com/articles/1249423" title="Migrate WebLogic Proprietary
Servlet Annotations" />
      <lineitem message="Proprietary WebLogic @WLServlet annotation found in file."/>
    </hint>
  </perform>
</rule>
```

3.3.2.5.2. <lineitem> 要素属性

属性名	タイプ	説明
message	STRING	行項目メッセージ。 <pre>message="Proprietary code found."</pre>

3.3.2.6. <iteration> 構文

3.3.2.6.1. 概要

<iteration> 要素は、ルール内で定義される暗黙的な変数または明示的な変数を繰り返し処理するように指定します。<iteration> アクションについてよく理解するには、[Iteration](#) の JavaDoc を参照してください。

以下は、反復を実行するルールの例です。

```
<rule id="jboss-eap5-xml-19000">
  <when>
    <xmlfile as="jboss-app" matches="/jboss-app"/>
    <xmlfile as="jboss-app-no-DTD" matches="/jboss-app" public-id=""/>
  </when>
  <perform>
    <iteration over="jboss-app">
      <classification title="JBoss application Descriptor" effort="5"/>
    </iteration>
    <iteration over="jboss-app-no-DTD">
      <classification title="JBoss application descriptor with missing DTD" effort="5"/>
    </iteration>
    <iteration over="jboss-app-no-DTD">
      <xslt title="JBoss application descriptor - JBoss 5 (Windup-generated)"
        template="transformations/xslt/jboss-app-to-jboss5.xsl" extension="-jboss5.xml"/>
    </iteration>
  </perform>
</rule>
```

3.3.2.6.2. <iteration> 要素属性

属性名	タイプ	説明
over	VARIABLE_NAME	この VARIABLE_NAME で特定された条件を繰り返します。 <pre>over="jboss-app"</pre>

3.3.2.6.3. <iteration> 子要素

子要素	説明
<iteration>	子要素には、 when 条件と、アクションの iteration 、 classification 、 hint 、 xslt 、 lineitem 、および otherwise が含まれます。

3.3.3. <where> 構文

XML ルールの他の要素で使用する、一致するパターンを指定するパラメーターを定義できます。これにより、複雑なマッチング式のパターンを単純化することができます。

<where> 要素を使用してパラメーターを定義します。param 属性を使用してパラメーター名を指定し、<matches> 要素を使用してパターンを指定します。このパラメーターは、構文 {<PARAM_NAME>} を使用して、ルール定義の他の場所で参照できます。

完全な XML ルールスキーマ を表示できます。

以下のルール例は、(activeio|activemq) パターンを指定する subpackage という名前のパラメーターを定義します。

```
<rule id="generic-catchall-00600">
  <when>
    <javaclass references="org.apache.{subpackage}.{*}">
    </javaclass>
  </when>
  <perform>
    ...
  </perform>
  <where param="subpackage">
    <matches pattern="(activeio|activemq)" />
  </where>
</rule>
```

subpackage で定義されたパターンは <javaclass> references 属性で置き換えられます。これにより、ルールは org.apache.activeio.* パッケージおよび org.apache.activemq.* パッケージで一致します。

3.4. MIGRATION TOOLKIT FOR APPLICATIONS へのルールの追加

Migration Toolkit for Applications ルールは、ルールを適切な MTA ディレクトリーにコピーしてインストールされます。MTA は、ルールをスキャンします。これは、次の場所にある .windup.xml または .mta.xml 拡張子のファイルです。

- MTA コマンドラインで --userRulesDirectory 引数で指定されたディレクトリー。
- <MTA_HOME>/rules/ directory.<MTA_HOME> は、Migration Toolkit for Applications 実行可能ファイルをインストールし、実行するディレクトリーです。
- \${user.home}/.mta/rules/ ディレクトリー。このディレクトリーは、最初に行われた時点で MTA により作成されます。これには、ルール、アドオン、および MTA ログが含まれます。



注記

Windows オペレーティングシステムでは、ルールは **\Documents and Settings\
<USER_NAME>\.mta\rules** または **\Users\
<USER_NAME>\.mta\rules** にあります。

第4章 XML ルールのテスト

XML ルールを作成したら、テストルールを作成して、そのルールが機能するようにする必要があります。

4.1. テストルールの作成

テストルールは、以下の違いで、テストルールを作成するプロセスを使用して作成されます。

- テストルールは、テストするルールの下にある **tests/** ディレクトリーに置く必要があります。
- テストクラスなどのデータは、**tests/** ディレクトリーの下に **data/** ディレクトリーに配置する必要があります。
- テストルールは、**.mta.test.xml** エクステンションを使用する必要があります。
- これらのルールは、Test XML Rule Structure で定義された構造を使用します。

さらに、テストするルールの名前に続くテストルールを作成することが推奨されます。たとえば、ファイル名 **proprietary-rule.mta.xml** でルールを作成した場合、テストルールは **proprietary-rule.mta.test.xml** と呼ばれます。

4.1.1. XML ルール構造のテスト

すべてのテスト XML ルールは、1つ以上のルールセット **rulesets** を含む **ruletests** 内の要素として定義されます。詳細は、[MTA XML rule schema](#) を参照してください。

ルールテストは、移行の特定の領域を対象とする1つ以上のテストのグループです。これは **<ruletest>** 要素の基本構造です。

- **<ruletest id="<RULE_TOPIC>-test">**: 一意の MTA ルールテストとして定義し、固有のルールテスト ID を提供します。
 - **<testDataPath>**: テストに使用するクラスやファイルなどのデータへのパスを定義します。
 - **<sourceMode>**: データで渡されたにソースファイルのみが含まれるかどうかを示します。EAR、WAR、JAR などのアーカイブが使用されている場合は、**false** に設定する必要があります。デフォルトは **true** です。
 - **<rulePath>**: テストされるルールへのパス。これは、テストするルールの名前で終了します。
 - **<ruleset>**: テストケースのロジックが含まれるルールセット。これらは Ruleset で定義されるものと同じです。

4.1.2. XML ルール構文のテスト

標準の XML ルール構文のタグに加え、以下の **when** 条件がテストルールの作成に一般的に使用されます。

- **<not>**
- **<iterable-filter>**
- **<classification-exists>**

- **<hint-exists>**

標準 **perform action** 構文のタグのほかに、以下の **perform** 条件がテストルールのアクションとして一般的に使用されます。

- **<fail>**

4.1.2.1. <not> 構文

概要

<not> 要素は標準の論理演算子 **not** で、通常は条件を満たさない場合に **<fail>** を実行するために使用されます。

以下は、分析の最後に特定のメッセージのみが存在する場合に失敗したテストルールの例です。

```
<ruletest xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  id="proprietary-servlet-test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <testDataPath>data/</testDataPath>
  <rulePath>../proprietary-servlet.mta.xml</rulePath>
  <ruleset>
    <rules>
      <rule id="proprietary-servlet-01000-test">
        <when>
          <!--
            The <not> will perform a logical not operator on the elements within.
          -->
          <not>
            <!--
              The defined <iterable-filter> has a size of 1. This rule will only match on a single instance of the
              defined hint.
            -->
            <iterable-filter size="1">
              <hint-exists message="Replace the proprietary @ProprietaryServlet annotation with the Java
              EE 7 standard @WebServlet annotation*" />
            </iterable-filter>
          </not>
        </when>
        <!--
          This <perform> element is only executed if the previous <when> condition is false.
          This ensures that it only executes if there is not a single instance of the defined hint.
        -->
        <perform>
          <fail message="Hint for @ProprietaryServlet was not found!" />
        </perform>
      </rule>
    </rules>
  </ruleset>
</ruletest>
```

<not> 要素には固有の属性や子要素がありません。

4.1.2.2. <iterable-filter> 構文

概要

<iterable-filter> 要素は、条件の検証回数をカウントします。詳細は、[IterableFilter](#) クラスを参照してください。

以下は、指定のメッセージの 4 つのインスタンスを検索する例です。

```
<ruletest xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  id="proprietary-servlet-test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <testDataPath>data/</testDataPath>
  <rulePath>../proprietary-servlet.mta.xml</rulePath>
  <ruleset>
    <rules>
      <rule id="proprietary-servlet-03000-test">
        <when>
          <!--
            The <not> will perform a logical not operator on the elements within.
          -->
          <not>
            <!--
              The defined <iterable-filter> has a size of 4. This rule will only match on four instances of the
              defined hint.
            -->
            <iterable-filter size="4">
              <hint-exists message="Replace the proprietary @ProprietaryInitParam annotation with the
              Java EE 7 standard @WebInitParam annotation*" />
            </iterable-filter>
          </not>
        </when>
        <!--
          This <perform> element is only executed if the previous <when> condition is false.
          In this configuration, it only executes if there are not four instances of the defined hint.
        -->
        <perform>
          <fail message="Hint for @ProprietaryInitParam was not found!" />
        </perform>
      </rule>
    </rules>
  </ruleset>
</ruletest>
```

<iterable-filter> 要素には一意の子要素がありません。

<iterable-filter> 要素属性

属性名	タイプ	説明
size	integer	検証される回数。

4.1.2.3. <classification-exists> 構文

<classification-exists> 要素は、分析に特定の分類タイトルが含まれているかどうかを判断します。詳細は、[ClassificationExists](#) クラスを参照してください。



重要

特殊文字 ([, ' など) が含まれるメッセージをテストする場合は、各特殊文字をバックスラッシュ (\) でエスケープして正しく一致させる必要があります。

以下は、特定の分類タイトルを検索する例です。

```
<ruletest xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  id="proprietary-servlet-test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <testDataPath>data/</testDataPath>
  <rulePath>../weblogic.mta.xml</rulePath>
  <ruleset>
    <rules>
      <rule id="weblogic-01000-test">
        <when>
          <!--
            The <not> will perform a logical not operator on the elements within.
          -->
          <not>
            <!--
              The defined <classification-exists> is attempting to match on the defined title.
              This classification would have been generated by a matching <classification title="WebLogic
scheduled job" .../> rule.
            -->
            <classification-exists classification="WebLogic scheduled job" />
          </not>
        </when>
        <!--
          This <perform> element is only executed if the previous <when> condition is false.
          In this configuration, it only executes if there is not a matching classification.
        -->
        <perform>
          <fail message="Triggerable not found" />
        </perform>
      </rule>
    </rules>
  </ruleset>
</ruletest>
```

<classification-exists> には一意の子要素がありません。

<classification-exists> 要素属性

属性名	タイプ	説明
classification	String	検索する <classification> title。
in	String	定義されたファイル名が含まれるファイルに一致することを制限する任意の引数。

4.1.2.4. **<hint-exists>** 構文

<hint-exists> 要素は、分析に特定のヒントが含まれているかどうかを判断します。これは定義されたメッセージのインスタンスをすべて検索し、通常は **<message>** 要素内の開始または特定のクラスを検索するために使用されます。詳細は、[HintExists](#) クラスを参照してください。



重要

特殊文字 ([, ' など) が含まれるメッセージをテストする場合は、各特殊文字をバックslash (\) でエスケープして正しく一致させる必要があります。

以下は、特定のヒントを検索する例です。

```
<ruletest xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  id="proprietary-servlet-test" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <testDataPath>data/</testDataPath>
  <rulePath>../weblogic.windup.xml</rulePath>
  <ruleset>
    <rules>
      <rule id="weblogic-eap7-05000-test">
        <when>
          <!--
            The <not> will perform a logical not operator on the elements within.
          -->
          <not>
            <!--
              The defined <hint-exists> is attempting to match on the defined message.
              This message would have been generated by a matching <message> element on the <hint>
              condition.
            -->
            <hint-exists message="Replace with the Java EE standard method
.java\transaction\TransactionManager.resume\(Transaction tx\)." />
          </not>
        </when>
        <!--
          This <perform> element is only executed if the previous <when> condition is false.
          In this configuration, it only executes if there is not a matching hint.
        -->
        <perform>
          <fail message="Note to replace with standard TransactionManager.resume is missing!" />
        </perform>
      </rule>
    </rules>
  </ruleset>
</ruletest>
```

<hint-exists> 要素には固有の子要素がありません。

<hint-exists> 要素属性

属性名	タイプ	説明
message	String	検索する <hint> メッセージ。

属性名	タイプ	説明
in	String	指定のファイル名を参照する InLineHintModels に一致することを制限する任意の引数。

4.1.2.5. <fail> 構文

<fail> 要素は実行を失敗として報告し、関連するメッセージを表示します。通常、これは、条件が満たされない場合のみメッセージを表示するために <not> 条件とともに使用されます。

<fail> 要素には、一意の子要素がありません。

<fail> 要素属性

属性名	タイプ	説明
message	String	表示されるメッセージ。

4.2. XML ルールの手動テスト

アプリケーションファイルに対して XML ルールを実行してテストすることができます。

```
$ <MTA_HOME>/bin/mta-cli [--sourceMode] --input <INPUT_ARCHIVE_OR_FOLDER> --output
<OUTPUT_REPORT_DIRECTORY> --target <TARGET_TECHNOLOGY> --packages
<PACKAGE_1> <PACKAGE_2> <PACKAGE_N>
```

以下の結果が表示されるはずです。

```
Report created: <OUTPUT_REPORT_DIRECTORY>/index.html
Access it at this URL: file:///<OUTPUT_REPORT_DIRECTORY>/index.html
```

MTA の実行方法のその他の例は、Migration Toolkit for Applications の [CLI ガイド](#) に記載されています。

4.3. JUNIT を使用したルールのテスト

テストルールが作成されたら、JUnit テストの一部として分析し、ルールが実行のすべての基準を満たしていることを確認できます。MTA ルールリポジトリの **WindupRulesMultipleTests** クラスは、複数のルールを同時にテストし、不足している要件についてフィードバックを提供するように設計されています。

前提条件

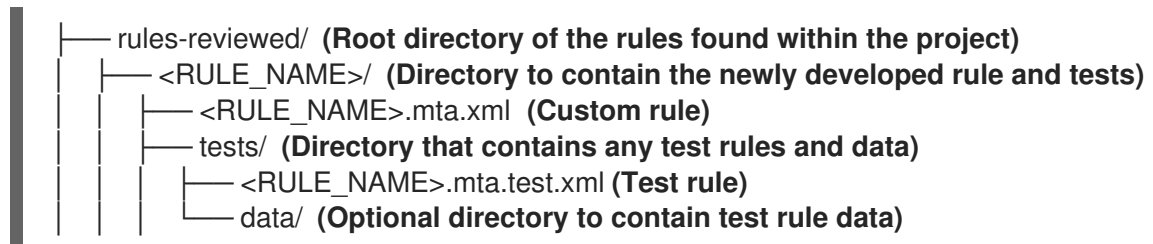
- MTA XML ルールをフォークし、クローンを作成します。このリポジトリの場所は <RULESETS_REPO> と呼ばれます。
- テスト XML ルールを作成します。

JUnit テスト設定の作成

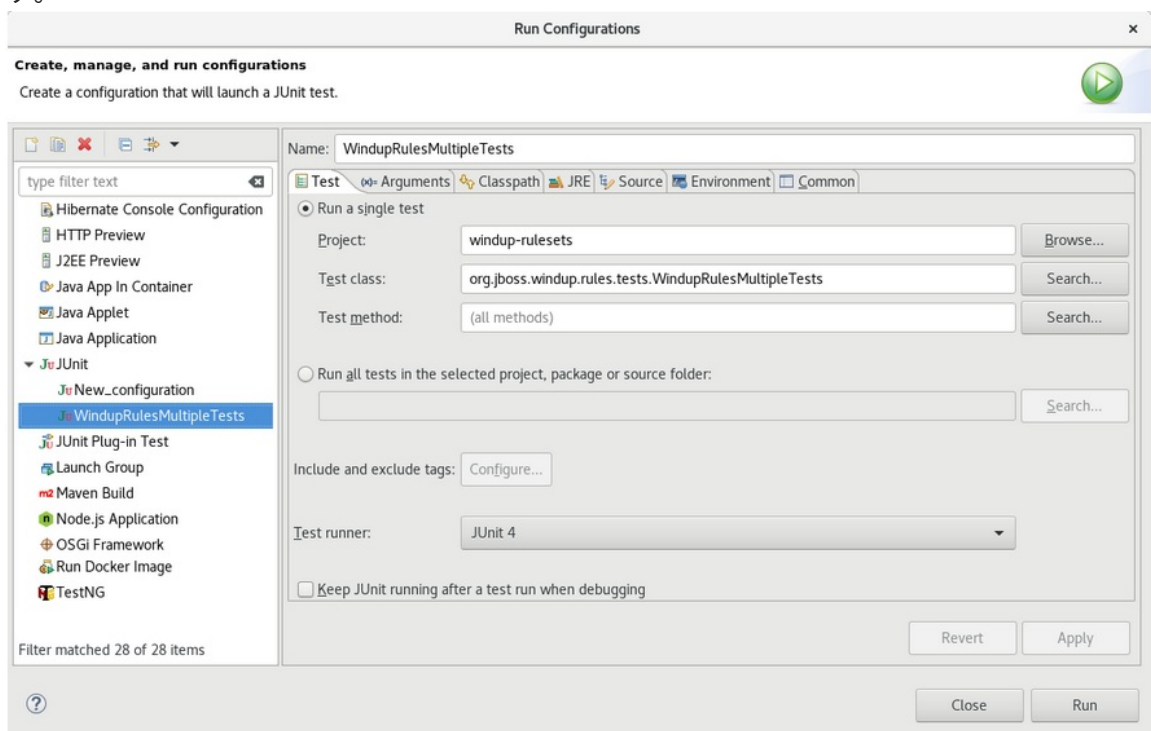
以下の手順では、Red Hat CodeReady Studio を使用して JUnit テストを作成する方法を説明します。別の IDE を使用する場合は、JUnit テストの作成方法について IDE のドキュメントを参照してください。

1. MTA ルールセットリポジトリを IDE にインポートします。
2. カスタムルールに対応するテストとデータと共に `</path/to/RULESETS_REPO>/rules-reviewed/<RULE_NAME>/` にコピーします。これにより、以下のディレクトリ構造を作成する必要があります。

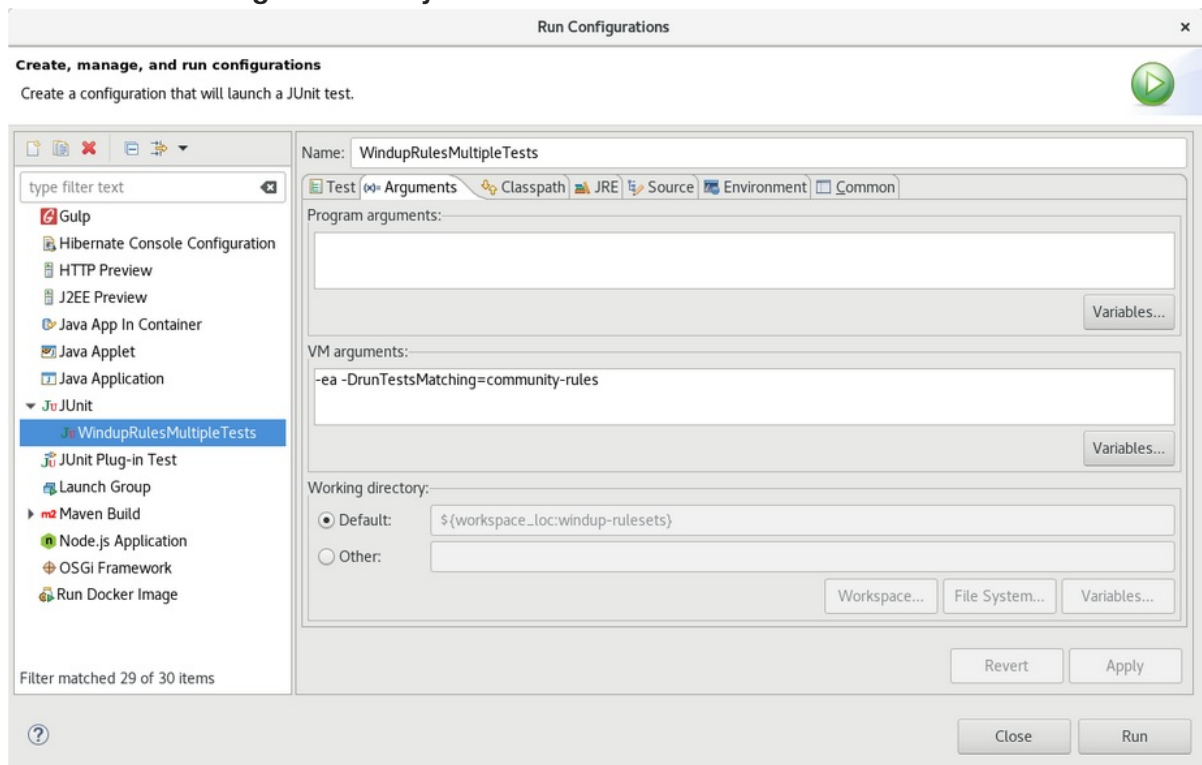
ディレクトリ構造



3. トップメニューバーから **Run** を選択します。
4. 表示されるドロップダウンメニューから **Run Configurations...** を選択します。
5. 左側のオプションから **JUnit** を右クリックし、**New** を選択します。
6. 以下のコマンドを入力します。
 - **Name:** JUnit テストの名前 (例: **WindupRulesMultipleTests**)。
 - **Project:** これが **windup-rulesets** に設定されていることを確認します。
 - **Test class:** これを **org.jboss.windup.rules.tests.WindupRulesMultipleTests** に設定します。



7. **Arguments** タブを選択し、**-DrunTestsMatching=<RULE_NAME>** 仮想マシン引数を追加します。たとえば、ルール名が **community-rules** であった場合は、以下のイメージのように **-DrunTestsMatching=community-rules** を追加します。



8. 右下隅の **Run** をクリックしてテストを開始します。

実行が完了すると、結果を分析できるようになります。すべてのテストに合格した場合は、テストルール形式の問題はありません。それ以外の場合は、テストの失敗で発生した各問題に対処することをお勧めします。

4.4. 検証レポートについて

検証レポートは、テストルールおよび失敗の詳細を提供し、以下のセクションが含まれます。

- **概要**
本セクションでは、実行したテストの総数とエラーと失敗数を報告します。レポートが生成されるまでの合計成功率と所要時間を秒単位で表示します。
- **パッケージ一覧**
本セクションでは、各パッケージに対して実行されたテストの数を説明し、エラーと失敗の数を報告します。分析する各パッケージの成功率と所要時間を秒単位で表示します。

追加のテストケースが定義されない限り、**org.jboss.windup.rules.tests** という名前の1つのパッケージが表示されます。
- **テストケース**
本セクションでは、テストケースを説明します。それぞれの失敗には、エラーのソースを示す人間が判読できる行など、アサーションのスタックトレースを表示するために拡張できる **Details** セクションが含まれます。

4.4.1. 検証レポートの作成

カスタムルールの検証レポートを作成できます。

前提条件

- MTA XML ルールをフォークし、クローンする必要があります。
- 検証するには、1つ以上のテスト XML ルールが必要です。

手順

1. ローカルの **windup-rulesets** リポジトリに移動します。
2. カスタムルールのディレクトリー (**windup-rulesets/rules-reviewed/myTests**) を作成し、テストします。
3. カスタムルールおよびテストを、**windup-rulesets/rules-reviewed/<myTests>** ディレクトリーにコピーします。
4. **windup-rulesets** リポジトリのルートディレクトリーから以下のコマンドを実行します。

```
$ mvn -Dtest=WindupRulesMultipleTests -DrunTestsMatching=<myTests> clean
<myReport>:report 1 2
```

- 1 カスタムルールおよびテストが含まれるディレクトリーを指定します。 - **runTestsMatching** 引数を省略すると、検証レポートにはすべてのテストが含まれ、生成にかかる時間が非常に長くなります。
- 2 レポート名を指定します。

検証レポートは、**windup-rulesets/target/site/** リポジトリに作成されます。

4.4.2. 検証レポートのエラーメッセージ

検証レポートには、ルールおよびテストの実行中に発生したエラーが含まれます。

以下の表にはエラーメッセージとエラーの解決方法が記載されています。

表4.1 検証レポートのエラーメッセージ

エラーメッセージ	説明	解決策
No test file matching rule	このエラーは、対応するテストファイルがないルールファイルが存在すると発生します。	既存のルールにテストファイルを作成します。
Test rule ids <RULE_NAME> not found!	このエラーは、対応するルールテストがないルールが存在する場合に出力されます。	既存のルールのテストを作成します。
XML parse fail on file <FILE_NAME>	XML ファイルの構文は無効であり、ルールバリデーターにより正常に解析できません。	無効な構文を修正します。

エラーメッセージ	説明	解決策
<testDataPath> タグのテストファイルパスが見つかりませんでした。ファイルのテストを行うのに期待されるパス: <RULE_DATA_PATH>	テストルール内の <testDataPath> タグで定義されるパスにファイルはありません。	<testDataPath> タグに定義されたパスを作成し、必要なすべてのデータファイルがこのディレクトリーに置かれていることを確認します。
id="<RULE_ID>" のルールは実行されませんでした。	この検証中、提供された ID を持つルールは実行されません。	指定したルールで定義された条件に一致するテストデータファイルが存在することを確認します。

第5章 ルールの上書き

MTA で (またはカスタムルールでも)、配布されるコアルールを上書きできます。たとえば、ルールに一致する条件、作業、またはヒントテキストを変更できます。これは、元のルールのコピーを作成し、それをルールオーバーライドとしてマークし、必要な調整を行うことで行われます。

ルールを無効にするには、空の `<rule>` 要素を使用してルールのオーバーライドを作成します。

5.1. ルールの上書き

コアまたはカスタムルールを上書きできます。

手順

1. 上書きするルールが含まれる XML ファイルをカスタムの rules ディレクトリーにコピーします。
カスタムルールは、`<MTA_HOME>/rules`、`${user.home}/.mta/rules/`、または `--userRulesDirectory` コマンドライン引数で指定されたディレクトリーに配置できます。
2. 上書きするルールの `<rule>` 要素のみが含まれるように XML ファイルを編集します。



注記

新しいルールセットによって上書きされない元のルールセットは通常通りに実行されます。

3. 同じルールとルールセット ID を保持するようにしてください。元のルール XML をコピーすると、ID が一致します。
4. `<overrideRules>true</overrideRules>` 要素をルールセットメタデータに追加します。
5. ルール定義を更新します。
ルール定義では、すべてを変更することができます。新しいルールは、元のルールを完全に上書きします。

以下のルールは、**weblogic** ルールセットの **weblogic-02000** ルールの **effort** を **1** から **3** に変更します。

ルール上書き定義の例

```
<?xml version="1.0"?>
<ruleset id="weblogic"
  xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd"> ❶
  <metadata>
    ...
    <overrideRules>true</overrideRules> ❷
  </metadata>
  <rules>
    <rule id="weblogic-02000" xmlns="http://windup.jboss.org/schema/jboss-ruleset"> ❸
      <when>
        <javaclass references="weblogic.utils.StringUtils.{*}">
```

```

        </when>
        <perform>
            <hint effort="3" category-id="mandatory" title="WebLogic StringUtils Usage"> 4
                <message>Replace with the StringUtils class from Apache Commons.</message>
                <link href="https://commons.apache.org/proper/commons-lang/" title="Apache Commons
Lang"/>
                <tag>weblogic</tag>
            </hint>
        </perform>
    </rule>
</rules>
</ruleset>

```

- 1 ruleset id が元の ruleset id と一致していることを確認します。
- 2 <overrideRules>true</overrideRules> を <metadata> セクションに追加します。
- 3 rule id が元の rule id と一致していることを確認します。
- 4 effort が更新されました。

MTA を実行すると、このルールは同じルール ID で元のルールを上書きします。Rule Provider Executions Overview の内容を確認して、新規ルールが使用されていることを確認できます。

5.2. ルールの無効化

ルールを無効にするには、以下の例に従って、空の <rule> 要素でルールオーバーライド定義を作成します。

ルールを無効にするルールオーバーライドの定義の例

```

<?xml version="1.0"?>
<ruleset id="weblogic"
  xmlns="http://windup.jboss.org/schema/jboss-ruleset"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://windup.jboss.org/schema/jboss-ruleset
http://windup.jboss.org/schema/jboss-ruleset/windup-jboss-ruleset.xsd">
  <metadata>
    ...
    <overrideRules>true</overrideRules>
  </metadata>
  <rules>
    <rule id="weblogic-02000" xmlns="http://windup.jboss.org/schema/jboss-ruleset">
      1
    </rule>
  </rules>
</ruleset>

```

- 1 <rule> 要素は空であるため、weblogic ルールセットの weblogic-02000 ルールが無効になります。

第6章 カスタムルールカテゴリーの使用

カスタムルールカテゴリーを作成し、MTA ルールを割り当てることができます。



注記

MTA は従来の **severity** フィールドでルールを処理しますが、新しい **category-id** フィールドを使用するようにカスタムルールを更新する必要があります。

カスタムカテゴリーの追加

カスタムカテゴリーをルールカテゴリーファイルに追加できます。

手順

1. `<MTA_HOME>/rules/migration-core/core.windup.categories.xml` にあるルールカテゴリーファイルを編集します。
2. 新しい **<category>** 要素を追加し、以下のパラメーターを入力します。
 - **id**: MTA ルールがカテゴリーの参照に使用する ID。
 - **priority**: 他のカテゴリーに対するソートの優先順位です。値が最も低いカテゴリーが最初に表示されます。
 - **name**: カテゴリーの表示名。
 - **description**: カテゴリーの説明。

カスタムルールカテゴリーの例

```
<?xml version="1.0"?>
<categories>
  ...
  <category id="custom-category" priority="20000">
    <name>Custom Category</name>
    <description>This is a custom category.</description>
  </category>
</categories>
```

このカテゴリーは MTA ルールで参照できます。

カスタムカテゴリーへのルールの割り当て

新しいカスタムカテゴリーにルールを割り当てることができます。

手順

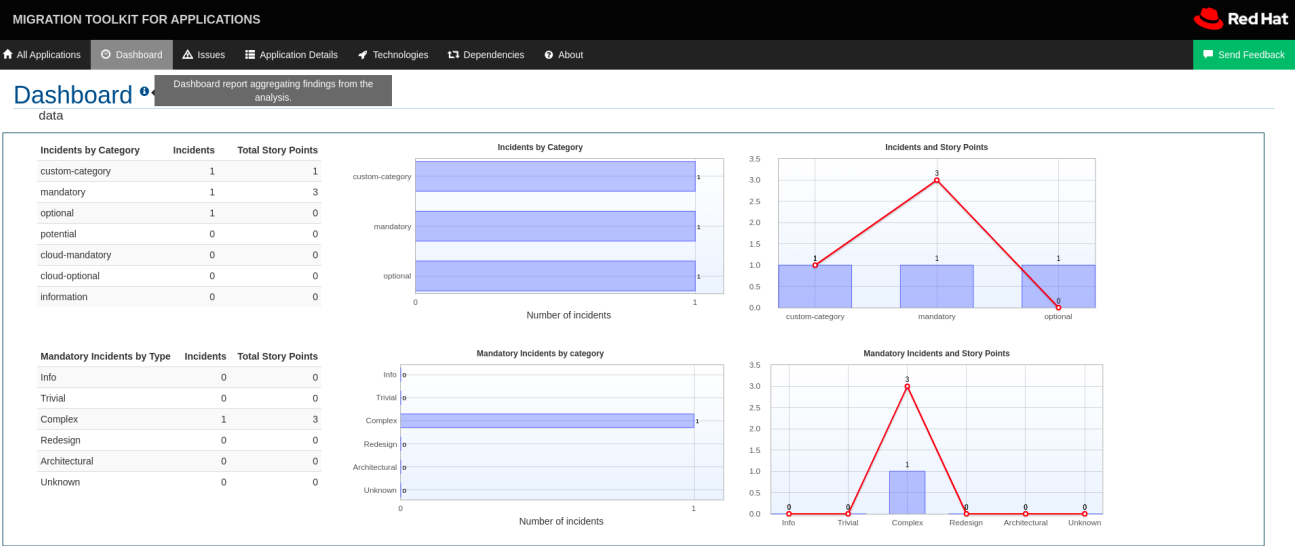
MTA ルールで、以下の例のように **category-id** フィールドを更新します。

```
<rule id="rule-id">
  <when>
    ...
  </when>
  <perform>
    <hint title="Rule Title" effort="1" category-id="custom-category">
      <message>Hint message.</message>
```

```
</hint>
</perform>
</rule>
```

このルール条件が満たされると、このルールによって識別されるインシデントはカスタムカテゴリーを使用します。カスタムカテゴリーはダッシュボードおよび Issues レポートに表示されます。

図6.1 ダッシュボードのカスタムカテゴリー



付録A 参考資料

A.1. ルールのストーリーポイントについて

A.1.1. ストーリーポイントとは

ストーリーポイントは、アジャイルソフトウェア開発で一般的に使用される抽象メトリクスで、機能や変更を実装するのに必要な作業量を予測します。

Migration Toolkit for Applications はストーリーポイントを使用して、特定のアプリケーションコンストラクトとアプリケーション全体を移行するために必要な作業のレベルを表現します。必ずしも工数に変換される訳ではありませんが、この値はタスク全体で一貫性を持たせる必要があります。

A.1.2. ルールにおけるストーリーポイントの見積方法

ルールのストーリーポイントの作業レベルを見積もることは複雑です。以下は、ルールに必要な作業レベルを見積もる際に MTA が使用する一般的なガイドラインです。

作業レベル	Story Points	説明
Information	0	移行の優先度が非常に低いか、優先度のない情報警告。
Trivial	1	移行は、些細な変更または単純なライブラリスワップであり、API の変更はないか、最小限となります。
Complex	3	移行タスクに必要な変更は複雑ですが、解決策が文書化されています。
Redesign	5	移行タスクでは、API が大幅に変更され、再設計または完全なライブラリの変更が必要になります。
Rearchitecture	7	移行には、コンポーネントまたはサブシステムの完全な再アーカイブが必要です。
Unknown	13	移行ソリューションは不明なため、完全な再書き込みが必要になる場合があります。

A.1.3. タスクカテゴリー

作業量レベルに加えて、移行タスクを分類してタスクの重大度を示すことができます。移行作業の優先順位付けに役立つ問題をグループ化するために、以下のカテゴリーが使用されます。

Mandatory

移行を成功させるには、タスクを完了する必要があります。変更が行われないと、生成されるアプリケーションはビルドまたは実行に成功しません。たとえば、ターゲットプラットフォームでサポートされないプロプライエタリー API の置き換え例が含まれます。

Optional

移行タスクが完了しない場合、アプリケーションは動作しますが、結果が最適になるとは限りません。移行時に変更が行われない場合は、移行の完了後すぐにスケジュールに配置することが推奨されます。これには、EJB 2.x コードの EJB 3 へのアップグレードが挙げられます。

Potential

移行プロセス中にタスクを調べる必要があります。しかし、移行を成功させるためにタスクが必須かどうかを判断するのに十分な詳細情報がありません。この例は、直接互換性のあるタイプがないサードパーティのプロプライエタリタイプの移行です。

Information

タスクは、特定のファイルの存在を通知するために含まれています。これらは、モダナイゼーション作業の一部として検証または変更する必要がある場合がありますが、通常は変更が必要ありません。これには、ロギング依存関係または Maven **pom.xml** があります。

タスクの分類に関する詳細は、[カスタムルールカテゴリーの使用](#) を参照してください。

A.2. 関連情報

A.2.1. 既存の MTA XML ルールの確認

MTA XML ベースのルールは、GitHub の <https://github.com/windup/windup-rulesets/tree/master/rules-reviewed> にあります。

ローカルマシンで MTA XML ルールをフォークし、クローンを作成できます。

ルールは、ターゲットプラットフォームおよび機能でグループ化されます。新しいルールを作成する場合は、必要なルールと似たルールを見つけ、これを開始テンプレートとして使用すると便利です。

新しいルールは継続的に追加されるため、頻繁に更新を確認することをお勧めします。

A.2.1.1. Migration Toolkit for Applications XML ルールのフォークおよびクローン作成

Migration Toolkit for Applications の **windup-rulesets** リポジトリは、カスタム Java ベースのルールアドオンおよび XML ルールを作成する方法に関する作業例を提供します。カスタムルールを作成する際の開始点として使用することができます。

git クライアントがマシンにインストールされていること。

1. [Migration Toolkit for Applications Rulesets](#) GitHub ページで **Fork** リンクをクリックし、自分用の Git にプロジェクトを作成します。fork によって作成されたフォークされた GitHub リポジトリの URL は **https://github.com/<YOUR_USER_NAME>/windup-rulesets.git** のようになります。
2. Migration Toolkit for Applications ルールセットリポジトリをローカルファイルシステムにクローンします。

```
$ git clone https://github.com/<YOUR_USER_NAME>/windup-rulesets.git
```

3. これにより、ローカルファイルシステムに **crushup-rulesets** ディレクトリが作成され、設定されます。新規作成されたディレクトリに移動します。以下に例を示します。

```
$ cd windup-rulesets/
```

4. 最新のコード更新を取得できるようにする場合は、リモートの **upstream** リポジトリを追加して、元のフォークしたリポジトリに変更を取得できるようにします。

```
$ git remote add upstream https://github.com/windup/windup-rulesets.git
```

5. **upstream** リポジトリから最新のファイルを取得します。

```
$ git fetch upstream
```

A.2.2. リソース

- MTA Javadoc: <http://windup.github.io/windup/docs/latest/javadoc>
- MTA フォーク: <https://developer.jboss.org/en/windup>
- MTA JIRA 問題トラッカー
 - コア MTA: <https://issues.jboss.org/browse/WINDUP>
 - MTA ルール: <https://issues.jboss.org/browse/WINDUPRULE>
- MTA メーリングリスト: jboss-migration-feedback@redhat.com
- MTA IRC チャンネル: Server FreeNode (irc.freenode.net)、チャンネル ([#windup](#) ([transcripts](#)))

改訂日時: 2022-12-03 17:27:14 +1000