Migration Toolkit for Runtimes 1.2

规则开发指南

创建自定义规则以增强迁移覆盖。

Last Updated: 2024-06-13

Migration Toolkit for Runtimes 1.2 规则开发指南

创建自定义规则以增强迁移覆盖。

法律通告

Copyright © 2024 Red Hat, Inc.

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.

摘要

本指南论述了如何为 Migration Toolkit for Runtimes 创建自定义 XML 规则。

目录

使开源包含更多 3

第 1 章 简介 4
1.1. 关于规则开发指南 4
1.2. MTR 规则 4

第 2 章 使用规则入门 5
2.1. 创建第一个 XML 规则 5
2.2. 查看 MIGRATION TOOLKIT FOR RUNTIMES 快速入门 10

第 3 章 创建 XML 规则 12
3.1. XML 规则结构 12
3.2. 创建基本 XML 规则 13
3.3. XML 规则语法 16
3.4. 为 MIGRATION TOOLKIT FOR RUNTIMES 添加一个规则 34

第 4 章 测试 XML 规则 35
4.1. 创建测试规则 35
4.2. 手动测试 XML 规则 40
4.3. 使用 JUNIT 测试规则 40
4.4. 关于验证报告 41

第 5 章 覆盖规则 44
5.1. 覆盖一个规则 44
5.2. 禁用已规则 45

第 6 章 使用自定义规则类别 46
6.1. 添加一个自定义类别 46
6.2. 为自定义类别分配规则 46

附录 A. 参考材料 48
A.1. 关于规则故事点 48
A.2. 其他资源 49

使开源包含更多

红帽致力于替换我们的代码、文档和 Web 属性中存在问题的语言。我们从这四个术语开始：master、slave、黑名单和白名单。由于此项工作十分艰巨，这些更改将在即将推出的几个发行版本中逐步实施。有关更多详情，请参阅我们的首席技术官 Chris Wright 提供的消息。

第 1 章 简介

1.1. 关于规则开发指南

本指南适用于希望为 Migration Toolkit for Runtimes (MTR)工具创建自定义基于 YAML 的规则的软件工程师。

详情请参阅 *Migration Toolkit for Runtimes 简介* 了解概述和 *CLI 指南*。

1.1.1. 本指南使用 <MTR_HOME>

本指南使用 **<MTR_HOME>** 可替换变量来指示您的 MTR 安装的路径。安装目录是您提取 MTR **.zip** 文件的 **mtr-1.2.6.GA-offline** 目录。

在本指南中遇到 **<MTR_HOME>** 时，将其替换为 MTR 安装的实际路径。

1.2. MTR 规则

Migration Toolkit for Runtimes (MTR)包含基于规则的迁移工具（分析器），可用于分析您计划迁移的应用程序使用的应用程序用户界面(API)、技术和架构。MTR 分析器规则使用以下规则模式：

```
when(condition)
 message(message)
 tag(tags)
```

您可以在内部使用 MTR 规则来执行以下任务：

- 从存档中提取文件.
- 解译文件.
- 扫描和分类文件类型。
- 分析 XML 和其他文件内容。
- 分析应用程序代码。
- 构建报告。

MTR 根据规则执行结果构建数据模型，并将组件数据和关系存储在图形数据库中。然后，可以根据迁移规则和报告目的的要求查询和更新此数据库。

注意

您可以创建自己的自定义分析器规则。您可以使用自定义规则来识别自定义库或可能未被提供的标准迁移规则涵盖的其他组件。

有关如何编写自定义规则的说明，请参阅*[规则开发指南]*。

第 2 章 使用规则入门

您可以通过创建规则或查看快速入门来开始创建自定义 MTR 规则。

2.1. 创建第一个 XML 规则

本小节介绍了创建和测试第一个基于 MTR XML 的规则的过程。这假设您已安装了 MTR。有关安装说明，请参阅 *CLI 指南*中的安装和运行 CLI。

在本例中，您将编写一个规则来发现应用程序定义了包含 **<class-loading>** 元素的 **jboss-web.xml** 文件，并提供描述如何迁移代码的文档的链接。

2.1.1. 为规则创建目录结构

创建目录结构以包含您的第一条规则和用于测试的数据文件。

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

此目录结构也将用于存放生成的 MTR 报告。

2.1.2. 创建数据来测试规则

1. 在 **/home/<USER_NAME>/migration-rules/data/** 子目录中创建一个 **jbossweb.xml** 文件。
2. 复制以下内容。

```
<!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>
```

2.1.3. 创建规则

MTR 基于 XML 的规则使用以下规则模式：

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

流程

1. 在 **/home/<USER_NAME>/migration-rules/rules/** 目录中，创建一个名为 **JBoss5-web-class-loading.windup.xml** 的文件，其中包含以下内容：

```
<?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>
```

注意

XML 文件名必须包含 **.windup.xml** 扩展。否则，MTR 不会评估新规则。

2. 为 ruleset 和 rule 添加唯一标识符：
   - 将 **<UNIQUE_RULESET_ID>** 替换为适当的规则集 ID，如 **JBoss5-web-class-loading**。
   - 将 **<UNIQUE_RULE_ID>** 替换为适当的规则 ID，例如 **JBoss5-web-class-loading_001**。
3. 添加以下 ruleset 附加组件依赖项：

```
<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. 添加源和目标技术：
   - 将 **<SOURCE_ID>** 替换为 **eap**。
   - 将 **<TARGET_ID>** 替换为 **eap**。
5. 设置源和目标技术版本。
   - 将 **<SOURCE_VERSION_RANGE>** 替换为 **(4,5)**。

- 将 **<TARGET_VERSION_RANGE>** 替换为 **(6,)**。

如需更多信息，请参阅 Apache Maven 版本范围规格。

6. 完成 **when** 条件。由于此规则测试在 XML 文件中的一个匹配项，因此 **xmlfile** 被用于评估文件。要在作为 **jboss-web** 的子 **class-loading** 项上匹配，请使用 xpath 表达式 **jboss-web/class-loading**。

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

7. 为这个规则完成 **perform** 操作。

- 添加带有描述性标题的分类，以及 **1** 的工作量级别。
- 提供提示以及描述迁移详情的文档的链接。

```
<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/zh-CN/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>
```

该规则现已完成，应类似以下示例。

```
<?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"/>
         <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/zh-
CN/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>
```

2.1.4. 安装规则

通过将规则放在适当的目录中，会安装 MTR 规则。

将 **JBoss5-web-class-loading.windup.xml** 复制到 **<MTR_HOME>/rules/** 目录。

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

2.1.5. 测试规则

打开一个终端，再运行以下命令，将测试文件作为输入参数和输出报告的目录传递。

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

您应看到以下结果：

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

2.1.6. 查看报告

查看报告以确保它提供了预期的结果。有关 MTR 报告的详细指导，请参阅 MTR *CLI 指南*中的审阅报告部分。

1. 在一个 web 浏览器打开 **/home/<USER_NAME>/migration-rules/reports/index.html**。
2. 验证规则是否已成功运行。
    a. 在主登录页面中，点 **Rule providers execution overview** 链接，以打开 Rule Providers

Execution Overview。

b. 找到 **JBoss5-web-class-loading_001** 规则，验证它的 **Status?** 是 **Condition met**，它的 **Result?** 是 **success**。

图 2.1. 测试规则执行

JBoss5-web-class-loading
Phase: MigrationRulesPhase

Rule-ID	Rule	Statistics	Status?	Result?	Failure Cause
JBoss5-web-class-loading_001	<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>	Vertices Created: 6 Edges Created: 11 Vertices Removed: 0 Edges Removed: 0	Condition met.	success	

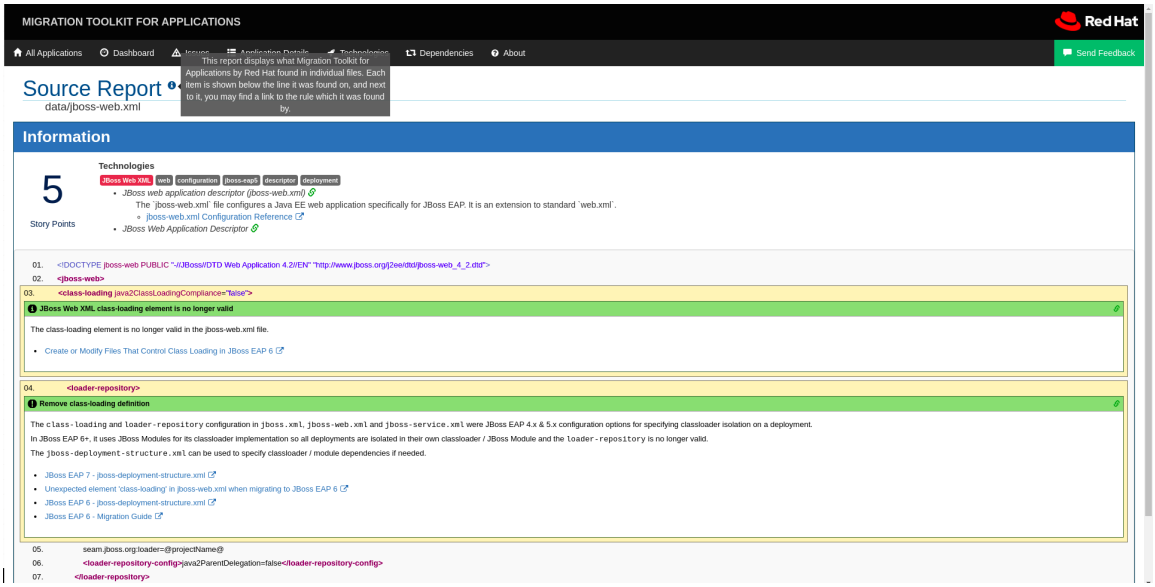
3. 验证规则是否与测试数据匹配：

a. 在主登录页面中，点应用程序或输入文件夹的名称，本例中为 **data**。

b. 点 **Application Details** 报告链接。

c. 点 **jboss-web.xml** 链接，以查看 **Source** 报告。
您可以看到 **<class-loading>** 行已被突出显示，自定义规则中的 hint 会显示内联。

图 2.2. 规则匹配



文件顶部列出了匹配规则的分类。您可以使用链接图标查看该规则的详情。请注意在本示例中，**jboss-web.xml** 文件匹配另外一个规则 (**JBoss web application descriptor (jboss-web.xml)**)，它生成的故事点为 **1**。此故事点和来自自定义规则的 **1** 个故事点将此文件的总故事点设置为 **2**。

2.2. 查看 MIGRATION TOOLKIT FOR RUNTIMES 快速入门

Migration Toolkit for Runtimes 快速入门提供了如何创建自定义基于 Java 的规则附加组件和 XML 规则的示例。您可以使用它们作为创建自己的自定义规则的起点。

每个快速入门都有一个 **README.adoc** 文件，其中包含该快速入门说明。

您可以下载快速入门最新版本的 **.zip** 文件。如果您希望使用源代码，您可以 fork 和 clone **windup-quickstarts** 项目存储库。

2.2.1. 下载最新的快速入门

您可以下载快速入门的最新版本。

流程

1. 使用浏览器进入 https://github.com/windup/windup-quickstarts/releases。
2. 点最新版本将 **.zip** 文件下载到本地文件系统。
3. 将存档文件提取到本地目录。
   您可以查看 quickstart **README.adoc** 文件。

2.2.2. 分叉并克隆快速入门 GitHub 项目

您可以在本地机器上分叉并克隆 Quickstart Github 项目。

先决条件

- 已安装 **git** 客户端。

流程

1. 在 Migration Toolkit for Runtimes quickstart GitHub 页面中，点 **Fork** 在您自己的 Git 中创建该项目。派生的 GitHub 存储库 URL 应如下所示：
   https://github.com/<YOUR_USER_NAME>/windup-quickstarts.git。
2. 将运行时快速入门库克隆到本地文件系统中：

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

   这会在本地文件系统中创建 **windup-quickstarts** 目录。

3. 进入新创建的目录：

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

4. 要检索最新的代码更新，添加远程的 **upstream** 仓库，以便您可以获取原始分叉仓库的变化：

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

5. 从 **upstream** 仓库下载最新的文件：

```
$ git fetch upstream
```

-

第 3 章 创建 XML 规则

3.1. XML 规则结构

这部分论述了 XML 规则的基本结构。所有 XML 规则定义为规则集中的元素。如需了解更多详细信息，请参阅 MTR XML 规则模式。

3.1.1. Rulesets（规则集）

规则集是一组以特定迁移领域为目标的一个或多个规则。这是 **<ruleset>** 元素的基本结构。

- **<ruleset id="<UNIQUE_RULESET_ID>">**: 将其定义为 MTR 规则集，并将其赋予唯一的规则 ID。
  - **<metadata>** :有关规则集的元数据。
    - **<description>** :规则集的描述。
    - **<dependencies/>** :这个规则集所需的规则附加组件。
    - **<sourceTechnology/>**: 源技术。
    - **<targetTechnology/>**: 目标技术。
    - **<overrideRules/>** : 设置为 **true** 表示此规则集中的规则使用通过 MTR 分发的核心规则集中的相同 ID 覆盖规则。ruleset id 和 rule id 必须与核心规则集中的规则匹配，否则规则将被忽略。另外，此规则集中的目标技术必须与您为运行分析指定的目标之一匹配。默认为 **false**。
  - **<rules>** : 一组单独的规则。
    - **<rule id="<UNIQUE_RULE_ID>">**: 定义规则并给它赋予一个唯一的 ID。建议将规则 ID 作为规则 ID 的一部分，如 **<UNIQUE_RULESET_ID_UNIQUE_RULE_ID>**。可以为规则集定义一个或多个规则。
      - **<when>** ：要匹配的条件。
      - **<perform>** : 当规则条件匹配时，要执行的操作。
      - **<otherwise>** ：当规则条件不匹配时，要执行的操作。这个元素采用与 **<perform>** 元素相同的子元素。
      - **<where>** ：字符串模式定义为参数，可在规则定义其他位置使用。
    - **<file-mapping/>** :映射到图形类型的扩展。
    - **<package-mapping/>** :映射软件包模式(regular 表达式)到机构名称。

3.1.2. 预定义的规则

MTR 为常见的迁移要求提供预定义规则。这些核心 MTR 规则位于 MTR 的安装中 **<MTR_HOME>/rules/migration-core/**。

以下是在专有实用程序类上匹配的核心 MTR 规则示例。

```
<?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" />
               <tag>weblogic</tag>
            </hint>
         </perform>
      </rule>
      ...
   </rules>
</ruleset>
```

3.2. 创建基本 XML 规则

这部分论述了如何创建 MTR XML 规则。这假设您已安装了 MTR。有关安装说明，请参阅 MTR *CLI 指南*。

3.2.1. 创建基本 XML 规则模板

MTR XML 规则由 *conditions* 和 *actions* 组成，并使用以下规则模式：

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

创建包含以下内容的文件，这是 XML 规则的基本语法：

重要

XML 文件名必须包含 **.windup.xml** 扩展。否则，MTR 不会评估新规则。

```
<?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. 创建规则集元数据

XML 规则集 **metadata** 元素提供有关规则集的额外信息，如描述、源和目标技术以及附加依赖项。元数据还允许指定标签，它们允许您提供有关规则集的额外信息。

<metadata> 示例

```
<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. 创建一个规则

单个规则包含在 **<rules>** 元素中。它们包含一个或多个 **when** 条件以及要执行的操作。

有关有效规则语法，请查看 XML 规则模式。

3.2.3.1. 创建一个 <when> 条件

XML 规则 **<when>** 元素用于测试条件。以下是一个有效 **<when>** 条件的列表。

元素	描述
<and>	标准的逻辑 *and* 运算符。
<filecontent>	在文件中查找字符串或文本，例如属性文件。
<file-mapping>	为内部存储文件模型定义文件名。
<javaclass>	测试 Java 类中的匹配项。
<javaclass-ignore>	排除要在处理发现时要忽略的 javaclass。
<not>	标准的逻辑 *not* 运算符。
<or>	标准的逻辑 *or* 运算符。
<package-mapping>	定义到组织或库的软件包名称。
<project>	测试项目特征，如依赖项。
<true>	始终匹配。
<xmlfile>	在 XML 文件中测试匹配项.

具体语法取决于您是创建规则来评估 Java 类、XML 文件、项目或文件内容。

3.2.3.2. 创建一个 <perform> 操作

XML 规则 **<perform>** 元素在满足条件时执行操作。此规则允许的操作包括应用资源分类，以及迁移步骤的线提示、迁移信息的链接和项目行报告。以下是一个有效 **<when>** 条件的列表。

元素	描述
数据分类	此操作会添加您要应用到整个文件的元数据。例如，如果 Java 类是 JMS Message Listener，您可以添加标题为"JMS Message Listener"的分类，其中包含适用于整个文件的信息。您还可以为整个文件设置工作级别。
<hint>	此操作会将元数据添加到文件中的一行中。这提供了有关如何迁移代码部分的提示或内联信息。
<iteration>	这将指定迭代规则中定义的隐式或显式变量。
<lineitem>	这提供了在应用程序概览页面中显示的高级消息。
<link>	这提供了一个 HTML 链接，用于附加有关迁移任务的信息或文档。
<xslt>	这指定了如何转换 XML 文件。

3.3. XML 规则语法

3.3.1. <when> 语法

在一个规则的 **when** 部分中允许的条件需要扩展 GraphOperation，当前包括对 Java 类、XML 文件、项目和文件内容的评估。因为在基于 Java 的规则附加组件后对 XML 规则建模，因此会提供到 JavaDocs 链接到相关 Java 类的链接，以便更好地了解它们的行为。

完整的 XML 规则模式位于：http://windup.jboss.org/schema/windup-jboss-ruleset.xsd.

以下小节描述了更多常见的 XML **when** 规则条件。

- <javaclass> 条件语法
- <xmlfile> 条件语法
- <project> 条件语法
- <filecontent> 条件语法
- <file> 条件语法
- <has-hint> 条件语法
- <has-classification> 条件语法
- <graph-query> 条件语法
- <dependency> 条件语法

默认情况下，如果提供了对于一个 **when** 规则条件时，则必须满足规则匹配的所有条件。

3.3.1.1. <javaclass> 语法

3.3.1.1.1. 概述

使用 **<javaclass>** 元素查找导入、方法、变量声明、注释、类实施和其他与 Java 类相关的项目。要更好地了解 **<javaclass>** 条件，请参阅 JavaClass 类的 JavaDoc。

以下是测试特定于 WebLogic 的 Apache XML 软件包的规则示例：

```
<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. 构造 <javaclass> 元素

3.3.1.1.2.1. <javaclass> 元素属性

属性名称	类型	描述
参考	CLASS_NAME	要匹配的软件包或类名称。可以使用的通配符。此属性是必需的。 **注意** 出于性能原因，在启动引用时不应该使用通配符。例如，使用 **weblogic.apache.xml.{*}** 而不是 **{web}.apache.xml.{*}**。 `references="weblogic.apache.xml.{*}"`
matchesSource	字符串	完全匹配的确切正则表达式。这可用于区分硬编码的字符串。此属性是必需的。 `matchesSource="log4j.logger"`
as	VARIABLE_NAME	分配到该规则的变量名称，以便它可以在以后的处理中用作参考。请参见以下的 **from** 属性。 `as="MyEjbRule"`

属性名称	类型	描述
from	VARIABLE_NAME	使用之前由其 **as** VARIABLE_NAME 识别的搜索结果开始搜索查询。 from="MyEjbRule"
in	PATH_FILTER	过滤与此正则表达式匹配的输入文件（正则表达式）命名模式。可以使用的通配符。 in="{*}File1"

3.3.1.1.2.2. <javaclass> 子元素

子元素	描述
<location>	在 Java 类中找到参考的位置。位置可以参考注解、字段和变量声明、导入和方法。有关有效值的完整列表，请查看 TypeReferenceLocation 的 JavaDoc 。 <location>IMPORT</location>
<annotation-literal>	匹配注解中的字面值。 以下示例匹配 **@MyAnnotation (myvalue="test")**。 <javaclass references="org.package.MyAnnotation"> <location>ANNOTATION</location> <annotation-literal name="myvalue" pattern="test"/> </javaclass> 请注意，在这种情况下，**<javaclass>** 代表注解(**@MyAnnotation**)，因此顶级注解过滤器 **<annotation-literal>** 必须指定 **name** 属性。如果 **<javaclass>** 引用了注解的类，则使用顶级注解过滤器将为 **<annotation-type>**。
<annotation-type>	匹配特定注解类型。您可以提供与注解元素匹配的子条件。 以下示例将匹配 **Calendar** 字段声明，并标上 **@MyAnnotation (myvalue="test")**。 <javaclass references="java.util.Calendar"> <location>FIELD_DECLARATION</location> <annotation-type pattern="org.package.MyAnnotation"> <annotation-literal name="myvalue" pattern="test"/> </annotation-type> </javaclass>

子元素	描述
<annotation-list>	匹配注解中数组中的项目。如果没有指定数组的索引，则只在条件应用到数组中的所有项时才匹配。您可以提供与这个元素匹配的子条件。 以下示例将匹配 **@MyAnnotation (mylist={"one","two"})**。 `<javaclass references="org.package.MyAnnotation" > <location>ANNOTATION</location> <annotation-list name="mylist"> <annotation-literal pattern="two"/> </annotation-list> </javaclass>` 请注意，在这种情况下，**<javaclass>** 代表注解(**@MyAnnotation**)，因此顶级注解过滤器 **<annotation-list>** 必须指定 **name** 属性。如果 **<javaclass>** 引用了注解的类，则使用顶级注解过滤器将为 **<annotation-type>**。

3.3.1.2. <xmlfile> 语法

3.3.1.2.1. 概述

使用 **<xmlfile>** 元素在 XML 文件中找到信息。要更好地了解 **<xmlfile>** 条件，请参阅 XmlFile 类的 JavaDoc。

以下是一个测试 XML 文件的示例：

```
<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. 构建一个 <xmlfile> 元素

3.3.1.2.2.1. <javaclass> 元素属性

属性名称	类型	描述

属性名称	类型	描述
matches	XPATH	在一个 XML 文件条件上匹配 matches="/w:web-app/w:resource-ref/w:res-auth[text() = 'Container']"
xpathResultMatch	XPATH_RESULT_STRING	返回匹配指定正则表达式的结果 <xmlfile matches="//foo/text()" xpathResultMatch="Text from foo."/>
as	VARIABLE_NAME	分配到该规则的变量名称，以便它可以在以后的处理中用作参考。请参见以下的 **from** 属性。 as="MyEjbRule"
in	PATH_FILTER	过滤与此正则表达式匹配的输入文件（正则表达式）命名模式。可以使用的通配符。 in="{*}File1"
from	VARIABLE_NAME	使用之前由其 **as** VARIABLE_NAME 识别的搜索结果开始搜索查询。 from="MyEjbRule"
public-id	PUBLIC_ID	DTD public-id 正则表达式。 public-id="public"

3.3.1.2.2.2. <xmlfile> matches 自定义函数

matches 属性可以使用几个内置自定义 XPath 功能，该功能可能具有有用的副作用，例如对规则变量堆栈设置匹配值。

功能	描述
windup:matches()	针对字符串匹配 XPath 表达式，可能包含 MTR 参数化占位符。 matches="windup:matches(//foo/@class, '{javaclassname}')" 这将匹配所有 **<foo/>** 元素和一个 **class** 属性，并将其值存储到每个迭代的 **javaclassname** 参数中。

3.3.1.2.2.3. <xmlfile> 子元素

子元素	描述
<namespace>	XML 文件中引用的命名空间。此元素包含两个可选属性：**prefix** 和 **uri**。 `<namespace prefix="abc" uri="http://maven.apache.org/POM/4.0.0"/>`

3.3.1.3. <project> 语法

3.3.1.3.1. 概述

使用 **<project>** 元素查询 Maven POM 文件以获取项目特征。要更好地了解 **<project>** 条件，请参阅 Project 类的 JavaDoc。

以下是一个规则的示例，该规则将检查 2.0.0.Final 和 2.2.0.Final 之间的 JUnit 依赖项版本。

```
<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. 构建 <project> 元素

3.3.1.3.2.1. <project> 元素属性

<project> 元素用于与项目的 Maven POM 文件匹配。您可以使用此条件来查询项目的依赖项。它本身没有任何属性。

3.3.1.3.2.2. <project> 子元素

子元素	描述
<artifact>	**<project>** 中使用的 Subcondition 来针对项目依赖项查询。**<artifact>** 元素属性如下所述。

3.3.1.3.2.3. <artifact> 元素属性

属性名称	类型	描述
groupId	PROJECT_GROUP_ID	匹配依赖项的项目 **<groupId>**。

属性名称	类型	描述
artifactId	PROJECT_ARTIFACT_ID	匹配依赖项的项目 **<artifactId>**。
fromVersion	FROM_VERSION	指定工件的低版本边界。例如 **2.0.0.Final**.
toVersion	TO_VERSION	指定工件的高边界。例如 **2.2.0.Final**。

可以验证 POM 文件中的元素，该文件包含规则正在搜索的工件。这可以通过使用可选的 **<location>** 元素来实现。以下示例显示了在 POM 文件的 **<plugins>** 元素中搜索工件的规则。

```
<rule id="location-custom-00001">
    <when>
        <project>
            <artifact groupId="org.springframework.boot" artifactId="spring-boot-maven-plugin">
                <location>PLUGINS</location>
            </artifact>
        </project>
    </when>
    <perform>
        <hint title="CUSTOM RULE - PLUGINS location" effort="1" category-id="potential">
            <message>Custom rule `location-custom-00001` has fired.</message>
            <link title="Migration Toolkit for Applications" href="https://developers.redhat.com/products/mta/getting-started"/>
        </hint>
    </perform>
</rule>
```

有效的位置列表如下：

- DEPENDENCY_MANAGEMENT
- DEPENDENCIES
- PLUGIN_MANAGEMENT
- PLUGINS
- PARENT

3.3.1.4. <filecontent> 语法

3.3.1.4.1. 概述

使用 **<filecontent>** 元素在文件中查找字符串或文本，例如： Properties 文件中的一行。要更好地了解 **<filecontent>** 条件，请参阅 FileContent 类的 JavaDoc。

3.3.1.4.2. 构建 <filecontent> 元素

3.3.1.4.2.1. <filecontent> 元素属性

属性名称	类型	描述
pattern	字符串	将文件内容与提供的参数化字符串匹配。此属性是必需的。

属性名称	类型	描述
filename	字符串	将文件名与提供的参数化字符串匹配。
as	VARIABLE_NAME	分配到该规则的变量名称，以便它可以在以后的处理中用作参考。请参见以下的 **from** 属性。 as="MyEjbRule"
from	VARIABLE_NAME	使用之前由其 **as** VARIABLE_NAME 识别的搜索结果开始搜索查询。 from="MyEjbRule"

3.3.1.5. <file> 语法

3.3.1.5.1. 概述

使用 **<file>** 元素查找具有特定名称的文件是否存在，例如： **ibm-webservices-ext.xmi** 文件。要更好地了解 **<file>** 条件，请参阅 File 类的 JavaDoc。

3.3.1.5.2. 构造 <file> 元素

3.3.1.5.2.1. <javaclass> 元素属性

属性名称	类型	描述
filename	字符串	将文件名与提供的参数化字符串匹配。此属性是必需的。
as	VARIABLE_NAME	分配到该规则的变量名称，以便它可以在以后的处理中用作参考。请参见以下的 **from** 属性。 as="MyEjbRule"
from	VARIABLE_NAME	使用之前由其 **as** VARIABLE_NAME 识别的搜索结果开始搜索查询。 *Example:* from="MyEjbRule"

3.3.1.6. <has-hint> 语法

3.3.1.6.1. 概述

使用 **<has-hint>** 元素测试某个文件或行是否已关联了 hint。它主要用于防止存在提示时触发，或者在没有其他条件时为默认执行实施规则。要更好地了解 **<has-hint>** 条件，请参阅 HasHint 类的 JavaDoc。

以下是一条规则，它检查是否存在用于 IBM JMS 目的地消息的提示，如果不包含它。

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

3.3.1.6.2. 构造 <has-hint>

<has-hint> 元素用于判断文件或行是否存在提示。它没有任何子元素。

3.3.1.6.2.1. <artifact> 元素属性

属性名称	类型	描述
message	字符串	可选参数允许您将提示与提供的消息字符串匹配。

3.3.1.7. <has-classification> 语法

3.3.1.7.1. 概述

使用 **<has-classification>** 元素测试文件或行是否具有分类。它主要用于防止 classification 已存在时触发，或者在没有其他条件时为默认执行实施规则。要更好地了解 **<has-classification>** 条件，请参阅 HasClassification 类的 JavaDoc。

3.3.1.7.2. 构造 <has-classification>

has-classification 元素用于判断是否存在指定的分类。它没有任何子元素。

3.3.1.7.2.1. <has-classification> 元素属性

属性名称	类型	描述
title	字符串	与分类匹配的可选标题。

3.3.1.8. <graph-query> 语法

3.3.1.8.1. 概述

使用 **<graph-query>** 元素搜索生成的任何元素图表。这个元素主要用于搜索特定的存档。要更好地了解 **<graph-query>** 条件，请参阅 QueryHandler 类的 JavaDoc。

以下是一个规则示例，用于测试来确定是否找到了任何 **ehcache** 软件包。

```
<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. 构造一个 <graph-query>

3.3.1.8.2.1. <artifact> 元素属性

属性名称	类型	描述
discriminator	MODEL_TYPE	用于搜索的模态类型这可以是任何有效的模型，但建议使用 **JarArchiveModel** 检查存档。此属性是必需的。
as	VARIABLE_NAME	分配到该规则的变量名称，以便它可以在以后的处理中用作参考。请参见以下的 **from** 属性。 as="MyEjbRule"
from	VARIABLE_NAME	使用之前由其 **as** VARIABLE_NAME 识别的搜索结果开始搜索查询。 from="MyEjbRule"

3.3.1.8.2.2. <graph-query> 属性

属性名称	类型	描述
name	字符串	在所选模型中要匹配的属性名称。当使用任何基于文件的模型时，建议在 **fileName** 上匹配。此属性是必需的。
type	property-type	定义预期的属性类型，可以是 **STRING** 或 **BOOLEAN**。
searchType	property-search-type	定义如何匹配条件。如果设置为 **equals**，则必须进行完全匹配。如果使用 **regex**，则使用正则表达式进行匹配。

3.3.1.9. <dependency> 语法

3.3.1.9.1. 概述

使用 **<dependency>** 元素搜索在应用程序的 POM 文件中定义的依赖项，以确定目标运行时是否支持它们。

以下是一个规则，它可检查属于 **org.springframework.boot** 组的所有工件（最多为 1.6.0）的规则。

```
<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> 语法

在规则的 **perform** 部分可用的操作包括应用资源分类，以及迁移步骤的线提示、迁移信息的链接和项目行报告。因为在基于 Java 的规则附加组件后对 XML 规则建模，因此会提供到 JavaDocs 链接到相关 Java 类的链接，以便更好地了解它们的行为。

您可以查看 完整的 XML 规则模式。

以下小节描述了比较常见的 XML 规则执行操作。

3.3.2.1. <classification> 语法

3.3.2.1.1. 概述

<classification> 元素用于识别或分类与规则匹配的应用程序资源。它提供报告中显示的标题、工作量级别，还可提供有关如何迁移此资源分类的其他信息的链接。要更好地了解 **<classification>** 条件，请参阅 Classification 类的 JavaDoc。

以下是将资源分类为 WebLogic EAR 应用部署描述符文件的规则示例。

```
<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. <has-classification> 元素属性

属性名称	类型	描述
title	字符串	给此资源的标题。此属性是必需的。 `title="JBoss Seam Components"`
effort	BYTE	分配给此资源的工作量程度。 `effort="2"`

属性名称	类型	描述
category-id	字符串	对 **MTR_HOME/rules/migration-core/core.windup.categories.xml** 中定义的类别的引用。默认类别为 **mandatory**, **optional**, **potential**, 和 **information**。 category-id="mandatory"
的	VARIABLE_NAME	为所给的引用创建新分类。 of="MySeamRule"

3.3.2.1.3. <classification> 子元素

子元素	描述
<link>	为其他信息提供链接 URI 和文本标题。 <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>
<tag>	为分类提供额外的自定义信息。 <tag>Seam3</tag>
<description>	此资源的描述。 <description>JBoss Seam components must be replaced</description>

3.3.2.2. <link> 语法

3.3.2.2.1. 概述

<link> 元素用于分类或提示，以提供信息内容的链接。要更好地了解 **<link>** 条件，请参阅 Link 类的 JavaDoc。

以下是创建其他信息链接的规则示例。

```
<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. <javaclass> 元素属性

属性名称	类型	描述
href	URI	所引用链接的 URI。 href="https://access.redhat.com/articles/1249423"
title	字符串	链接的标题。 title="Migrate WebLogic Proprietary Servlet Annotations"

3.3.2.3. <hint> 语法

3.3.2.3.1. 概述

<hint> 原始提供了有关如何迁移代码部分的提示或内联信息。要更好地了解 **<hint>** 条件，请参阅 Hint 类的 JavaDoc。

以下是一个测试 XML 文件的示例：

```
<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> 元素属性

属性名称	类型	描述
title	字符串	使用指定字符串替换此提示。此属性是必需的。 `title="JBoss Seam Component Hint"`
category-id	字符串	对 **MTR_HOME/rules/migration-core/core.windup.categories.xml** 中定义的类别的引用。默认类别为 **mandatory**, **optional**, **potential**, 和 **information**。 `category-id="mandatory"`
in	VARIABLE_NAME	在由给定变量解析的 FileLocationModel 中创建一个新的 Hint。 `in="Foo"`
effort	BYTE	分配给此资源的工作量程度。 `effort="2"`

3.3.2.3.3. <hint> 子元素

子元素	描述
<message>	描述迁移提示的消息。 `<message>EJB 2.0 is deprecated</message>`
<link>	识别或分类信息内容的链接。 `<link href="http://docs.oracle.com/javaee/6/api/" title="Java Platform, Enterprise Edition 6 API Specification" />`
<tag>	为这个 **hint** 定义自定义标签。 `<tag>Needs review</tag>`

子元素	描述

<quickfix>	包含在满足规则条件时，MTR 插件使用的信息来执行快速修复。
	
	`<quickfix name="slink-qf" type="REPLACE">`
	`  <replacement>h:link</replacement>`
	`  <search>s:link</search>`
	`</quickfix>`

3.3.2.4. <xslt> 语法

3.3.2.4.1. 概述

<xslt> 元素指定了如何转换 XML 文件。要更好地了解 **<xslt>** 条件，请参阅 XSLTTransformation 类的 JavaDoc。

以下是定义 XSLT 操作的规则示例。

```
<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.xsl" extension="-jboss.xml"/>
    </iteration>
  </perform>
</rule>
```

3.3.2.4.2. <xslt> 元素属性

属性名称	类型	描述
title	字符串	设置报告中此 XSLTTransformation 的标题。此属性是必需的。 `title="XSLT Transformed Output"`
的	字符串	为给定参考创建新转换。 `of="testVariable_instance"`
extension	字符串	设置此 XSLTTransformation 的扩展。此属性是必需的。 `extension="-result.html"`

属性名称	类型	描述
模板	字符串	设置 XSL 模板。此属性是必需的。 `template="simpleXSLT.xsl"`
effort	BYTE	转型所需的工作量水平。

3.3.2.4.3. <xslt> 子元素

子元素	描述
<xslt-parameter>	指定 XSLTTransformation 参数作为属性值对 `<xslt-parameter property="title" value="EJB Transformation"/>`

3.3.2.5. <lineitem> 语法

3.3.2.5.1. 概述

<lineitem> 元素对应用程序提供常规的迁移要求，例如需要替换弃用的库，或需要解决潜在的类加载问题。此信息显示在项目或应用程序概览页面中。要更好地了解 **<lineitem>** 条件，请参阅 LineItem 类的 JavaDoc。

以下是创建 lineitem 消息的规则示例。

```
<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> 元素属性

属性名称	类型	描述

属性名称	类型	描述
message	字符串	一个 lineitem 信息。 message="Proprietary code found."

3.3.2.6. <iteration> 语法

3.3.2.6.1. 概述

<iteration> 元素指定迭代规则中定义的隐式或显式变量。要更好地了解 **<iteration>** 条件，请参阅 Iteration 类的 JavaDoc。

以下是执行迭代的规则示例。

```
<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> 元素属性

属性名称	类型	描述
over	VARIABLE_NAME	迭代此 VARIABLE_NAME 标识的条件。 over="jboss-app"

3.3.2.6.3. <iteration> 子元素

子元素	描述

子元素	描述
<iteration>	子元素包括 **when** 条件，以及操作 **iteration**, **classification**, **hint**, **xslt**, **lineitem**, 和 **otherwise**。

3.3.3. <where> 语法

您可以定义用于指定在 XML 规则的其他元素中使用的匹配模式的参数。这有助于简化复杂匹配表达式的模式。

使用 **<where>** 元素来定义参数。使用 **param** 属性指定参数名，使用 **<matches>** 元素提供特征。然后可以使用 **{<PARAM_NAME>}** 语法在规则定义中引用此参数。

您可以查看 完整的 XML 规则模式。

以下的示例规则定义了一个名为 **subpackage** 的参数，它指定 **(activeio|activemq)** 的特征。

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

由 **subpackage** 定义的特征然后会在 **<javaclass> references** 属性中别替换。这会导致规则与 **org.apache.activeio.*** 和 **org.apache.activemq.*** 软件包匹配。

3.4. 为 MIGRATION TOOLKIT FOR RUNTIMES 添加一个规则

通过将规则复制到适当的 MTR 文件夹，安装 Migration Toolkit for Runtimes 规则。MTR 扫描规则，它们是位于以下位置的带有 **.windup.xml** 扩展的文件：

- 在 MTR 命令行上由 **--userRulesDirectory** 参数指定的目录。
- **<MTR_HOME>/rules/** 目录。**<MTR_HOME>** 是安装和运行运行时可执行文件的目录。
- **${user.home}/.mtr/rules/** 目录。该目录由 MTR 首次运行创建。它包含规则、附加组件和 MTR 日志。

注意

在 Windows 操作系统中，规则位于 **\Documents and Settings\<USER_NAME>\.mtr\rules** 或 **\Users\<USER_NAME>\.mtr\rules**。

第 4 章 测试 XML 规则

创建 XML 规则后，您应该创建一个测试规则以确保它可以正常工作。

4.1. 创建测试规则

使用与创建测试规则的过程类似的进程创建一个 XML 测试，但有以下区别：

- 测试规则应当放在要测试的规则下的 **test/** 目录中。
- 测试类等任何数据都应放在 **tests/** 目录下的 **data/** 目录中。
- 测试规则应使用 **.windup.test.xml** 扩展。
- 这些规则使用 Test XML 规则结构中定义的结构。

另外，建议您创建一个遵循它测试的规则名称的测试规则。例如，如果创建了一个规则，其文件名为 **proprietary-rule.mtr.xml**，则测试规则应称为 **proprietary-rule.windup.test.xml**。

4.1.1. 测试 XML 规则结构

所有测试 XML 规则都定义为包含一个或多个 **rulesets** 的 **rulesets** 规则集中的一个元素。如需了解更多详细信息，请参阅 MTR XML 规则模式。

规则测试是针对特定迁移区域的一个或多个测试组。这是 **<ruletest>** 元素的基本结构。

- **<ruletest id="<RULE_TOPIC>-test">**: 定义它作为唯一 MTR ruletest，并将其指定为唯一的 ruletest id。
  - **<testDataPath>**：定义用于测试的所有数据的路径，如类或文件。
  - **<sourceMode>** : 指定传递的数据是否只包括源文件。如果存档（如 EAR、WAR 或 JAR）正在使用，则这应设置为 **false**。默认值为 **true**。
  - **<rulePath>** : 要测试的规则的路径。这应该以要测试的规则的名称结尾。
  - **<ruleset>**: Rulesets 包括测试的逻辑。它们与 Rulesets 中定义的相同。

4.1.2. XML 规则语法

除了标准 XML 规则语法中的标签外，**when** 条件通常用于创建测试规则时：

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

除了标准 **perform action** 语法中的标签外，以下 **when** 条件通常用于测试规则中的操作：

- **<fail>**

4.1.2.1. <not> 语法

概述
<not> 元素是标准逻辑 *not* 操作符，如果不满足条件，通常用于执行 **<fail>**。

以下是测试规则的一个示例，如果分析末尾只有特定消息，则会失败。

```
<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.windup.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> 元素没有唯一属性或子元素。

4.1.2.2. <iterable-filter> 语法

概述
<iterable-filter> 元素统计验证条件的次数。详情请参阅 IterableFilter 类。

下例中查找指定消息的四个实例：

```
<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.mtr.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> 元素没有唯一的子元素。

<iterable-filter> 元素属性

属性名称	类型	描述
size	整数	验证的次数。

4.1.2.3. <classification-exists> 语法

<classification-exists> 元素决定分析中是否包含特定的分类标题。如需更多信息，请参阅 ClassificationExists 类。

重要

当测试包含特殊字符（如 **[** 或 **'** ）的消息时，您必须用反斜杠(\)进行转义来正确匹配。

以下是搜索特定分类标题的示例。

```
<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.mtr.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>
```

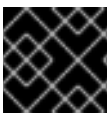
<classification-exists> 没有唯一的子元素。

<has-classification> 元素属性

属性名称	类型	描述
classification	字符串	要搜索的 **<classification> title**。
in	字符串	可选参数，限制对包含所定义文件名的文件的匹配。

4.1.2.4. <hint-exists> 语法

<hint-exists> 元素决定分析中是否包含特定的提示。它搜索已定义消息的任何实例，通常用于搜索 **<message>** 元素的开头或特定类。详情请参阅 HintExists 类。



重要

当测试包含特殊字符（如 **[** 或 **'**）的消息时，您必须用反斜杠(\)进行转义来正确匹配。

以下是搜索特定提示的示例。

```
<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
.*javax\.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> 元素没有唯一的子元素。

<hint-exists> 元素属性

属性名称	类型	描述
message	字符串	要搜索的 **<hint> message**。
in	字符串	可选参数限制与引用所给文件名的 **InLineHintModels** 匹配。

4.1.2.5. <fail> 语法

<fail> 元素将执行报告为失败，并显示相关的消息。通常与 **<not>** 条件一起使用，仅在不满足条件时才显示消息。

<fail> 元素没有唯一的子元素。

<fail> 元素属性

属性名称	类型	描述
message	字符串	要显示的消息。

4.2. 手动测试 XML 规则

您可以针对应用程序文件运行 XML 规则来测试它：

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

您应看到以下结果：

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

有关如何运行 MTR 的更多信息，请参见 Migration Toolkit for Runtimes *CLI 指南*。

4.3. 使用 JUNIT 测试规则

创建了测试规则后，它可以作为 JUnit 测试的一部分进行分析，以确认该规则是否符合执行的所有条件。MTR 规则存储库中的 **WindupRulesMultipleTests** 类旨在同时测试多个规则，并根据任何缺失的要求提供反馈。

先决条件

- 分叉并克隆 MTR XML 规则。此仓库的位置被称为 <RULESETS_REPO>。
- 创建一个测试文件。

创建 JUnit 测试配置

以下说明详细介绍了使用 Eclipse 创建 JUnit 测试。当使用其他 IDE 时，建议您参考 IDE 文档了解创建 JUnit 测试的说明。

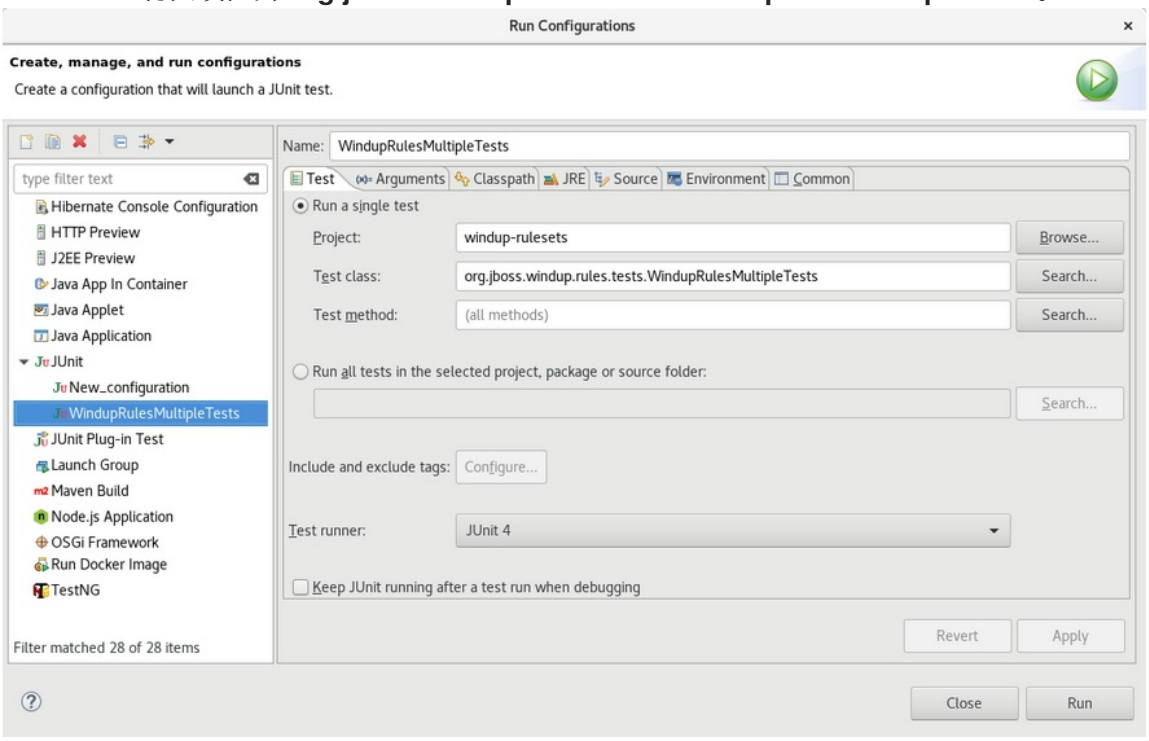
1. 将 MTR 规则集存储库导入到 IDE 中。
2. 将自定义规则以及相应的测试和数据复制到 **</path/to/RULESETS_REPO>/rules-reviewed/<RULE_NAME>/** 中。这应当创建以下目录结构：

   目录结构

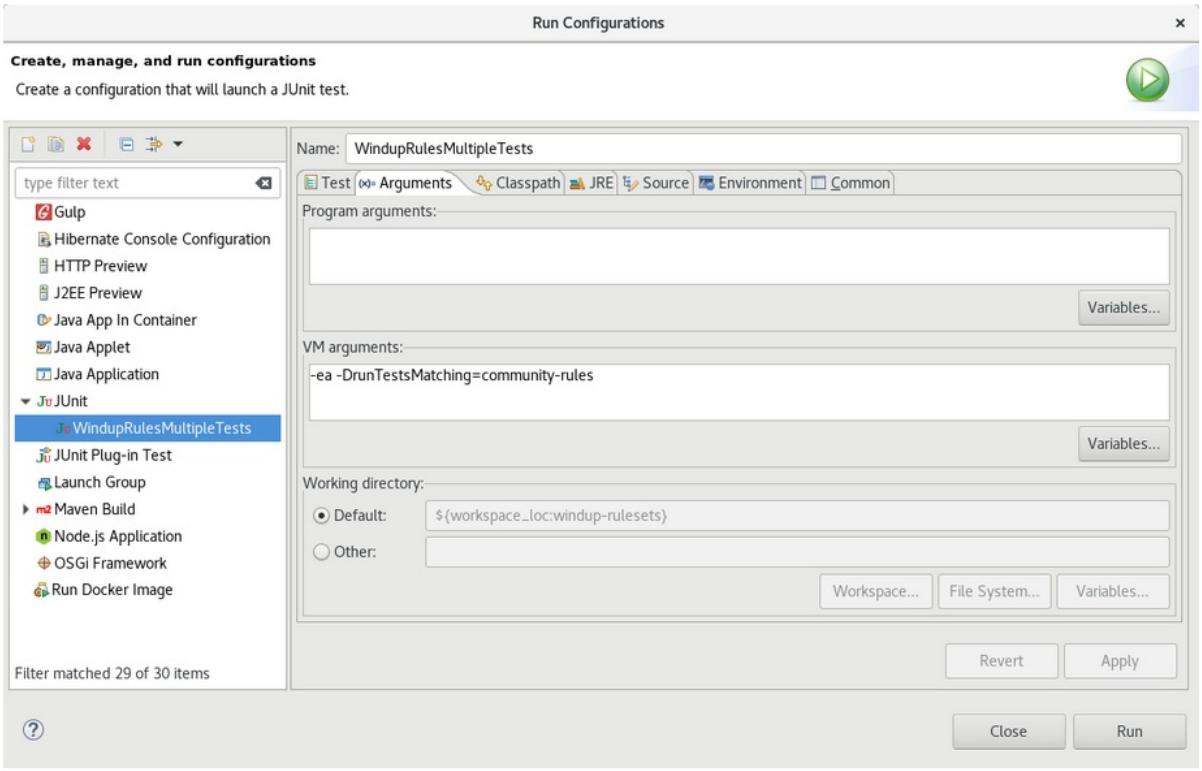
   ```
   ├── *rules-reviewed/*  _(Root directory of the rules found within the project)_
   │   ├── *<RULE_NAME>/*  _(Directory to contain the newly developed rule and tests)_
   │   │   ├── *<RULE_NAME>.windup.xml*  _(Custom rule)_
   │   │   ├── *tests/*  _(Directory that contains any test rules and data)_
   │   │   │   ├── *<RULE_NAME>.windup.test.xml* _(Test rule)_
   │   │   │   └── *data/*  _(Optional directory to contain test rule data)_
   ```

3. 从顶部菜单栏中选择 **Run**。
4. 从出现的下拉列表中选择 **Run Configuration...**。
5. 在左侧的选项中，右键单击 **JUnit**，再选择 **New**。
6. 使用以下命令：
   - **Name** ：JUnit 测试的名称，如 **WindupRulesMultipleTests**。

- Project: 确保将其设置为 **windup-rulesets**。
- Test class: 将其设置为 **org.jboss.windup.rules.tests.WindupRulesMultipleTests**。



7. 选择 **Arguments** 选项卡，并添加 **-DrunTestsMatching=<RULE_NAME>** VM 参数。例如，如果您的规则名称是 **community-rules**，则您可以添加 **-DrunTestsMatching=community-rules**，如以下镜像中所示。



8. 点右下角的 **Run** 以开始测试。
执行完成后，结果就可以进行分析。如果所有测试都通过，则正确格式化测试规则。如果所有测试都未通过，建议解决测试失败中出现的每个问题。

4.4. 关于验证报告

验证报告提供有关测试规则和失败的详细信息，并包含以下部分：

- **概述**
  本节包含测试运行总数并报告错误和失败的数量。它显示总成功率以及生成报告的时间（以秒为单位）。

- **软件包列表**
  本节包含为每个软件包执行的测试数量，并报告错误和失败的数量。它显示成功率以及要分析的每个软件包的时间（以秒为单位）。

  此时会显示一个名为 **org.jboss.windup.rules.tests** 的软件包，除非定义了额外的测试案例。

- **测试问题单**
  本节描述了测试问题单。每个失败都包含一个 Details 部分，可用于显示断言堆栈 trace，包括可表示错误源的人类可读行。

4.4.1. 创建验证报告

您可以为自定义规则创建验证报告。

先决条件

- 您必须分叉并克隆 MTR XML 规则。
- 您必须具有一个或多个测试 XML 规则才能进行验证。

流程

1. 进入到本地 **windup-rulesets** 存储库。
2. 为您的自定义规则和测试创建一个目录：**windup-rulesets/rules-reviewed/myTests**。
3. 将自定义规则和测试复制到 **windup-rulesets/rules-reviewed/<myTests>** 目录中。
4. 从 **windup-rulesets** 存储库的根目录中运行以下命令：

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

(1) 指定包含自定义规则和测试的目录。如果省略 **-DrunTestsMatching** 参数，验证报告将包含所有测试，并且生成需要更长的时间。

(2) 指定您的报告名称。

验证报告在 **windup-rulesets/target/site/** 仓库中创建。

4.4.2. 验证报告错误消息

验证报告包含运行规则和测试时遇到的错误。

下表包含错误消息以及如何解决错误。

表 4.1. 验证报告错误消息

错误消息	描述	解决方案
没有与规则匹配的测试文件	当规则文件没有对应的测试文件时，会发生此错误。	为现有规则创建测试文件。
测试规则 Ids <RULE_NAME> 没有找到!	当规则存在但没有对应的 ruletest 时，会抛出此错误。	为现有规则创建测试。
XML 解析在文件 <FILE_NAME> 上失败	XML 文件中的语法无效，无法通过规则验证器成功解析。	更正无效的语法。
未找到来自 **<testDataPath>** 标签的测试文件路径。预期的测试文件的路径为：<RULE_DATA_PATH>	测试规则中 **<testDataPath>** 标签中定义的路径中没有找到任何文件。	创建 **<testDataPath>** 标签中定义的路径，确保所有所需的数据文件都位于此目录中。
未执行带有 id="<RULE_ID>" 的规则。	在此验证过程中没有执行具有提供的 id 的规则。	确保存在与指定规则中定义的条件匹配的测试数据文件。

第 5 章 覆盖规则

您可以使用 MTR 或者自定义规则覆盖分发的核心规则。例如，您可以更改规则的匹配条件、工作或提示文本。这可以通过复制原始规则，将其标记为规则覆盖，并进行必要的调整。

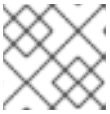
您可以通过创建一个带有空 <rule> 元素的 **<rule>** 覆盖来禁用规则。

5.1. 覆盖一个规则

您可以覆盖内核或自定义规则。

流程

1. 复制包含您要覆盖的规则的 XML 文件到自定义规则目录。
   自定义规则可以放置在 **<MTR_HOME>/rules**, **${user.home}/.mtr/rules/** 中，或由 **--userRulesDirectory** 命令行参数指定的目录。
2. 编辑 XML 文件，使其只包含您要覆盖的规则的 **<rule>** 元素。



注意

新规则集中未被新规则集覆盖的规则会正常运行。

3. 确保保留相同的规则和规则集 ID。当您复制原始规则 XML 时，这将确保 ID 相匹配。
4. 确保覆盖规则集中的目标技术与您为运行分析指定的目标之一匹配。
5. 将 **<overrideRules>true</overrideRules>** 元素添加到 ruleset 元数据。
6. 更新规则定义。
   您可以更改规则定义中的任何内容。新规则覆盖其整个原始规则。

以下规则覆盖示例将 **weblogic** 规则集中的 **weblogic-02000** 规则 **的工作量** 从 **1** 改为 **3**：

规则**覆盖定义示例**

```
<?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"> 1
    <metadata>
        ...
        <overrideRules>true</overrideRules> 2
    </metadata>
    <rules>
        <rule id="weblogic-02000" xmlns="http://windup.jboss.org/schema/jboss-ruleset"> 3
            <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. 确保 **ruleset id** 与原始 **ruleset id** 匹配。
2. 将 **<overrideRules>true</overrideRules>** 添加到 **<metadata>** 部分。
3. 确保 **rule id** 匹配原始的 **rule id**。
4. 更新的 **effort**。

当您运行 MTR 时，该规则会用相同的规则 ID 覆盖原始规则。您可以通过查看 Rule Provider Executions Overview 的内容来验证已使用的新规则。

5.2. 禁用已规则

要禁用规则，请按照以下示例创建带有空 **<rule>** 元素的规则覆盖定义：

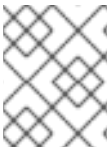
规则**覆盖定义示例来禁用一个**规则

```
<?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>** 原始为空，因此 **weblogic** ruleset 中的 **weblogic-02000** 规则被禁用。

第 6 章 使用自定义规则类别

您可以创建自定义规则类别，并为它们分配 MTR 规则。



注意

尽管 MTR 使用旧的 **severity** 字段处理规则，但您必须更新自定义规则以使用新的 **category-id** 字段。

6.1. 添加一个自定义类别

您可以将自定义类别添加到规则类别文件中。

流程

1. 编辑位于 **<MTR_HOME>/rules/migration-core/core.windup.categories.xml** 的类别文件。
2. 添加新的 **<category>** 元素并填写以下参数：
   - **id** ：用于引用类别的 MTR 规则的 ID。
   - **priority** ：相对于其他类别的排序优先级。首先会显示具有最低值的类别。
   - **name** ：类别的显示名称。
   - **description**: 类别的描述。

     自定义规则类别示例

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

     此类别可供 MTR 规则引用。

6.2. 为自定义类别分配规则

您可以为新的自定义类别分配一个规则。

流程

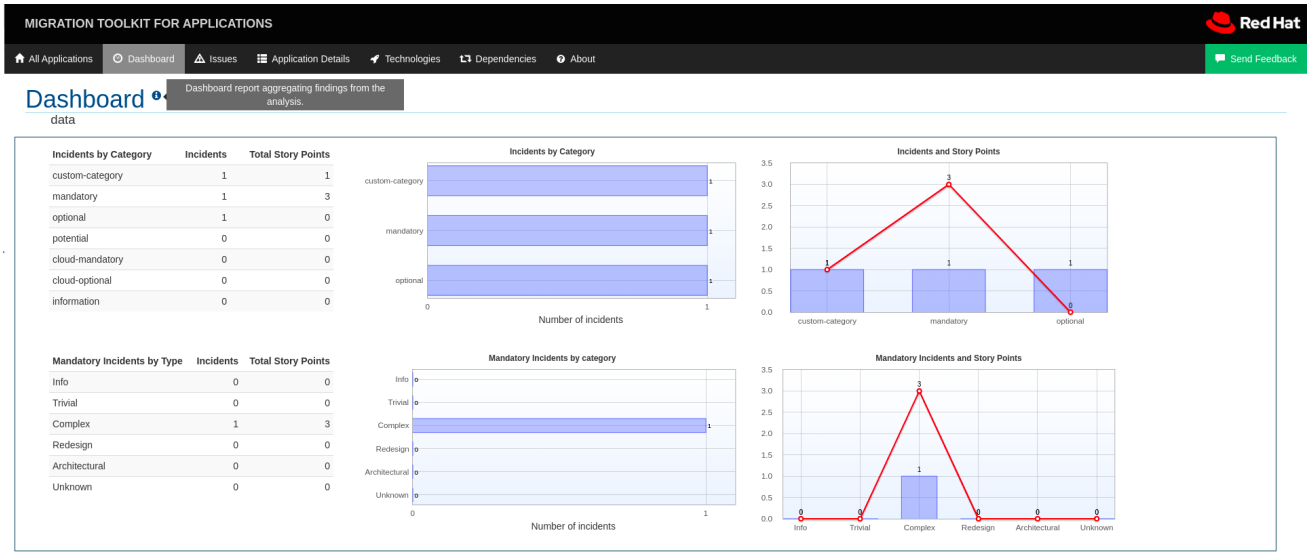
在 MTR 规则中，按如下所示更新 **category-id** 字段。

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

```
            <message>Hint message.</message>
        </hint>
    </perform>
</rule>
```

如果满足此规则条件，则此规则识别的事件会用到您的自定义类别。自定义类别显示在仪表板上，并在问题报告中显示。

图 6.1. 仪表板上的自定义类别



附录 A. 参考材料

A.1. 关于规则故事点

A.1.1. 什么是故事点？

*故事点*是敏捷软件开发中常用的抽象指标，用于估算实施功能或更改所需的*工作量水平*。

应用的 Migration Toolkit for Runtimes 使用故事点来代表迁移特定应用程序构造所需的工作程度，以及整个应用程序。它不一定转换为"人-小时"，但该值在不同的任务间应保持一致。

A.1.2. 故事点如何在规则中估计

估算一个规则的故事点的工作量水平可能很棘手。以下是估算规则所需工作量时的一般准则 MTR。

努力级别	故事点	描述
信息	0	迁移过程中具有非常低或没有优先级的信息。
微小	1	迁移是一个微小的变化或一个简单的库交换，没有或有最小的 API 更改。
复杂	3	迁移任务所需的更改比较复杂，但有一个已包括在文档中的解决方案。
重新设计	5	迁移任务需要重新设计或完整的库更改，并有显著的 API 更改。
架构重组	7	迁移会需要组件或子系统的完全的架构重组。
Unknown	13	迁移解决方案并非已知的，可能需要进行彻底的重写。

A.1.3. 任务类别

除了工作程度外，您还可以对迁移任务进行分类，以指明任务的严重性。下列类别用于对问题进行分组，以帮助确定迁移的工作量。

Mandatory（必需）

必须成功完成该任务才能成功迁移。如果没有进行任何更改，则生成的应用不会成功构建或运行。例如，替换在目标平台中不支持的专有 API。

选填

如果没有完成迁移任务，应用程序应该可以正常工作，但结果可能不是最佳。如果迁移时没有进行任何更改，建议在迁移完成后尽快按计划设置。

Potential

应在迁移过程中检查该任务，但没有足够的详细信息来确定任务是否成功完成。当没有直接兼容类型时，这将迁移第三方专有类型。

信息

该任务会包括告知您存在某些文件。可能需要将它们检查或修改为现代化工作的一部分，但通常不需要进行更改。

有关分类任务的更多信息，请参阅使用自定义规则类别。

A.2. 其他资源

A.2.1. 查看现有的 MTR XML 规则

基于 MTR XML 的规则位于 GitHub 上，其位置为：https://github.com/windup/windup-rulesets/tree/master/rules/rules-reviewed。

您可以在本地机器上分叉并克隆 MTR XML 规则。

规则按目标平台和功能分组。当您创建新规则时，找到与所需规则类似的规则，并将它用作入门模板。

新规则不断添加，因此最好经常检查更新。

A.2.1.1. 分叉并克隆运行时 XML 规则的 Migration Toolkit

Migration Toolkit for Runtimes **windup-rulesets** 存储库提供如何创建基于 Java 的规则附加组件和 XML 规则的工作示例。您可以使用它们作为创建自己的自定义规则的起点。

必须在您的机器上安装了 **git** 客户端。

1. 点 Migration Toolkit for Runtimes Rulesets GitHub 页面上的 **Fork** 链接，在您自己的 Git 中创建项目。fork 创建的 fork GitHub 存储库 URL 应该类似如下：**https://github.com/<YOUR_USER_NAME>/windup-rulesets.git**。
2. 将 Migration Toolkit for Runtimes rulesets 存储库克隆到本地文件系统：

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

3. 这会在本地文件系统上创建并填充 **windup-rulesets** 目录。导航到新创建的目录，例如

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

4. 如果要能够检索最新的代码更新，请添加远程 **upstream** 仓库，以便您可以获取原始分叉的存储库的任何更改。

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

5. 从 **upstream** 仓库获取最新的文件。

```
$ git fetch upstream
```

A.2.2. 其他资源

- MTR Javadoc: http://windup.github.io/windup/docs/latest/javadoc
- MTR Jira issue tracker: https://issues.redhat.com/projects/WINDUP
- MTR 邮件列表： windup-eng@redhat.com

更新于 2024-06-12