JBoss 企业级 BRMS 平台 5
BRMS 用户指南

适用于 JBoss Rules 开发人员、规则作者和商业分析员。
版 5.2.0

Landmann

JBoss 企业级 BRMS 平台 5 BRMS 用户指南

适用于 JBoss Rules 开发人员、规则作者和商业分析员。
版 5.2.0

Landmann
rlandmann@redhat.com

摘要

使用 JBoss 企业级 BRMS 平台产品的指南。

目录

序言 **4**
1. 文档约定 4
1.1. 排版约定 4
1.2. 抬升式引用约定 5
1.3. 备注及警告 5
2. 获得帮助并提供反馈信息 6
2.1. 您需要帮助吗？ 6
2.2. 我们需要您的反馈！ 6

第 1 章 简介 **8**
1.1. 本版本的新特征 8
1.2. 什么是 BRMS？ 8
1.3. 什么时候应该使用 BRMS？ 8
1.4. 谁会使用 BRMS？ 8
1.5. 功能概述 9

第 2 章 架构 **10**
2.1. 可重用的组件 10
2.2. 版本和存储 10

第 3 章 快速起步指南 **12**
3.1. 系统支持的浏览器 12
3.2. BRMS 还是 Guvnor? 12
3.3. 初始的配置 13
3.4. 编写规则 13
3.5. 搜索 13
3.6. 部署 13

第 4 章 BRMS 概念 **15**
4.1. 规则是资产 15
4.2. 分类 15
4.3. 规则授权 16
4.3.1. 资产编辑器（Asset Editor） 16
4.3.2. 用 Guided Editor 编辑商业规则 17
4.3.3. 规则的分解 17
4.3.4. Saliance 20
4.3.5. 用户驱动的下拉列表 20
4.3.6. 增加 DSL 语句 20
4.3.7. DSL 规则 21
4.3.8. 电子表格决策表 21
4.3.9. Guided 决策表（基于 Web） 22
4.3.9.1. 主要的组件 22
4.3.9.2. 列配置 22
4.3.9.2.1. 工具列 23
4.3.9.2.2. 属性列 23
4.3.9.2.3. 元数据列（Meta-data column） 23
4.3.9.2.4. 条件列（Condition Column） 23
4.3.9.2.5. 行为列（Action Column） 23
4.3.9.3. 规则定义 24
4.3.9.4. 单元合并 24
4.3.9.5. 单元分组 24
4.3.9.6. Otherwise 操作 25
4.3.10. 规则模板 25

4.3.10.1. 定义模板数据 28
4.3.10.1.1. 单元合并 29
4.3.10.1.2. 单元分组 30
4.3.10.2. 生成的 DRL 31
4.3.11. 规则流 32
4.3.12. 技术性规则（DRL） 32
4.3.13. 功能 33
4.3.14. 数据枚举（下拉列表配置） 33
4.3.15. 高级的枚举概念 33
4.4. 状态管理 34
4.5. 软件包管理 35
4.5.1. 导入 drl 软件包 37
4.6. 版本管理 38
4.7. 部署管理 38
4.8. 浏览库和查找规则 39

第 5 章 事实模型（对象模型） **41**
5.1. 全局区域（Global Area） 41
5.2. JAR 模型 41
5.3. 声明式模型（Declarative Model） 43
5.3.1. 在 Java 里消费声明式模型 44

第 6 章 工作集（Working Set） **46**
6.1. 检验 Field 约束 47

第 7 章 商业用户的视角 **49**

第 8 章 集成规则和应用程序 **50**
8.1. 知识代理（Knowledge Agent） 50
8.2. 手动部署 51
8.3. WebDAV 51
8.3.1. WebDav 和特殊字符 52
8.4. URL 52

第 9 章 收件箱和评论 **53**
9.1. 评论（Comment） 53
9.2. 收件箱（Inbox） 53

第 10 章 JBoss Developer Studio 集成 **55**
10.1. 功能概述 55
10.2. Guvnor 连接向导 56
10.3. Guvnor 库浏览器 57
10.4. Guvnor 文件的本地拷贝 60
10.5. 用于本地 Guvnor 资源的行为 61
10.6. 导入 Guvnor 库资源 65
10.7. Guvnor 插件的首选项 67

第 11 章 规则软件包签名（Rule Package Signing）的客户端配置 **68**

修订历史记录 **70**

序言

1. 文档约定

本手册使用几个约定来突出某些用词和短语以及信息的某些片段。

在 PDF 版本以及纸版中，本手册使用在 Liberation 字体套件中选出的字体。如果您在您的系统中安装了 Liberation 字体套件，它还可用于 HTML 版本。如果没有安装，则会显示可替换的类似字体。请注意：红帽企业 Linux 5 以及其后的版本默认包含 Liberation 字体套件。

1.1. 排版约定

我们使用四种排版约定突出特定用词和短语。这些约定及其使用环境如下。

单行粗体

用来突出系统输入，其中包括 shell 命令、文件名以及路径。还可用来突出按键以及组合键。例如：

> 要看到文件您当前工作目录中文件 **my_next_bestselling_novel** 的内容，请在 shell 提示符后输入 **cat my_next_bestselling_novel** 命令并按 **Enter** 键执行该命令。

以上内容包括一个文件名，一个 shell 命令以及一个按键，它们都以固定粗体形式出现，且全部与上下文有所区别。

按键组合与单独按键之间的区别是按键组合是使用加号将各个按键连在一起。例如：

> 按 **Enter** 执行该命令。
>
> 按 **Ctrl+Alt+F2** 切换到虚拟终端。

第一个示例突出的是要按的特定按键。第二个示例突出了按键组合：一组要同时按下的三个按键。

如果讨论的是源码、等级名称、方法、功能、变量名称以及在段落中提到的返回的数值，那么都会以上述形式出现，即**固定粗体**。例如：

> 与文件相关的等级包括用于文件系统的 **filesystem**、用于文件的 **file** 以及用于目录的 **dir**。每个等级都有其自身相关的权限。

比例粗体

这是指在系统中遇到的文字或者短语，其中包括应用程序名称、对话框文本、标记的按钮、复选框以及单选按钮标签、菜单标题以及子菜单标题。例如：

> 在主菜单条中选择**「系统」** → **「首选项」** → **「鼠标」**启动 **鼠标首选项**。在**「按钮」**标签中点击**「惯用左手鼠标」** 复选框并点击 **关闭**切换到主鼠标按钮从左向右（让鼠标适合左手使用）。
>
> 要在 **gedit** 文件中插入特殊字符，请在主菜单栏中选择**「应用程序」** → **「附件」** → **「字符映射表」**。接下来选择从 **Character Map** 菜单中选择**Search** → **「查找.....」**，在**「搜索」**字段输入字符名称并点击**「下一个」**按钮。此时会在**「字符映射表」**中突出您搜索的字符。双击突出的字符将其放在**「要复制的文本」**字段中，然后点击**「复制」**按钮。现在返回您的文档，并选择 **gedit** 菜单中的**「编辑」** → **「粘贴」**。

以上文本包括应用程序名称、系统范围菜单名称及项目、应用程序特定菜单名称以及按钮和 GUI 界面中的文本，所有都以比例粗体出现并与上下文区别。

固定粗斜体 或者 ***比例粗斜体***

无论固定粗体或者比例粗体，附加的斜体表示是可替换或者变量文本。斜体表示那些不直接输入的文本或者那些根据环境改变的文本。例如：

> 要使用 ssh 连接到远程机器，请在 shell 提示符后输入 **ssh *username@domain.name***。如果远程机器是 **example.com** 且您在该其机器中的用户名为 john，请输入 **ssh john@example.com**。
>
> **mount -o remount *file-system*** 命令会重新挂载命名的文件系统。例如：要重新挂载 **/home** 文件系统，则命令为 **mount -o remount /home**。
>
> 要查看目前安装的软件包版本，请使用 **rpm -q *package*** 命令。它会返回以下结果：***package-version-release***。

请注意上述使用黑斜体的文字 -- username、domain.name、file-system、package、version 和 release。每个字都是一个站位符，可用作您执行命令时输入的文本，也可作为该系统显示的文本。

不考虑工作中显示标题的标准用法，斜体表示第一次使用某个新且重要的用语。例如：

> Publican 是一个 *DocBook* 发布系统。

1.2. 抬升式引用约定

终端输出和源代码列表要与周围文本明显分开。

将发送到终端的输出设定为 **Mono-spaced Roman** 并显示为：

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

源码列表也设为 **Mono-spaced Roman**，但添加下面突出的语法：

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

1.3. 备注及警告

最后，我们使用三种视觉形式来突出那些可能被忽视的信息。

注意

备注是对手头任务的提示、捷径或者备选的解决方法。忽略提示不会造成负面后果，但您可能会错过一个更省事的诀窍。

重要

重要框中的内容是那些容易错过的事情：配置更改只可用于当前会话，或者在应用更新前要重启的服务。忽略'重要'框中的内容不会造成数据丢失但可能会让您抓狂。

警告

警告是不应被忽略的。忽略警告信息很可能导致数据丢失。

2. 获得帮助并提供反馈信息

2.1. 您需要帮助吗？

如果您对本文档论述的步骤有疑问，请访问红帽客户门户网站 http://access.redhat.com。通过客户门户网站，您可以：

- 搜索或者浏览有关红帽产品技术支持文章的知识库。
- 向红帽全球支持服务（GSS）提交支持案例。
- 访问其它红帽文档。

红帽还托管了大量讨论红帽软件和技术的电子邮件列表。公开列表位于 https://www.redhat.com/mailman/listinfo。点击任意列表名称即可订阅该列表或者访问列表归档。

2.2. 我们需要您的反馈！

如果你在本指南里发现了印刷错误，或者你有改进该手册的建议，我们希望听到你的声音！请提交报告到 Bugzilla 并指明产品 **JBoss Enterprise BRMS Platform 5** 和组件 **doc-BRMS_User_Guide**。下面的链接是针对本产品 http://bugzilla.redhat.com/ 预先填写的错误报告。

请填写 Bugzilla 的 **Description** 字段。请尽量将问题具体化，这样我们就可以尽快改正。

```
Document URL:

Section Number and Name:

Describe the issue:

Suggestions for improvement:

Additional information:
```

请提供你的名字以获得关于报告该问题的 credit。

第 1 章 简介

1.1. 本版本的新特征

表 1.1. 本版本的新特征

功能	修改
第 4.3.9 节 “ Guided 决策表（基于 Web）”	关于添加 Guide Decision Table 的其他细节。
第 4.3.10 节 “规则模板”	关于添加规则模板的内容。
第 5 章 *事实模型（对象模型）*	添加了关于 Fact Model 的其他信息。
第 6 章 *工作集（Working Set）*	添加了关于 Working Set 的新章节。
第 8.3.1 节 “WebDav 和特殊字符”	添加了关于在 WebDAV 里使用多字节字符的说明。

1.2. 什么是 BRMS？

JBoss 企业版 BRMS 是一个商业规则管理系统（Business Rules Management System）。

JBoss 企业版 BRMS 提供了在多用户环境里管理软件商业规则的工具。它是商业规则的 SPOT（Single point of truth），它使用对用户友好的界面，允许修改以可控的方式发生。

JBoss 企业版 BRMS 平台是一个基于 JBoss Rules 的、用于管理、存储、编辑和部署规则和其他 JBoss Rules 资产的服务器端解决方案。它也提供了和 JBoss Developer Studio 以及其他基于 Eclipse 集成开发环境集成的基于 web 的用户界面。

重要

在一些地方，JBoss 企业版 BRMS 平台及其文档指的是 Guvnor。JBoss Guvnor 是构建 JBoss 企业版 BRMS 平台的开源项目。对 Guvnor 的引用保留在 API、URL 和 JBoss Developer Studio 工具里（请参考『第 10 章 *JBoss Developer Studio 集成*』）。

1.3. 什么时候应该使用 BRMS？

商业规则管理系统可在下列情况下使用：

1. 需要管理用于部署和修改规则的系统。
2. 具有不同技能级别的多个用户需要访问和编辑规则。
3. 现有的用于管理规则的基础结构。
4. 需要管理许多“商业”规则，它们不是作为应用程序的一部分的技术规则。

1.4. 谁会使用 BRMS？

使用商业规则管理系统主要是担任这些职务的人：

- 商业分析员
- 规则专家
- 开发人员
- 规则管理员

JBoss 企业级 BRMS 平台允许为不同的用户分配不同的角色以控制开发哪些资产和功能。

1.5. 功能概述

- BRMS web 用户界面支持多语言，目前包括美式英语、日语和简体中文。
- 不同类型的规则编辑器（图形化和文本编辑器都有）。
- 本版控制（用于历史资产）
- 归类
- 构建和部署
- 将多个规则“资产”作为单个软件包存储。

第 2 章 架构

本章涵盖 JBoss 企业级 BRMS 平台设计的技术方面的内容。我们不要求 BRMS 应用程序的最终用户阅读这些内容。它面向的是集成 JBoss 企业级 BRMS 平台和其他系统的开发人员。

图 2.1 “架构图表” 展示了系统的主要组件以及如何集成和部署。

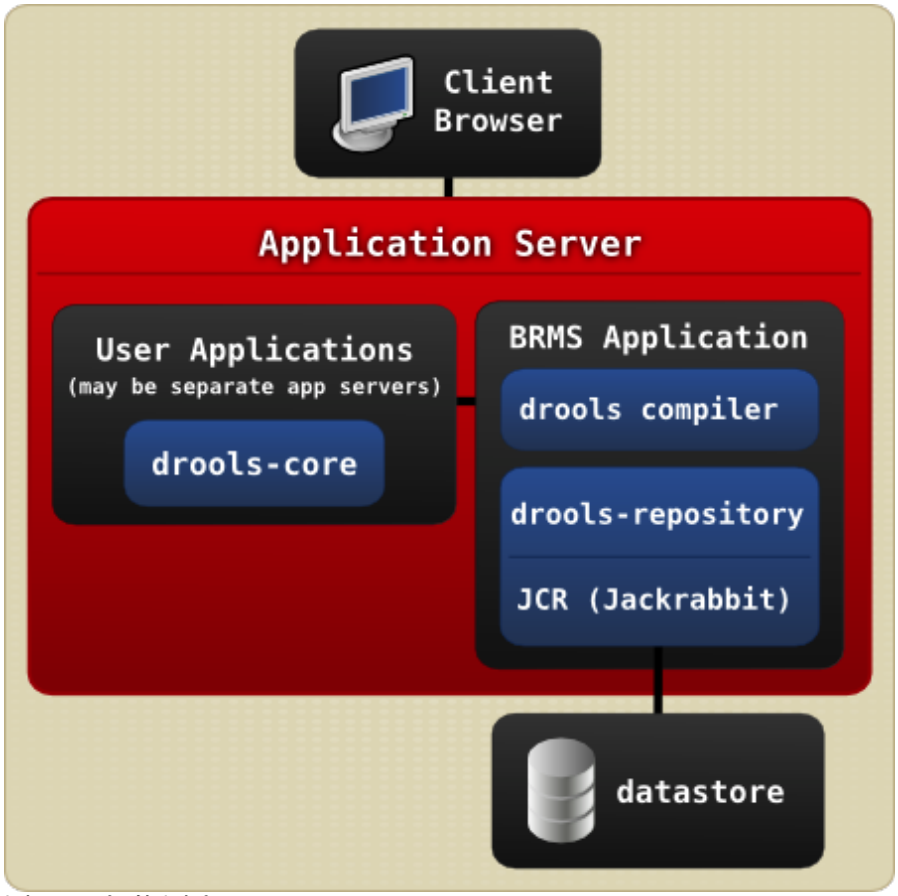


图 **2.1. 架构图表**

BRMS 是以 WAR 形式部署的，它提供 web 用户界面并通过 URL 提供二进制软件包。对于数据存储它采用 JSR-170。JBoss Seam 用于组件框架，而 GWT 作为工具微件以用于构建基于 ajax 的 web 用户界面。

2.1. 可重用的组件

BRMS 使用一个服务接口来分离 GUI 和后台功能。这里的后台包括资产库以及专用于规则处理的编译器。

主要的接口是 **`RepositoryService`**，它在 **`ServiceImplementation`** 里实现。GWT ajax 前端使用 GWT 的异步回调机制和这个接口交互。Seam 的配置文件是 **`components.xml`**。

这个服务接口可以被其他组件或前端重用。

GWT 用户界面可以常用，就像 GWT 是唯一的 HTML 页面：**`Guvnor.html`**。对于熟悉 GWT 的开发人员，每个*功能*都可以独立使用。**`JBRMSFeature`** 类和实现它的类（理论上也可以是独立的）包含了其他的信息。

2.2. 版本和存储

资产的版本和数据一起存储在数据库里。

当创建了*快照*后，整个软件包的备份将存入 JCR 数据库中独立的位置。

对于属性 JCR 和 jackrabbit 的开发人员，**.cnd** 文件在节点类型定义的源码里。软件包是一个*文件夹*，每个资产都是一个文件：资产可以是文本的也可以拥有二进制附件。

第 3 章 快速起步指南

本节将提供 JBoss 企业级 BRMS 平台的快速教程。我们将假设 BRMS 平台和库已经正确地安装并进行了配置。

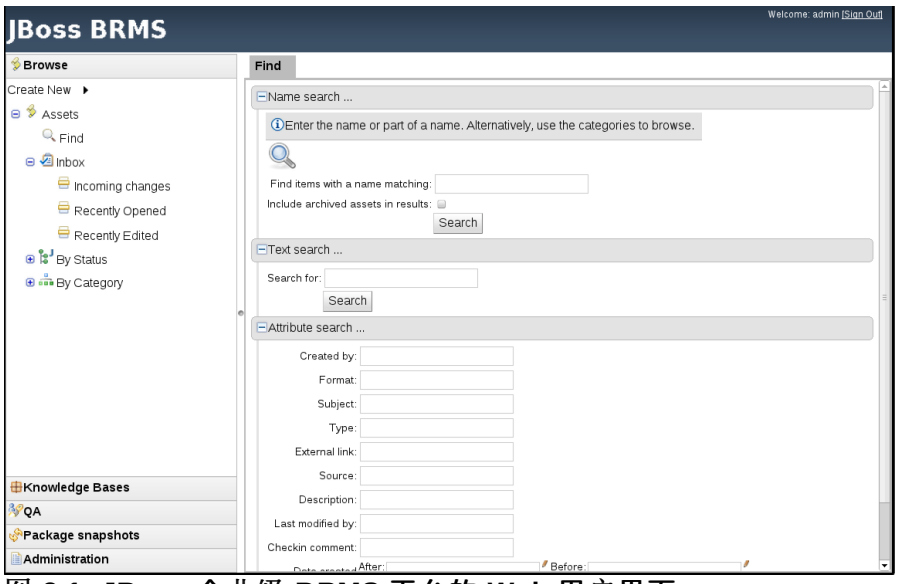


图 **3.1. JBoss 企业级 BRMS 平台的 Web 用户界面**

图 3.1 “JBoss 企业级 BRMS 平台的 Web 用户界面” 显示了 JBoss 企业级 BRMS 平台的主要界面。

左侧的导航面板提供对 BRMS Web UI 的主要区域的访问。这些区域包括：

- Info：这是初始页面，里面有到资源的链接。
- Rules：这是类别和商业用户视图。
- Package：这是配置和管理知识软件包的地方。
- Deployment：这是管理部署快照的地方。
- Admin：管理性功能（归类、状态、导入和导出）。

3.1. 系统支持的浏览器

表 3.1 “系统支持的浏览器” 列出了系统支持的访问 BRMS Web 用户界面的浏览器。

表 3.1. 系统支持的浏览器

操作系统	浏览器
RHEL 5.x 及更高版本	FireFox 3.0+
Microsoft Windows	FireFox 3.0+
Microsoft Windows	Internet Explorer 7+
Mac OSX 10.x	FireFox 3.0+
Mac OSX 10.x	Safari 4 和 5

3.2. BRMS 还是 Guvnor?

在以前的 Drools 版本里，“BRMS”经常指的是 drools 管理功能的 web 界面。现在，我们使用 BRMS 来表示

“整个软件包”- runtime、web 工具等等。但在某些情况下，你还是可以将“BRMS”理解为 Guvnor web 控制台和相关联的工具。

3.3. 初始的配置

有些步骤是在进行初始配置时要求的。服务器第一次启动时，它将创建一个空的库，然后按照下列步骤进行：

- 如果是一个全新的库，你需要进入“Admin”，然后选择“Manage Categories”。
  添加一些类别（注意这些类别只适用于分类目的）。
- 规则需要使用一个事实模型（也称为对象模型）。从 "Package Management" 功能里，你可以创建一个新的知识软件包（knowledge package）。软件包称应该有一个有意义的名称，而且不包括空格。
- 要上载模型，你可以使用一个包含代码和规则里将使用的事实模型（API）的 **`.jar`** 文件。在 "Model Editor" 屏幕里，你可以上传一个 **`.jar`** 文件。为此，请从你在前一步骤里创建的列表里选择软件包。
- 为了导入你上传的事实类型（add import 语句），现在需要对刚创建的软件包配置进行编辑。请保存修改。
- 此时，软件包就完成了配置且可以使用了。
  请注意，你也可以导入一个现有的 DRL（Drools Rule Language）软件包，其规则将和单个资产一样存储在库里。

3.4. 编写规则

- 在配置了至少一个类别和软件包之后，你就可以开始编写规则了。
- 虽然有多个规则格式，但 BRMS 都将其视为“资产”。
- 通过
- 你也得选择一个类别。分类提供了独立于知识软件包查看规则的途径（确实，你甚至可以使规则出现在多个知识软件包里）。你会发现把它当作某种标签将有助于理解。
- 选择 "Business Rule (Guided Editor)" 格式。
- 这将打开一个规则模型（Guided Editor）。你可以添加和编辑当前软件包里使用的模型的条件和行为。而且，为这个软件包配置的任何 DSL Sentence 模板都是可用的。
- 当你完成了规则的编辑后，你可以检入所做的修改（换句话说就是保存修改），或者你可以选择验证或“查看源”（对于有效的源而言）。
- 你也可以从规则编辑器里添加或删除类别并修改其属性，如文档等。（如果你不确定怎么去做，请用自然语言描述规则并检入到系统里。以后这可当作模板。）

3.5. 搜索

为了浏览系统，你可以使用规则功能（显示分类信息），或者你可以使用软件包功能并根据软件包（或规则类型）来查看。如果你知道资产的名称或是名称的一部分，你也可以使用“Quick Find（快速查找）”。输入规则名称后，BRMS 将返回一个匹配的列表。

3.6. 部署

- 在软件包里编辑了规则后，你可以点击“Package”功能，打开软件包并进行构建。
- 如果构建过程成功完成，你应该可以下载二进制软件包文件并将其部署到运行系统里。
- 你也可以制作软件包的“快照”以用于部署。这会固定某个时刻的软件包的内容，任何的并发修改都不会影响到它。这会使得软件包可通过下列格式的 URL 来访问：**`http://<your server>/jboss-`**

brms/org.drools.guvnor.Guvnor/packages/<packageName>/<snapshotName>。

第 4 章 BRMS 概念

4.1. 规则是资产

资产（Asset）是可以以某个版本存储在库里的任何东西。它包括规则、决策表、模型、测试和 DSL。

注意

术语“规则（rule）”用得很广泛。它有时候指的是实际上并非资产的规则。

4.2. 分类

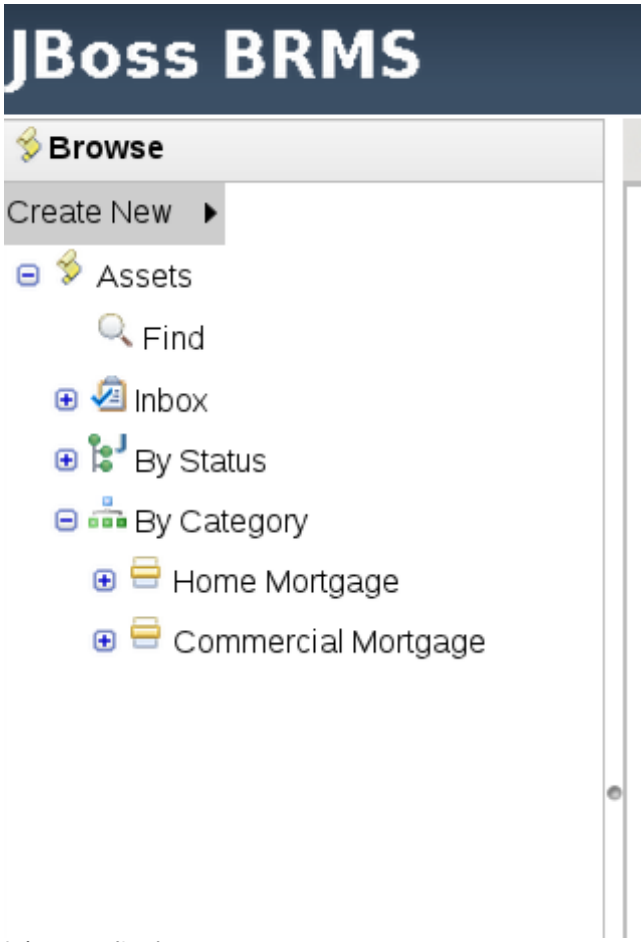


图 4.1. 类别

类别允许规则（资产）用你定义的任何数量的组别进行标记。这意味着你可以查看匹配某个类别的一个规则列表。规则可以属于任何数目的类别。在上面的图表里，你可以看到的是类似于文件夹/浏览器的资产视图。其名字是由 BRMS 管理员定义的（你也可以删除/添加新类别；但你只能在它们没有在使用时删除它们），可以是你希望的任意名称。

通常，类别是用匹配规则应用的商业领域的有意义的名称来创建的。（因此，如果规则应用在多个领域，你可以附加多个类别。）类别也可以用来标记规则为生命周期的某个阶段，例如标记为 "Draft“ 或 "For Review“。

图 4.2. 可以有多个类别的资产

上面的视图显示了你打开资产时看到的 Category Editor/Viewer。在这个例子里，你可以看到资产属于两个类别。这意味着当其中任何一个类别用于显示资产列表时，你都将看到该资产。（“+” 按钮和 Garbage Can 图标可分别用来添加和删除其他项目。）

在上面的例子里，第一个类别（'Home"）位于 "top level"。而第二个类别 “Mortage/Eligibility" 仍是一个单一的类别，但它是嵌套的，这是因为类别是具有层次结构的。要以另外一种方式表述它，就是名为 “Mortage" 的类别包含一个类别 “Eligibility“。屏幕将它显示为 ”Mortage/Eligibility”。如你所看到的，

当你打开一个资产以进行查看或编辑时，你会看到它所属的类别列表。如果你进行了修改（删除或添加类别），你将需要保存资产，且这个行为将在版本历史里创建一个新的条目。修改规则的类别不会影响其执行。

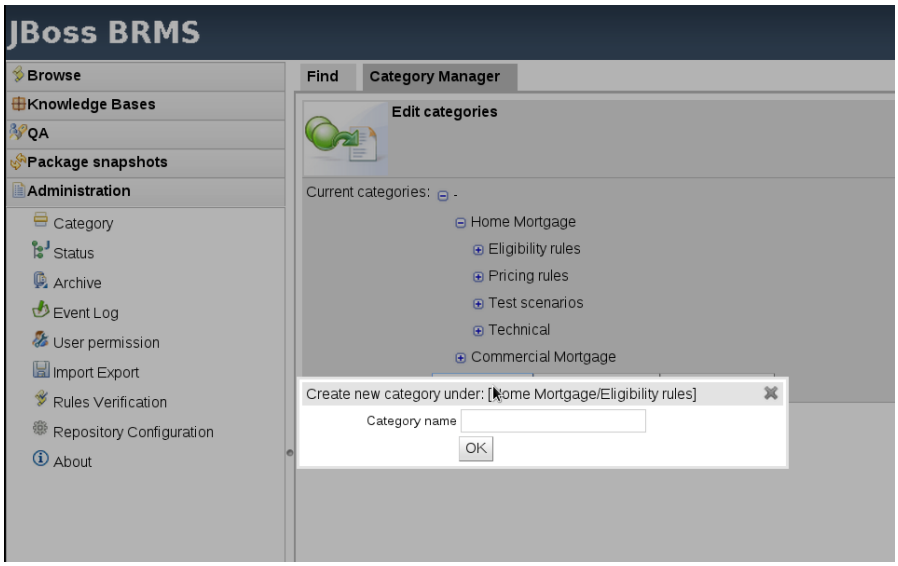


图 4.3. 创建类别

上面的视图显示了设立类别的管理屏幕。系统里缺省是没有类别的。因为类别是具有层次结构的，创建子类别时你必须选择“父”类别。在这里，你也可以删除类别（但只能在它们没有被任何资产使用的情况下）。

注意

作为一个普通的规则，资产应该属于一个或两个类别。当你有大量的规则时，类别是很重要的。其层次结构不需要太深。红帽设计它们是为了帮助你将规则/资产分解为可管理的小片内容。

4.3. 规则授权

BRMS 平台支持几个资产（规则）格式。我们描述的是关键的格式。本手册里的其他部分将涵盖另外一些格式，其细节则不会在此重复。

4.3.1. 资产编辑器（Asset Editor）

资产编辑器里的可用选项取决于编辑的资产类型。资产编辑器用于编辑规则，它也可以用于添加、删除、出

席排序条件、约束、行为、字段和选项。资产编辑器也允许用户使用工作集、检验和验证规则以及查看规则源。

相关的细节请查看资产编辑器里的工具提示。

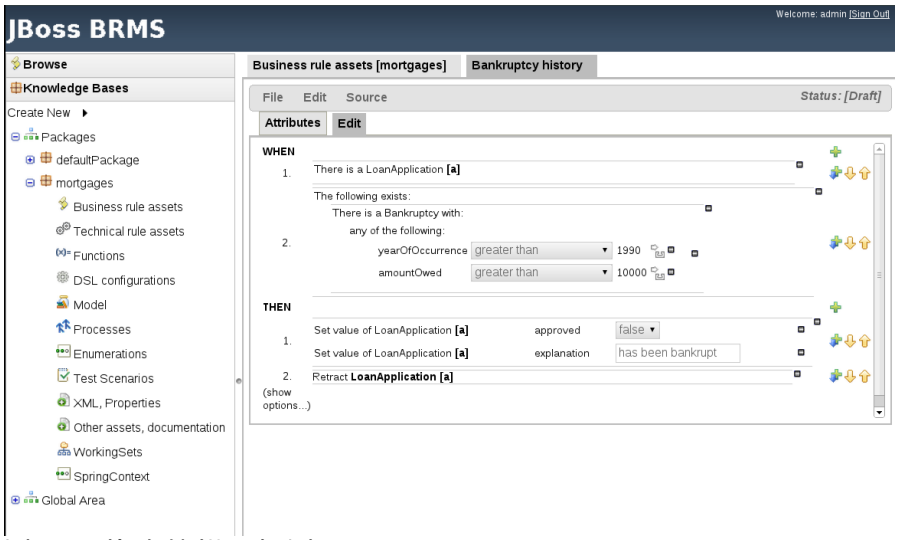


图 4.4. 资产编辑器视图

4.3.2. 用 Guided Editor 编辑商业规则

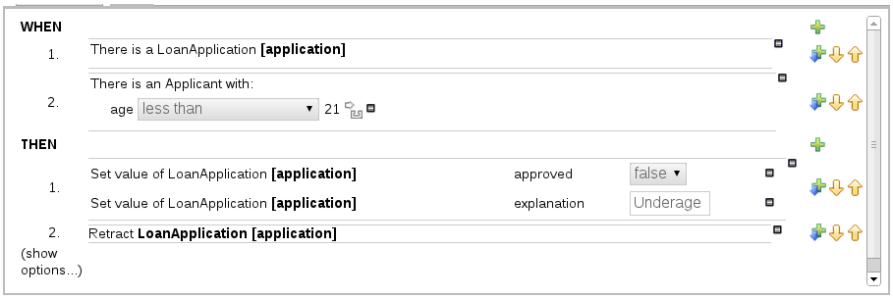
当编辑规则时，资产编辑器也称为 "Guided Editor"。**`Guided Editor`** 用于编辑商业规则语言（BRL）格式的规则。Guided Editor 提示用户根据正在编辑的规则的对象模型进行输入。

在你可以使用 BRL Guided Editor 之前，你必须先配置软件包访问。

注意

Eclipse 插件里也包含一个 Guided Editor。本节里大部分细节也适用于它。

例 4.1. Guided Editor



4.3.3. 规则的分解

规则由多个部分组成：

- When

  规则的 *When* 部分是一个必须满足的条件。例如，在 例 4.1 “Guided Editor” 里，when 部分是：当申请

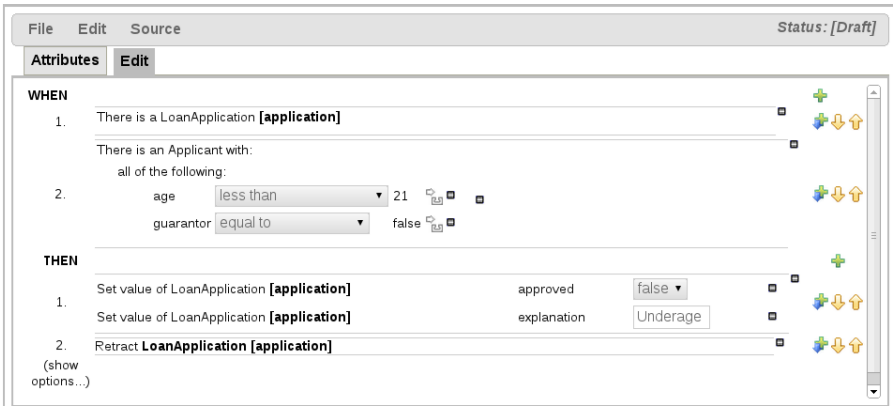
者低于 21 岁。

- Then

  规则的 *Then* 部分是规则的条件部分已经满足后必须执行的行为。例如，在 例 4.1 “Guided Editor” 里，when 部分是：当申请者低于 21 岁。

使用 Guided Editor 也可以在 When（或条件）部分添加更多的条件并在 Then（或行为）部分添加更多的行为。例如，如果一个 21 岁以下的申请者有贷款申请的担保人，银行可能决定批准它的贷款申请。为了将这种情形在 例 4.1 “Guided Editor” 里模型化，修改何时包括担保人条件是有必要的。

例 4.2. 修改规则



要在条件里添加担保人，你首先需要在 Mortgage 模型的 Fact Type 里添加 **`guarantor`** 字段。

过程 4.1. 在 Fact Type 里添加字段

1. **选择模型**

   从屏幕的左侧导航面板里选择 **`Knowledge Bases`**。扩展包含模型的软件包并选择 **`model`**。

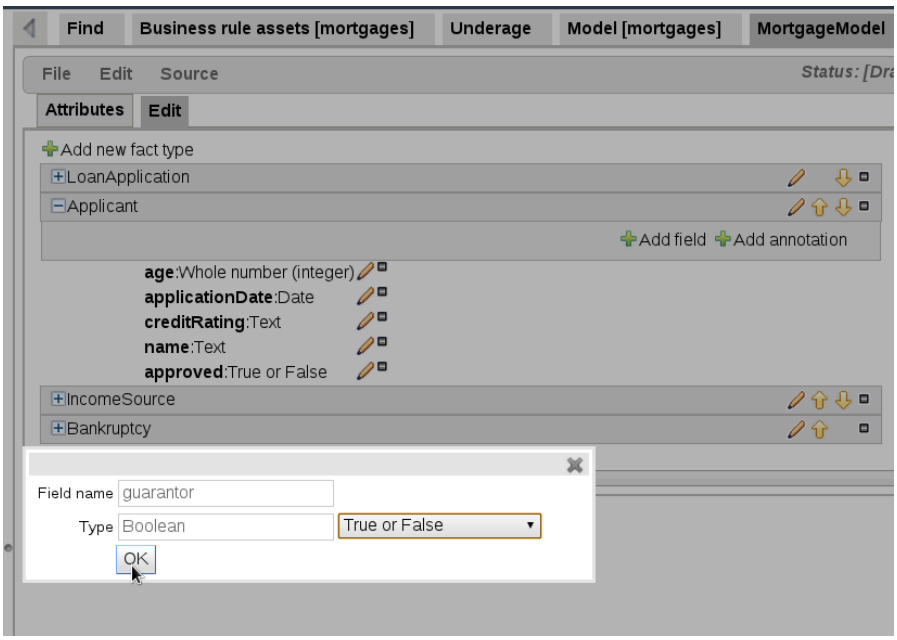
   点击 **`open`** 从列表里打开模型。

2. **添加字段**

   点击后面的加号以扩展 Fact Type，然后选择**添加字段（`Add Field`）**。

3. **输入字段细节**

   添加细节到弹出对话框里。在 **`Field name`** 字段里输入名称 *guarantor*，然后从 **`Type`** 下拉菜单里选择 **`True or False`**。



选择 **`File`** 和 **`Save changes`** 将修改保存到模型里。

因为在 applicant fact 类型里添加了 guarantor 字段，我们有可能修改规则以包括一个 guarantor。

过程 4.2. 在规则里添加限制

1. **删除当前的限制**

   当编辑现有的规则时，你有必要删除当前的限制。这可以确保处理多个限制时选择正确的逻辑操作符（And 或 Or）。

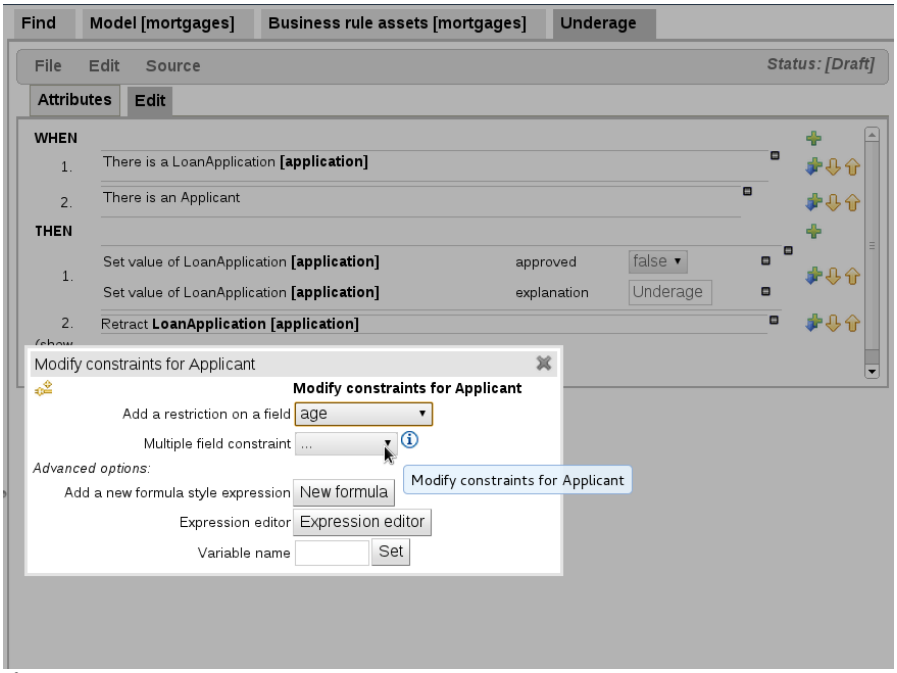
   在这个例子里，点击 Age 条件后面的减号。

2. **修改约束**

   点击文本 *There is an Applicant* 打开 **`Modify Constraints Dialogue`**。

   从 **`Add a restriction on a field`** 下列菜单里选择限制。在这个例子是选择 **`Age`**。

   从 **`Multiple field constraints`** 下列菜单里选择 **`All of (and)`**。



3. **指定第一个约束**

   点击文本 *all of the following* 以打开 **`Add fields to this constraint`** 对话框并添加限制。在

这个例子里是选择 **Age**。

从新的下拉菜单里选择 **less than**，然后点击铅笔图标以编辑它的值，选择 **literal**，然后点击 **Value** 输入合适的值，这个例子里是输入 21。

4. **指定第二个约束**

   点击文本 *all of the following* 以打开 **Add fields to this constraint** 对话框并添加限制。在这个例子里是选择 **Guarantor**。

   从新的下拉菜单里选择 **equal to**，然后点击铅笔图标以编辑它的值，选择 **literal**，然后点击 **true** 输入合适的值，并选择 False。

4.3.4. Saliance

你可以通过 Guided Editor 里的 Options 中编辑 Saliance。Salience 是一个代表优先级的数值。

4.3.5. 用户驱动的下拉列表

请注意，你可以将字段值限制为预先配置的列表条目。这个列表配置为软件包（用于填充值的数据枚举）的一部分。这些值可以是固定列表，如从数据库里加载的值。对于具有给定值的代码和其他字段，这尤其有用。它也可以以下拉列表的方式显示在屏幕上，和规则里使用的值（或代码）不一样。关于如何进行配置，请参考 "Data Enumerations" 章节里的内容。

4.3.6. 增加 DSL 语句

如果规则所属的软件包具有 DSL 配置，在你添加条件或行为时，它将提供一个 “DSL 语句” 列表，你可以从中选择（选择一个条目会添加一行规则），凭借来自用户的 DSL 指定值，编辑框（文本）将会显示（它组成一个表单）。请注意这是可选的，你可以选择其他 DSL 编辑器。

重要

Guided Editor 目前不支持完整的 DSL 功能，它只支持 DSL 配置的 **[when]** 和 **[then]** 部分。

下面的图表显示了 Guided Editor 里的 DSL 语句：

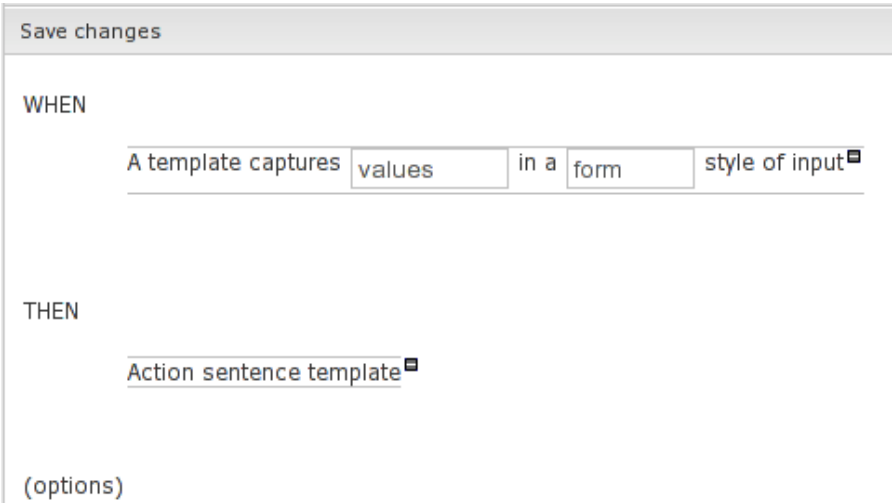


图 **4.5. Guided Editor** 里的 **DSL**

4.3.7. DSL 规则

DSL 规则是文本值，它使用一个语言配置资产来控制其外观。

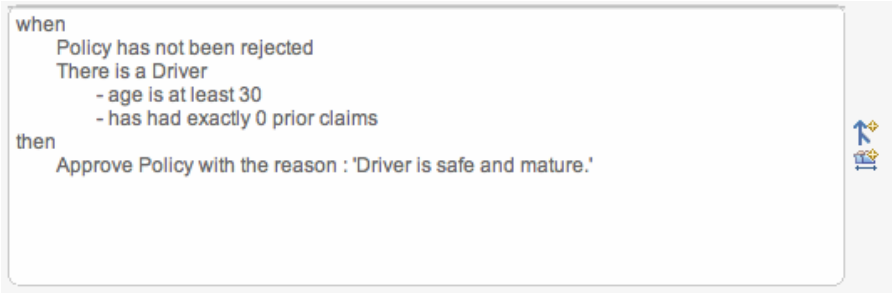


图 **4.6. DSL** 规则

DSL 是一个单一的规则。在上面的图片里，你可以看到文本编辑器。你可以使用右侧的图表来提供条件和行为列表以供选择（或者同时按住 Control 和空格键来弹出列表）。

4.3.8. 电子表格决策表

多个规则可以存储在 Microsoft Excel 的电子表格里（**`.xls`** 文件格式），每一行都代表一个规则。本章不会涉及电子表格的细节。

图 **4.7.** 电子表格决策表

要使用电子表格，如 图 4.7 “电子表格决策表” 所展示的，上传一个 **`.xls`** 文件。你也可以下载从相同的对话

框下载当前版本的电子表格。为了创建一个新的决策表，你必须启动“规则向导”，它包含一个这个过程的选项，然后你可以上传 **.xls** 文件。

4.3.9. Guided 决策表（基于 Web）

Guided 决策表功能允许决策表在 web 上编辑。它通过内省哪些 fact 和字段可用来引导决策表的创建，这和 Guided Editor 类似。规则属性、元数据、条件和行为都可以以治表格式定义，从而有助于大量相关规则的快速输入。象其他规则资产一样，基于 Web 的决策表会被编译成 DRL。

你可以多种方式选择单元格：

- 你可以双击单个的单元格以弹出对应底层数据类型的编辑器。相同列的单元格组可以通过点击第一个单元格并拖动光标或在点击第一个单元后再点击要选取范围里的最后一个（同时按住 Shift 键）来进行选择。
- 或者，你也可以使用箭头键在表格里移动。按住 Enter 键将弹出对应的编辑器。按住 Shift 同时用箭头键扩展选择范围可以选择一个区域。

如要修改列的大小，用鼠标在表头的分界处悬停，然后拖动鼠标光标将使列变窄或变宽。

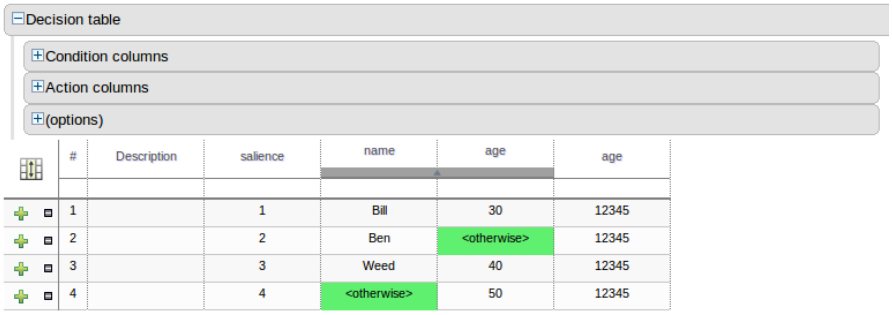


图 **4.8. 决策表**

4.3.9.1. 主要的组件

Guided 决策表分成两个主要的部分：

- 上面的部分允许定义表的列来代表规则属性、元数、条件和行为。
- 下面的部分包含实际的表，而每一行都定义单独的规则。

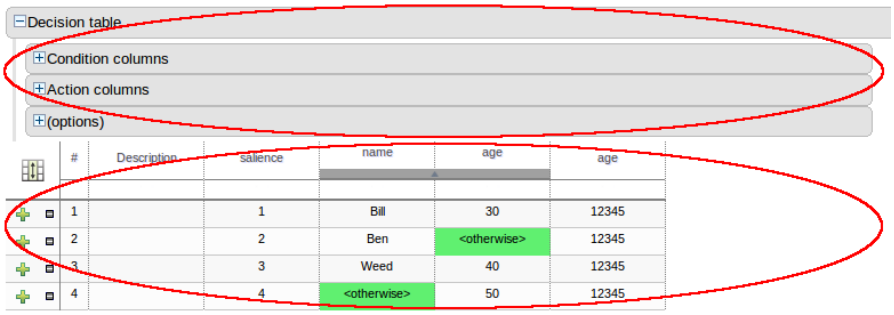


图 **4.9. 主要的组件**

4.3.9.2. 列配置

列可以有下列的约束类型：

- Literal
  单元格里的值将用操作符和字段进行比较。
- Formula
  单元格里的表达式将被评估，然后和字段进行比较。
- Predicate
  不需要字段，表达式将被评估为 true 或 false。

你可以设置缺省值，但通常在单元格里没有值的话，约束不会被应用。

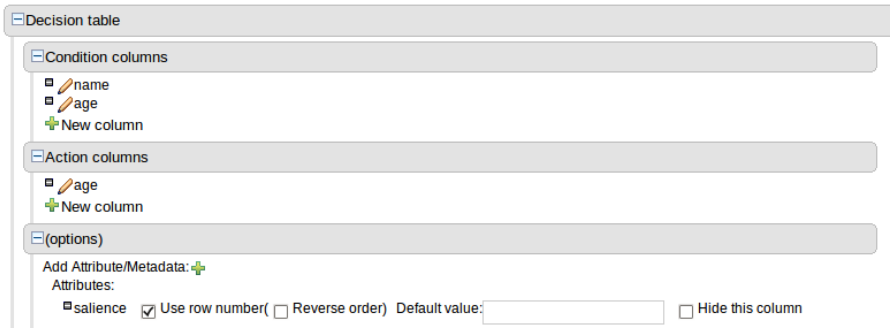


图 **4.10. 列配置**

4.3.9.2.1. 工具列

在缺省情况下，包含规则号码和描述的两列都会被提供。

4.3.9.2.2. 属性列

你可以添加代表任何 DRL 规则属性到零或多个属性列。在 Guided 决策表编辑器来还有一个额外的伪属性来 "否定（negate）" 规则。使用这个属性可以否定整个规则。例如，下面的简单规则可以被否定。

```
when
  $c : Cheese( name == "Cheddar" )
then
  ...
end
```

```
when
  not Cheese( name == "Cheddar" )
then
  ...
end
```

4.3.9.2.3. 元数据列（Meta-data column）

你可以定义零个或多个元数据，每个都代表 DRL 规则上正常的元数据注解。

4.3.9.2.4. 条件列（Condition Column）

条件代表右手边定义的事实模式（fact pattern）或规则的 "When" 部分。要定义一个条件列，你必须定义模型类绑定或者选择一个已经定义的绑定。完成后你就可以定义字段约束了。如果两个或多个列都使用了相同的事实模式绑定，字段约束将变成相同模式的复合字段约束。如果你为单个的模型类定义了多个绑定，每个绑定都会变成规则右手边的单独的模型类。

4.3.9.2.5. 行为列（Action Column）

你可以定义行为列来在规则引擎的工作内存里执行绑定事实上的简单操作或者创建全新的事实。新的事实可以在逻辑上插入到规则引擎的工作内存，从而服从平常的真相维护（truth maintenance ）。关于真相维护和逻辑插入的更多信息，请参考*《JBoss Rules 参考指南》*。

4.3.9.3. 规则定义

本节允许使用之前定义的列来定义个体规则。

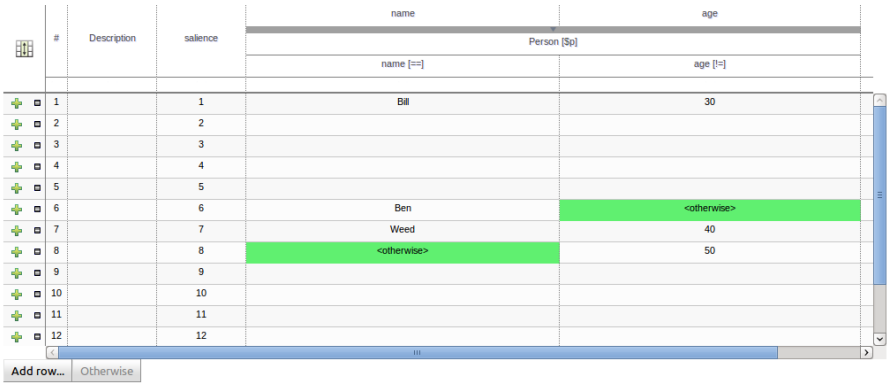


图 **4.11.** 规则定义

4.3.9.4. 单元合并

决策表左上角的图标切换单元格合并的开与关。当进行单元格合并时，那些在相同列里且具有相同值的单元格将合并成单个的单元格。这简化了修改分享相同原始值的多个单元格的内容。当单元格被合并时，它们也在单元格的左上角显示一个图标以允许对跨越合并单元格的行进行分组。

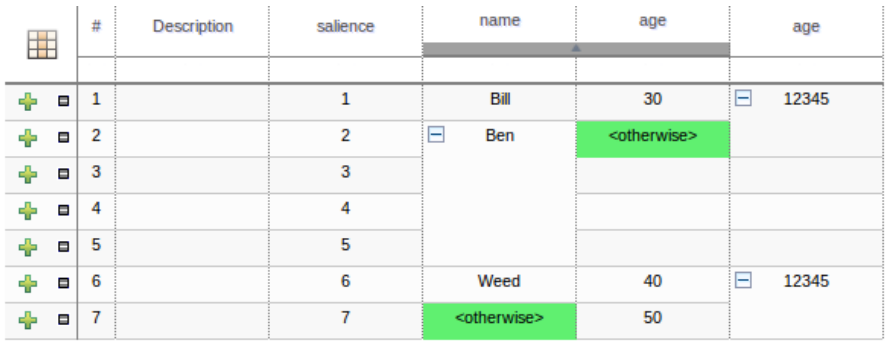


图 **4.12.** 单元合并

4.3.9.5. 单元分组

已经被合并的单元格可以进一步收缩成单一的行。点击左上角的 [+\-] 图标将收缩对应的行为一行。跨越收缩的具有相同值的行的其他列里的单元格将不受影响。而跨越收缩的具有不同值的行的其他列里的单元格将高亮显示且显示第一个值。

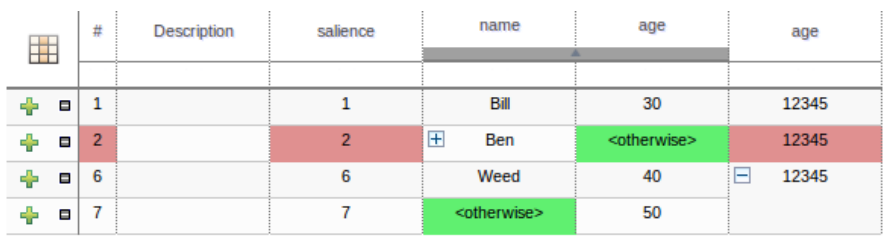


图 **4.13.** 单元分组

当分组单元格的值被改动时，已经收缩的所有的单元格都将更新它们的值。

4.3.9.6. Otherwise 操作

定义为使用等号 **==** 或不等号 **!=** 操作符的 literal 值的条件列可以利用一个特殊决策表的单元格值 **otherwise**。这个特殊的值允许定义规则来匹配没有在表里其他规则里显性定义的值。我们最好用以下的例子来解释：

```
when
  Cheese( name not in ("Cheddar", "Edam", "Brie") )
  ...
then
  ...
end
```

```
when
  Cheese( name in ("Cheddar", "Edam", "Brie") )
  ...
then
  ...
end
```

4.3.10. 规则模板

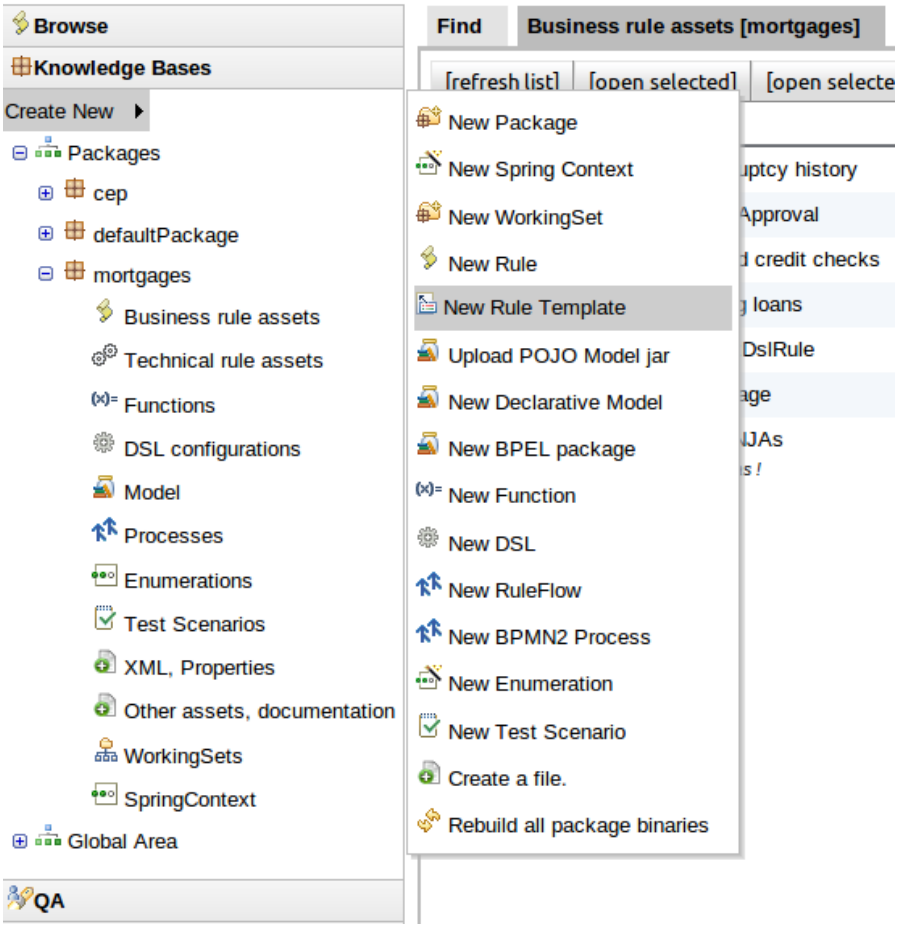
规则模板允许用户定义一个带有占位符（其值从表数据里获取）的规则结构。也你也可以继续使用 Literal 值、公式（Formula）和表达式（Expression）。

规则模板经常可用来替换决策表。

过程 4.3. 创建规则模板

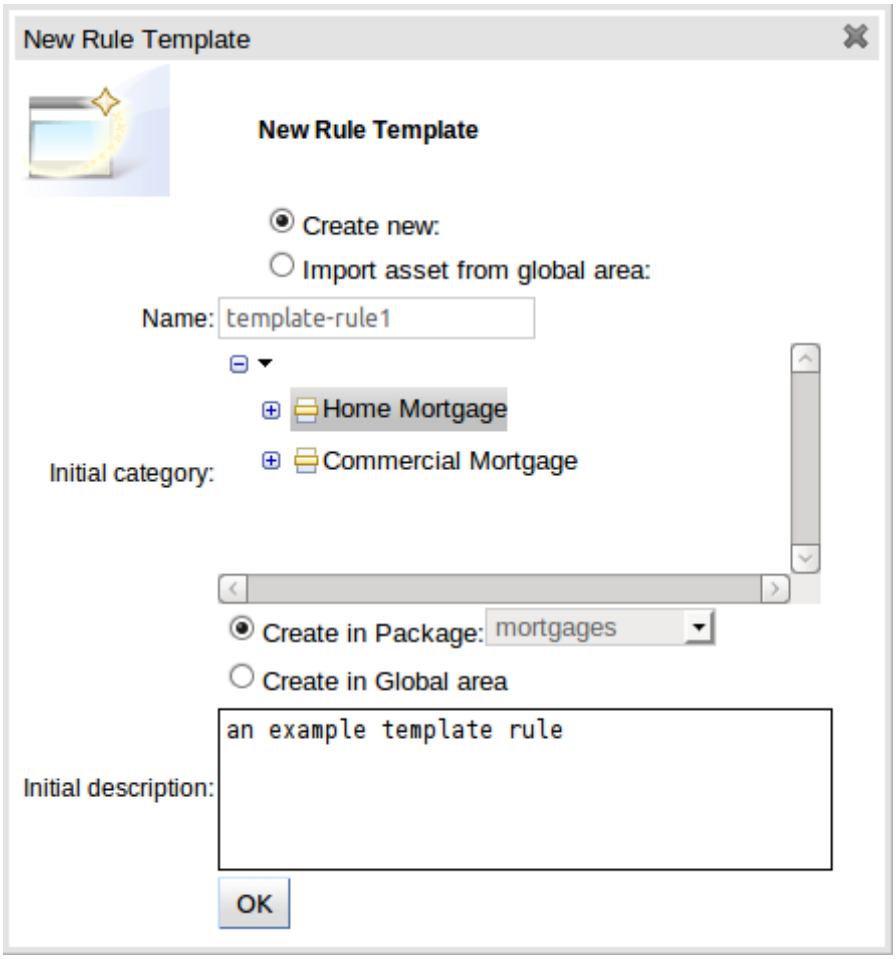
1. 创建规则模板资产

   从 **Knowledge Bases** 菜单，选择 **Create New**，然后选择 **New Rule Template**。



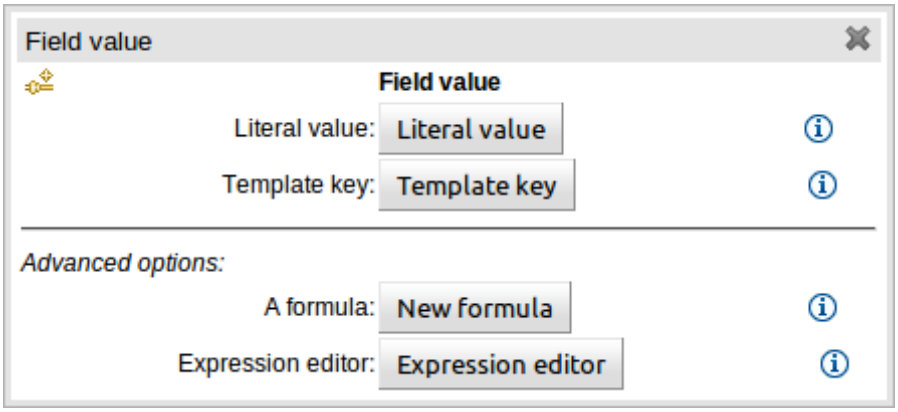
2. **定义资产**

   输入模板的名称、类别和描述。



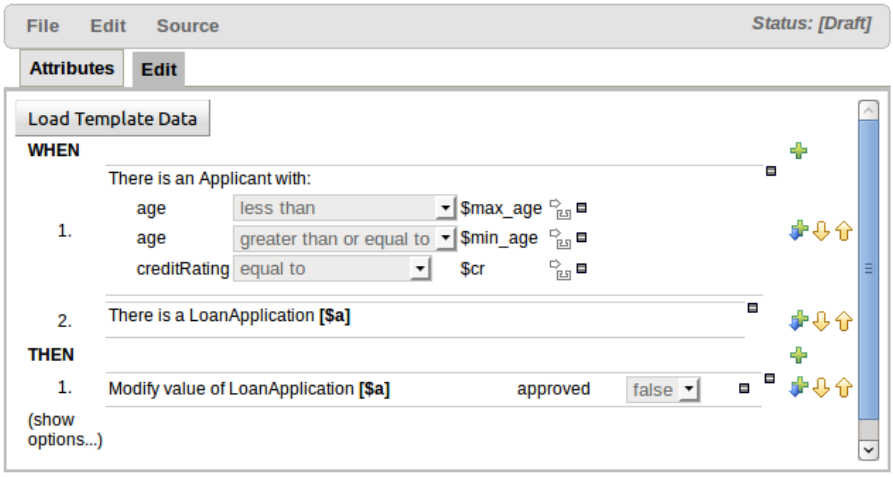
3. **定义模板**

   使用 guided editor 来构造规则。**模板关键字（`Template key`）** 是字段约束和行为部分里的占位符。你也可以在标准 guided editor 继续使用 Literal 值、公式（Formula）和表达式（Expression）。



例 4.3 “示例模板” 解释了一个已经用模板关键字（Template Key）定义的简单规则，它用于申请者的最大年龄、最小年龄和信用级别。模板关键字已经分别定义为 "$max_age"、"$min_age" 和 "$cr"。

例 4.3. 示例模板



4.3.10.1. 定义模板数据

当你完成规则模板的定义后，你需要输入用于插入“模板关键字”占位符的数据。BRMS 提供了在 Guided Editor 屏幕上的灵活的网格里输入数据的机制。在 Guided Editor 屏幕上按 **`Load Template Data`** 按钮可以启动网格编辑器。

规则模板的数据网格是非常灵活的，对于底层不同的数据类型有不同的弹出编辑器。列可以调整大小和排序，而单元格可以合并和分组以助于快速的数据录入。

每个**`Template Key`** 占位符可插入一行数据，由此每一行都称为了一条规则。

注意

如果某一样的任何单元格为空，那么该行规则就不会生成。

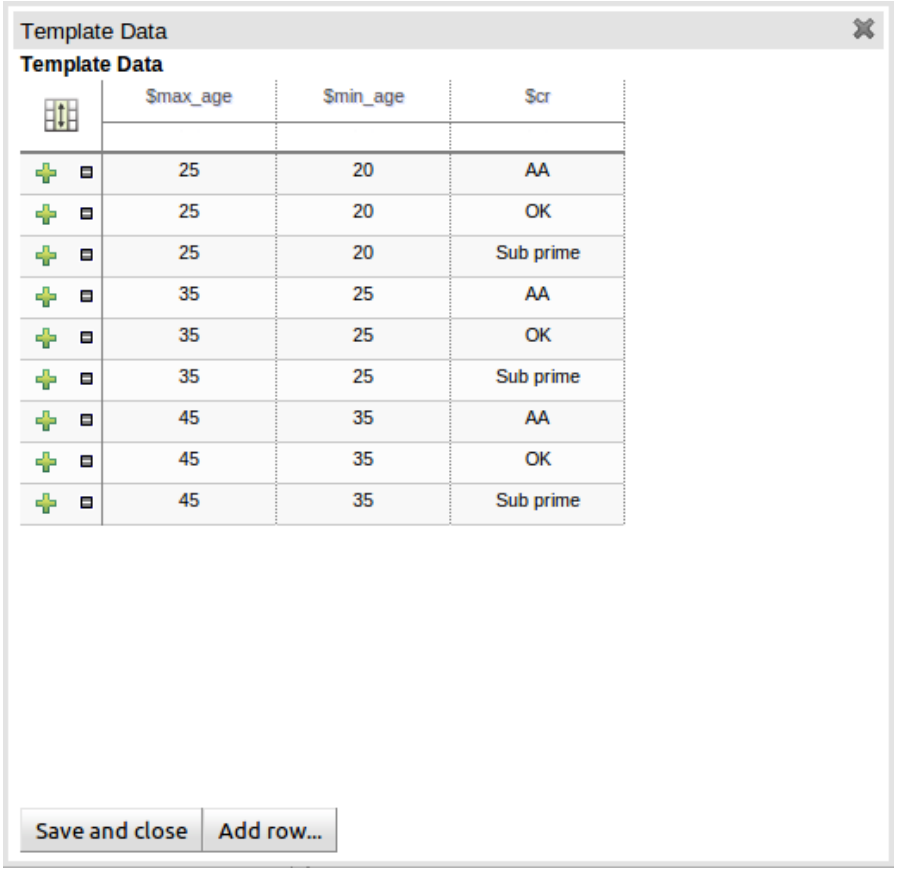


图 4.14. 模板数据网格

4.3.10.1.1. 单元合并

网格左上角的图标切换单元格合并的开与关。当进行单元格合并时，那些在相同列里且具有相同值的单元格将合并成单个的单元格。这简化了修改分享相同原始值的多个单元格的内容。当单元格被合并时，它们也在单元格的左上角显示一个图标以允许对跨越合并单元格的行进行分组。

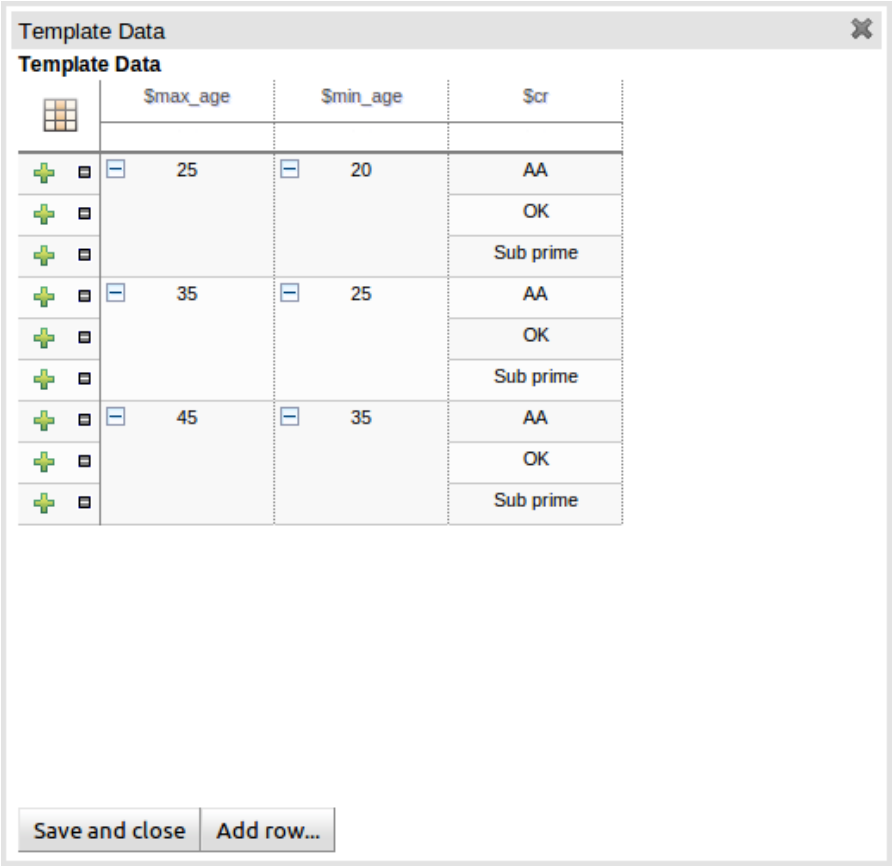


图 **4.15.** 单元合并

4.3.10.1.2. 单元分组

已经被合并的单元格可以进一步收缩成单一的行。点击左上角的 [+\-] 图标将收缩对应的行为一行。跨越收缩的具有相同值的行的其他列里的单元格将不受影响。而跨越收缩的具有不同值的行的其他列里的单元格将高亮显示且显示第一个值。

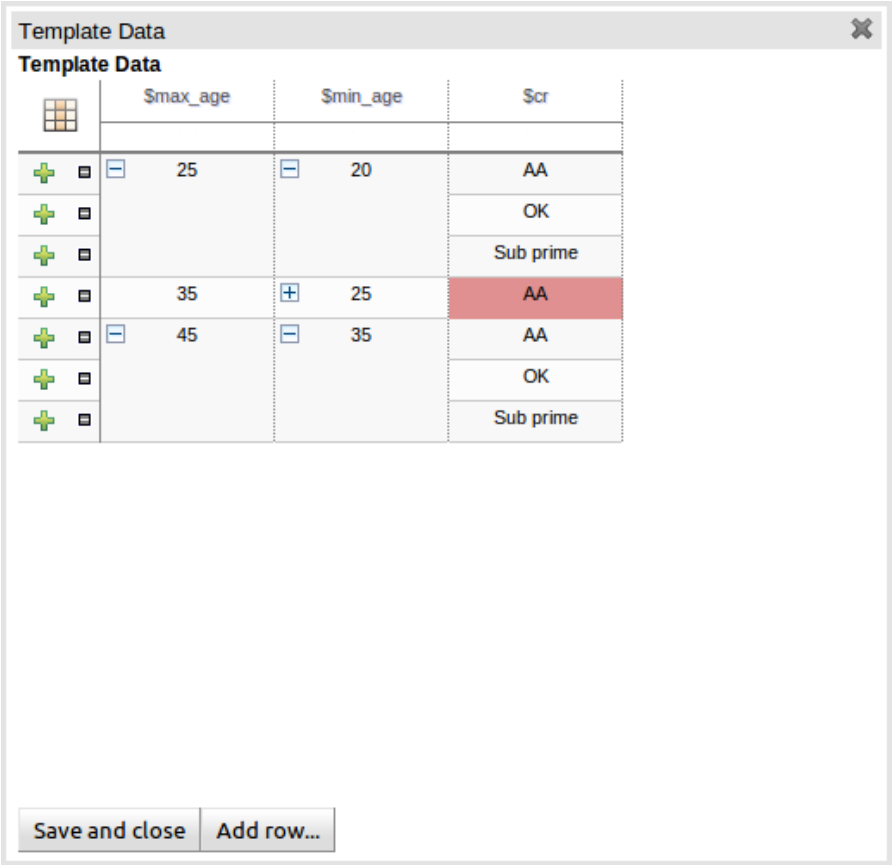


图 4.16. 单元分组

当分组单元格的值被改动时，已经收缩的所有的单元格都将更新它们的值。

4.3.10.2. 生成的 DRL

规则作者可以查看将为规则模板和相关联的数据生成的 DRL，虽然这并没有必要。这个功能及其操作和用于其他资产的没有区别。选择 **`Source`**，然后再从资产编辑器屏幕选择 **`View Source`** 菜单。这将显示所有规则的 DRL。

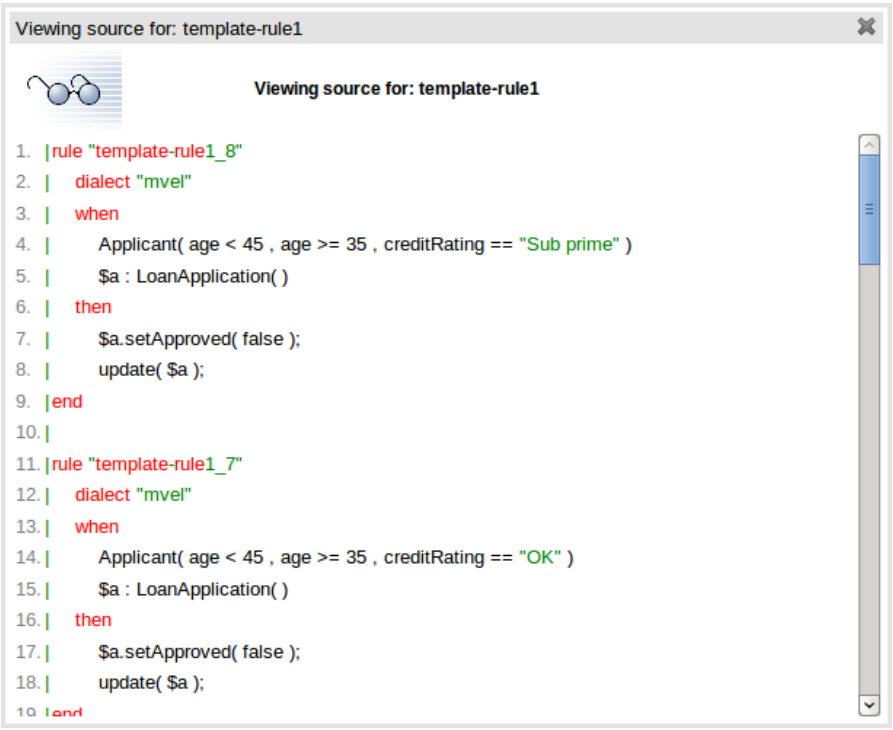


图 4.17. 生成的 DRL

4.3.11. 规则流

规则流：规则流允许你用可视方式描述进行的步骤 - 所以不是所有的规则都会立即进行评估，这是一个逻辑流。本章不会涵盖 BRMS 上的规则流，但你可以使用 IDE 画出图形化规则流并上传 .rfm 到 BRMS。

和电子表格相似，你可以上传/下载规则流文件（Eclipse IDE 有个对应的图形化编辑器）。我们不会在此讨论规则流的细节。

4.3.12. 技术性规则（DRL）

技术性规则（DRL）存储为文本且可以在 BRMS 里进行管理。DRL 文件可以包含一个或多个规则。如果文件只包含一个规则，那么软件包、导入和规则语句都是不必要的。你可以只使用 "When" 和 "Then" 来分别标记 "Condition“ 和 "Action" 部分。

通常你将使用集成的开发环境来编辑 “原生” DRL 文件，这是因为 IDE 拥有高级的工具、内容助手和调试功能。然而，有时候规则得处理 BRMS 里的软件包里的非常技术性的东西。在任何典型的规则软件包里，你通常都需要 “技术性规则”。当然你也可以将所有的规则类型混合并进行匹配。

```
salience 100 #this can short circuit any processing
when
    a : Approve()
    p : Policy()
then
    p.setApproved(true);
    System.out.println("APPROVED: " + a.getReason());
```

图 4.18. DRL 技术性规则

4.3.13. 功能

功能是另外一类资产。它们不是资产且应该只在需要时才被使用。"*Function Editor*" 是一个文本编辑器。

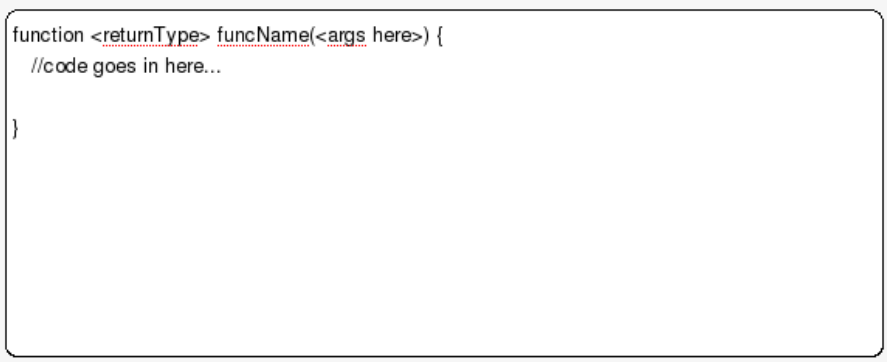


图 **4.19. 功能**

4.3.14. 数据枚举（下拉列表配置）

数据枚举（Data enumeration）是资产的一个可选类型，你可以配置它为 Guided Editor 提供下拉列表。它们和其他资产一样可以存储和编辑，且只应用于它们所属的软件包。

枚举配置的内容是 "Fact.field" 到一个值列表的映射。这些值用在一个下列列表里。这个列表可以是文字的，或者使用工具类（你放入 classpath 里的）来加载字符串。这些字符串包含显示在下拉列表里的值，或者代码值（规则里使用的结尾）的映射和显示值（请看下面的例子，它使用 '='）。

```
'Board.type' : [ 'Short', 'Long', 'M=Mini', 'Boogie']
'Person.age' : [ '20', '25', '30', '35' ]
```

在上面的枚举配置里，"M" 表示用在规则里的值，而 "Mini" 则是显示在图形界面里的值。

从外部源获取数据列表

BRMS 也可以调用加载字符串列表的代码。为此，你需要一段返回 **`java.util.List`** 到 BRMS 的 classpath 的代码。不是指定 BRMS 的值列表，代码是返回字符串列表。（通常，如果你想为规则使用不同的显示值，你可以在字符串里使用 "=" 符号。）例如，你可以修改上面的 **`Person.age`** 行为：

```
'Person.age' : (new com.yourco.DataHelper()).getListOfAges()
```

这假设你有一个名为 **`DataHelper`** 的类，它有一个返回字符串列表（位于 classpath 上）的方法 **`getListOfAges()`**。你当然可以将这些 “动态的“ 枚举和固定列表混合。例如，你可以用 JDBC 从数据库里加载它们。你第一次在会话里使用 Guided Editor 时将加载数据枚举。如果你有任何打开的 Guided Editor 会话，你将需要关闭它并重新打开规则才能看到发生的变化。要检查枚举是否已经加载，请进入软件包配置屏幕。你可以使用"保存并检验（save and validate）"软件包，这可以检查并反馈任何发生的错误。

4.3.15. 高级的枚举概念

对于数据枚举，你还可以执行一些其他的高级任务。

取决于字段值的下拉列表

假设有一个简单的事实模型，你有一个名为 Vehicle 的类，它有两个字段组成："**`engineType`**" 和 "**`fuelType`**"。对于 "**`engineType`**" 你希望选择 "**`Petrol`**" 或 "**`Diesel`**"。现在，很明显燃油的选择必须取决于引擎类型（对于汽油我们有 ULP 和 PULP，而对于柴油我们有 BIO and **`NORMAL`**）。我们可以在枚举

表达式里表达这种依赖关系：

```
 'Vehicle.engineType' : ['Petrol', 'Diesel']
 'Vehicle.fuelType[engineType=Petrol]' : ['ULP', 'PULP' ]
 'Vehicle.fuelType[engineType=Diesel]' : ['BIO', 'NORMAL' ]
```

这显示了如何通过根据其他字段的值来进行选择。请注意，一旦你选择了 **engineType**，**fuelType** 的选项列表就被决定里。

在程序里加载枚举

在某些情况下，我们胡会希望从外部数据源（如关系型数据库）加载自己的枚举数据。为此，你可以实现一个返回 Map 类型的类。这个 Map 的键是一个字符串（就是上面显示的 **Fact.field** 名称）。它的值是一个字符串的 **java.util.List**。

```
public class SampleDataSource2 {

  public Map<String>, List<String>> loadData() {
    Map data = new HashMap();

    List d = new ArrayList();
    d.add("value1");
    d.add("value2");
    data.put("Fact.field", d);

    return data;
 }

}
```

在 BRMS 的枚举里，输入下列内容：

```
=(new SampleDataSource2()).loadData()
```

"=" 符号表示通过执行你的代码加载数据。

更多高级的枚举

在上面的例子里，列表里的值会“先”被计算。对于相对静态或小型数据来说，这是非常足够的。然而，假设我们有一个国家列表，而每个国家都有一个州的列表，而每个州又有自己的地区列表，地区则有街道列表等等。此时你就会发现这需要大量的数据，这么大的数据是无法被一次加载的。所以，可以根据所选的国家来加载列表。

这种情况可以下列方式解决：

```
'Fact.field[dependentField1, dependentField2]' : '(new
com.yourco.DataHelper()).getListOfAges("@{dependentField1}", "@{dependentField2}")'
```

请注意，只有需要的字段才被指定。而且，在右边的 ":" 符号中，表达式是用引号括起的。这个表达式只在需要时才会被估值。此时，它将替代指定的字段的值。这意味着你可以使用图形化用户界面的字段值来驱动数据库查询，然后再向下查询数据，诸如此类。当加载了规则的下拉列表后，列表将根据这些字段进行刷新。"**depenentField1**" 和 "**dependentField2**" 是 "Fact" 类型上的字段名称。它们用于计算作为 "字段“值显示在下拉列表里的值列表。

BRMS 里的每个资产和软件包都有一套*状态标记（status flag）*。这些标记的值可以在 BRMS 的“管理”章节找到。在这里，你可以添加自己的状态名称。和类别相似，状态也不会以任何方式影响执行。它们只是提供信息而已。

注意

不像类别，资产在某个时刻只有一个状态。

状态的使用完全是可选的。你可以采用它们或类别来管理资产的生命周期。

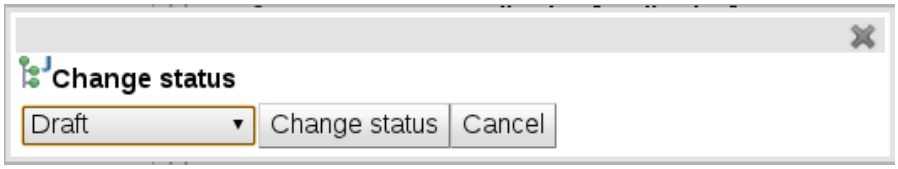


图 **4.20.** 资产状态

上面的图表描述单个资产的状态变化。这种变化会立即发生；它不需要单独地保存。

你也可以修改整个软件包的状态。这会设置软件包自身和属于它的所有资产的状态标记为同一个值。

4.5. 软件包管理

在 BRMS 里，你有一个"*知识库（Knowledge Base）*"，它包含一个或多个"*知识软件包（Knowledge Package）*"。在用户界面里，知识软件包通常被简单地称为 "*软件包*."。

警告

配置知识软件包通常只进行一次，它应该由具有开发规则和模型经验的人员来完成。很少有人需要配置知识软件包，只要配置完毕，它可以按需要多次进行复制。软件包配置肯定是一个需要规则开发经验的技术性任务。

所有的资产都位于 BRMS 的“软件包”里。软件包就像一个子目录（它也可作为“命名空间”）。它和规则资产所在的主目录等同。特别是规则需要知道 fact model 和命名空间的性质。

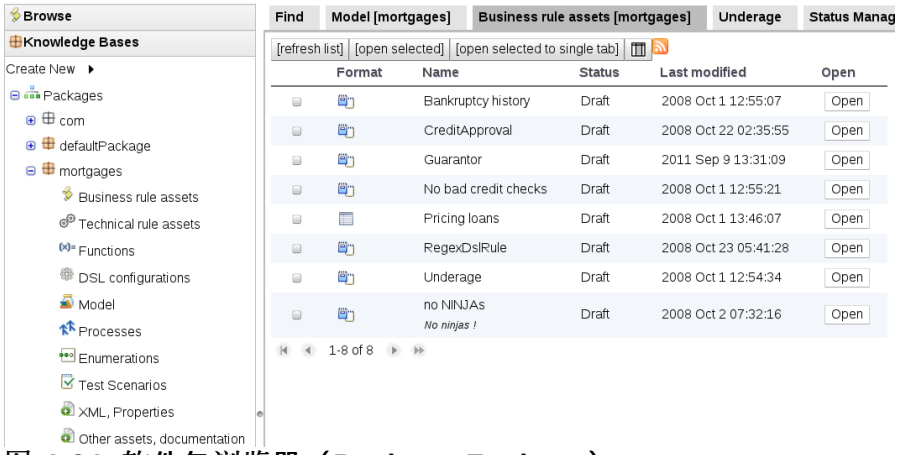


图 **4.21.** 软件包浏览器（**Package Explorer**）

上面的图片展示了*软件包浏览器（Package Explorer）*。点击资产类型将显示对应的一个列表（对于具有数千个规则的软件包，这需要几秒钟才会显示出来，因此使用类别来加速浏览很重要）。

总而言之，虽然规则（资产）可以出现多个类别里，它们只位于一个软件包内。如果你将 BRMS 当作一个文件系统，则每个软件包都是一个目录，而其中的资产就是文件列表。

软件包管理功能允许你查看知识软件包列表并"扩展"为每个"类型"的小型列表（资产可能很多，所以其中一些进行了分组）：

资产类型：

- 商业资产（Business Asset）：显示所有"商业规则"类型的列表，它包括决策表（Decision Table）、商业规则等。
- 技术资产（Technical Asset）：显示被当作技术性规则的资产列表（如 DRL 规则、数据枚举以及规则流）。
- 功能（Function）：在 BRMS 里，你也可以定义功能（这是可选的）。
- DSL：Domain Specific Language 也可作为资产存储。如果存在（通常只有一个），他们将和合适的编辑器 GUI 一起使用。
- 模型（Model）：一个软件包需要至少一个模型。它用于规则。

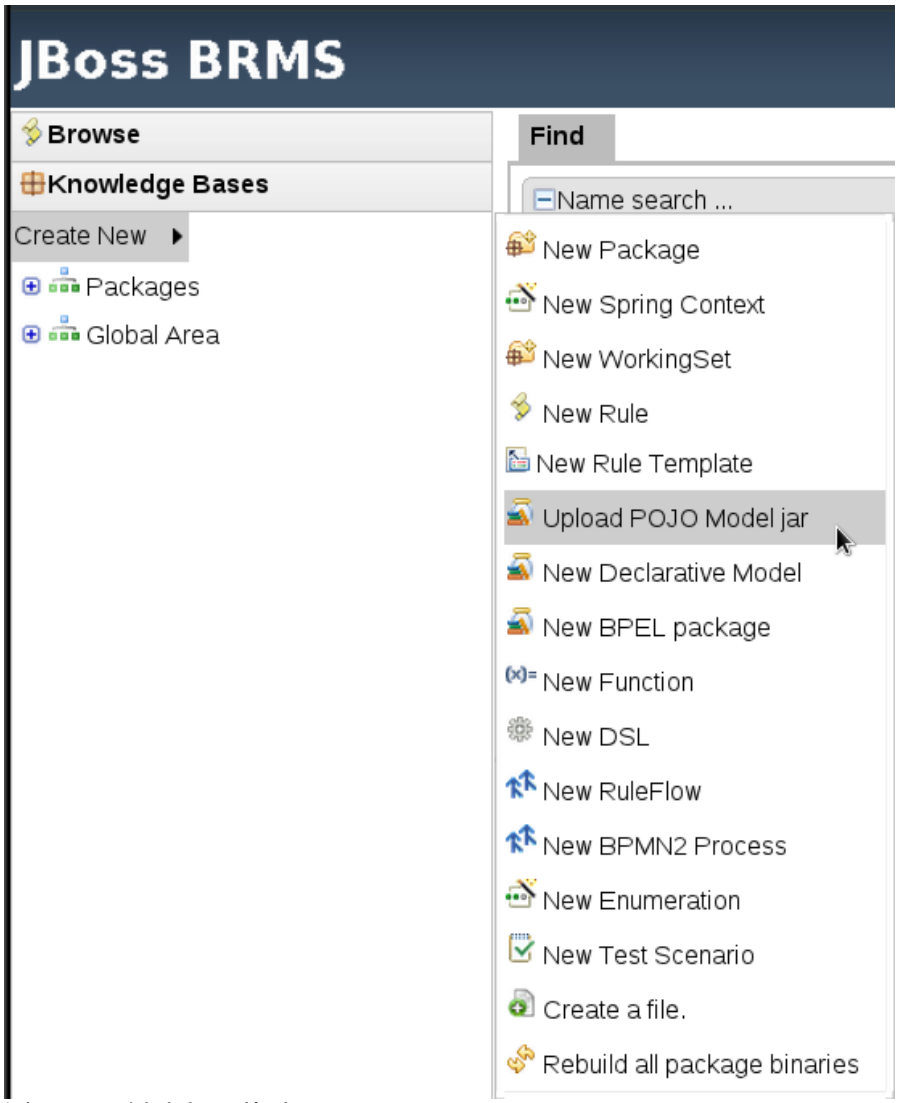


图 4.22. 创建新的资产

你可以使用软件包浏览器（Package Explorer）创建新的规则或资产。有些资产只能通过软件包浏览器创建。图 4.22 “创建新的资产” 展示了启动该功能的“向导”的图标。

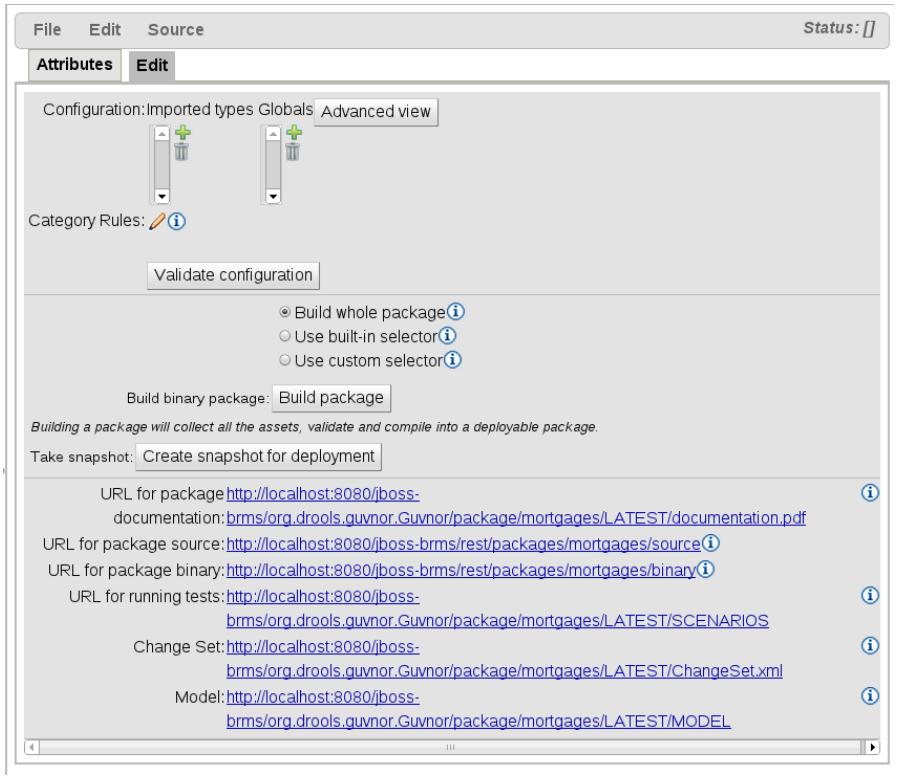


图 **4.23.** 软件包配置

你需要做的最重要的事情是配置软件包。这个任务涉及导入规则使用的类并定义全局变量两个方面。进行了修改后，你需要保存它。此时，软件包就已经完成配置并准备好进行构建了。例如，你可以添加一个具有类 **`com.something.Hello`** 的模型。然后你将在软件包配置里添加 **`import com.something.Hello`** 并保存修改。

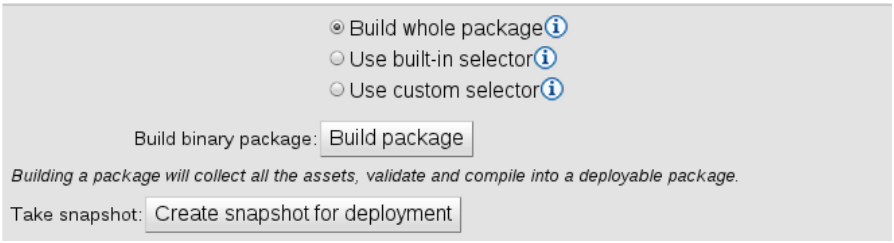


图 **4.24.** 构建软件包

在导入软件包的类并定义了全局变量后，你可以构建软件包。软件包包含的资产越多，构建需要的时间越长。构建成功后，你将可以为部署创建一个“快照”。此时，你也可以查看软件包构建时生成的“drl”。

4.5.1. 导入 drl 软件包

你也可以通过导入现有的 "drl" 文件来创建软件包。当你选择创建新的软件包时，你可以选择上载一个 **`.drl`** 文件。BRMS 将试图解析这个 drl 并自动创建一个软件包。它里面的规则将作为单个资产保存（仍然是 drl 的文本内容）。请注意，要实际上构建软件包，你需要上载一个合适的模型（以 **`.jar`** 形式）以供检验。这是一个单独的步骤。

4.6. 版本管理

在 BRMS 里，整个知识软件包和单个资产都带*版本*，单每个的机制都不太一样。单个资产在源码控制系统（**Subversion**）里以文件版本的形式保存。然而，资产软件包通过快照方式根据需要（“On Demand”）划分版本。这个快照用于部署。下一节我们讨论部署管理和快照的更多细节。

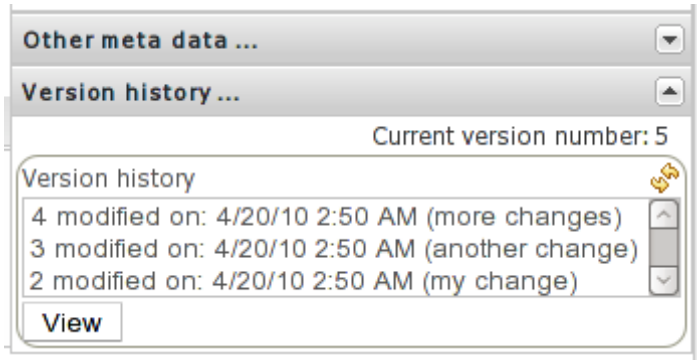


图 **4.25.** 资产**版本**

每次对资产进行修改后，它都在版本历史记录创建一个新的条目。这为你提供了无限制的“Undo”功能。你可以查看单个资产的历史（如上面的列表所示），然后按照时间进行恢复。

4.7. 部署管理

URL 是围绕如何提供构建的知识软件包的过程的中心。BRMS 通过 URL 提供知识软件包以供知识代理（Knowledge Agent）下载和使用。这些 URL 采用这种格式：http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/package/<packageName>/<packageVersion>

<packageName> 是你给软件包取的名字。<packageVersion> 是快照的名称或 "**LATEST**"（如果它是“**LATEST**”，那么它将是从主软件包里构建的最新版本而非快照）。你可以在代理里使用它或者将其粘贴到浏览器地址栏里作为文件下载。

关于如何在应用程序里使用这些 URL 和二进制下载、以及如何在运行时（On the fly）更新规则的详情，请参考 第 8.1 节 “知识代理（Knowledge Agent）”。

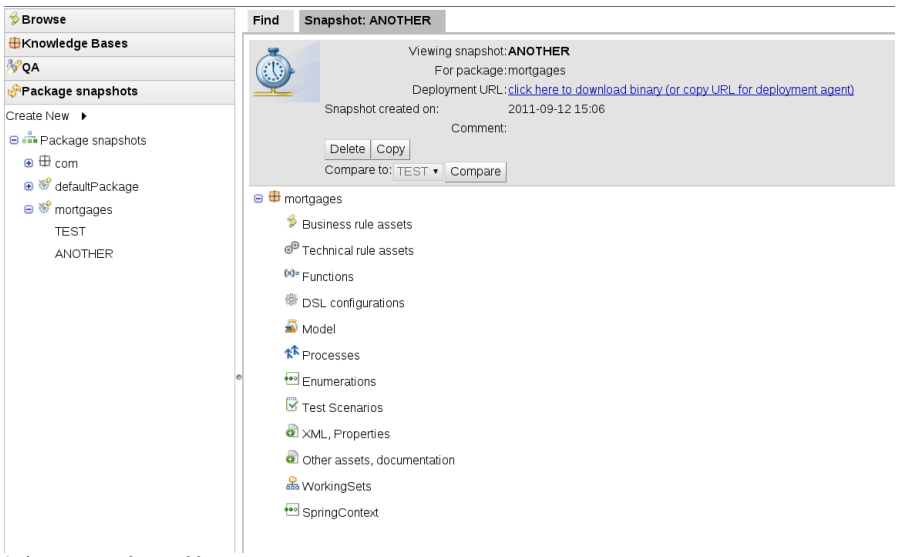


图 **4.26.** 部署快照

上面的映像展示了 *Deployment Snapshots* 视图。左侧有一个知识软件包列表。点击某个软件包，它将为你展示它的所有快照（如果有的话）。你可以复制、删除或查看快照。每个快照都可以下载或通过 URL 访问。下载后你就可以部署它。

4.8. 浏览库和查找规则

查看库的两个主要途径是使用用户驱动的类别（也称为 tagging）和*软件包浏览器（Package Explorer）*视图。

分类视图（Category View）为你提供了浏览与你的机构相关的规则的途径。

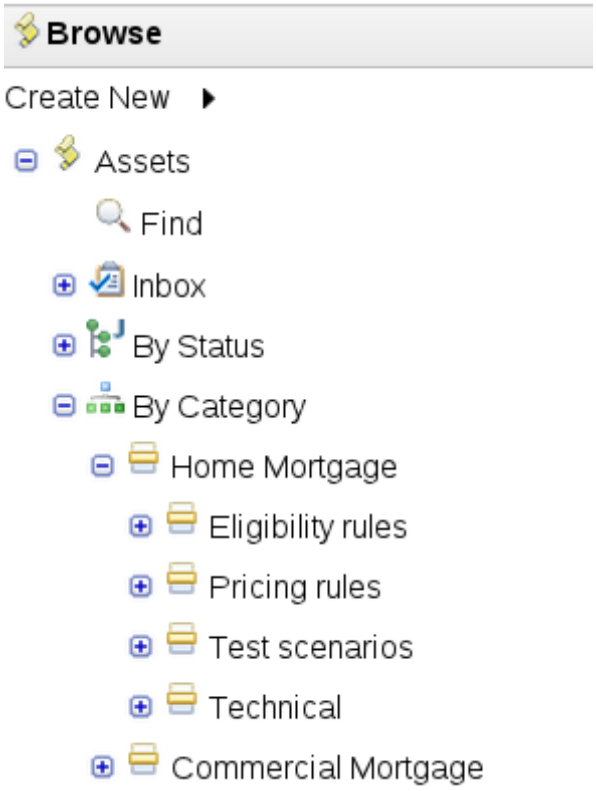


图 4.27. 分类视图（Category View）

上面的图表显示了分类功能。如果可能的话，将每个类别里的规则限制在几十个以下。

其他的视图使用软件包浏览器（Package Explorer）。这个视图以反映数据库里实际存储情况的方式显示资产。它也按照类型或格式将规则分开到知识软件包里。

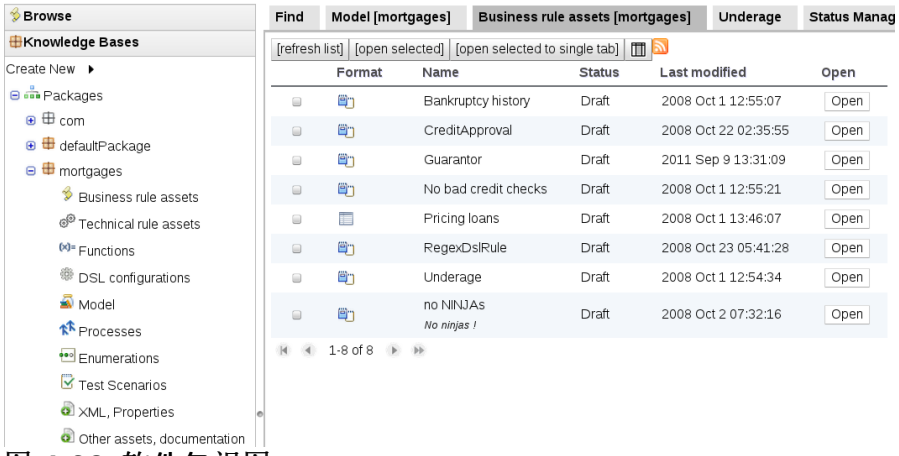

Browse
Knowledge Bases
Create New
Packages
com
defaultPackage
mortgages
Business rule assets
Technical rule assets
Functions
DSL configurations
Model
Processes
Enumerations
Test Scenarios
XML, Properties
Other assets, documentation
Find
Model [mortgages]
Business rule assets [mortgages]
Underage
Status Manag
[refresh list]
[open selected]
[open selected to single tab]
Format
Name
Status
Last modified
Open
Bankruptcy history
Draft
2008 Oct 1 12:55:07
Open
CreditApproval
Draft
2008 Oct 22 02:35:55
Open
Guarantor
Draft
2011 Sep 9 13:31:09
Open
No bad credit checks
Draft
2008 Oct 1 12:55:21
Open
Pricing loans
Draft
2008 Oct 1 13:46:07
Open
RegexDslRule
Draft
2008 Oct 23 05:41:28
Open
Underage
Draft
2008 Oct 1 12:54:34
Open
no NINJAs
No ninjas !
Draft
2008 Oct 2 07:32:16
Open
1-8 of 8

图 4.28. 软件包视图

第 5 章 事实模型（对象模型）

驱动基于规则的应用程序的规则需要*事实模型（Fact Model）*。事实模型通常和应用程序的*域模型（domain model）*重叠，但它应该从中分离，这从长远来说使得规则易于管理。

有两种定义事实模型的方法：

- 上传包含应用程序和规则使用的 Java 类的 **JAR** 文件。
- 在 BRMS 里声明的模型可以作为 **KnowledgeBase** 导出并用在 Java 代码里。

5.1. 全局区域（Global Area）

当创建事实模型时，它们可以导入将使用它们的特定软件包里，或者导入到**全局区域（Global Area）**里。请注意 **Global Area** 里的资产并不是对全局可用的，它们仍然必须导入到使用它们的软件包里。对于还未使用的资产，**Global Area** 应该用作一个存储位置，而对于用在多个软件包里的资产，它应该用作一个中心位置。

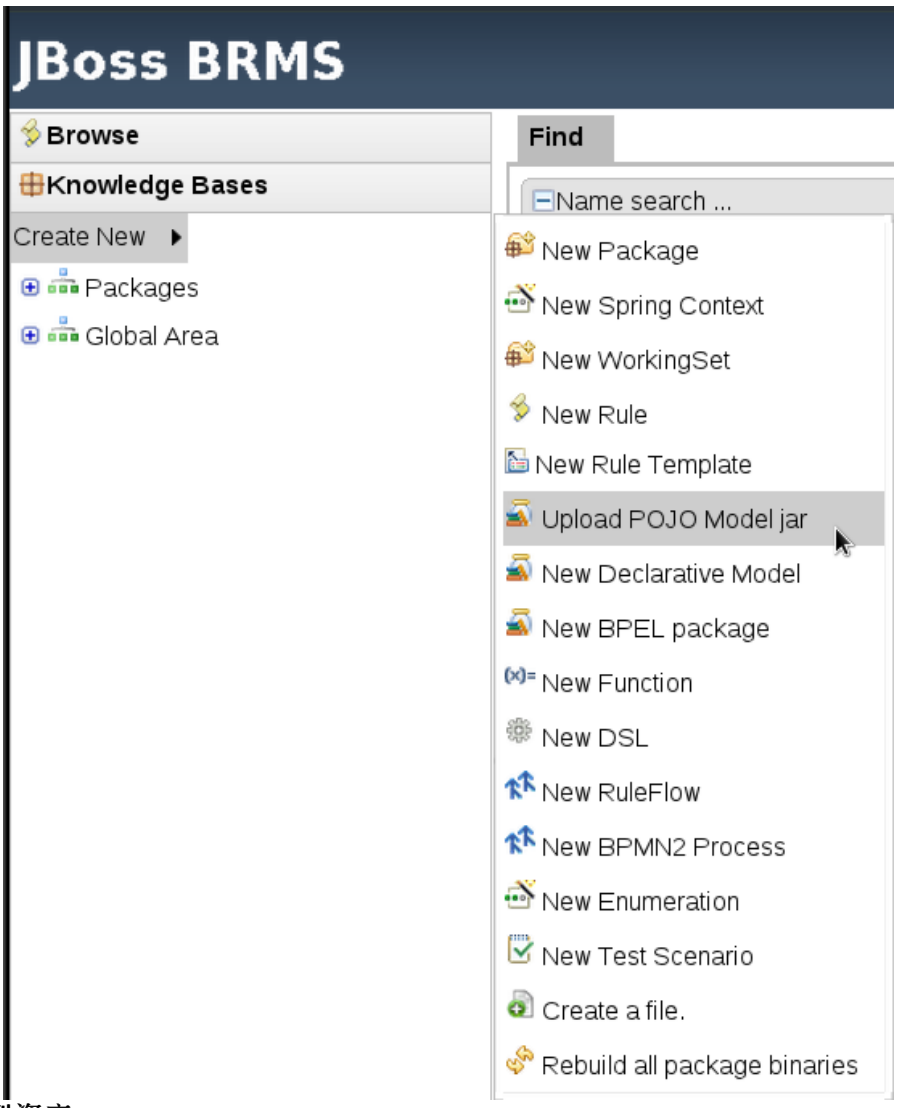
注意

当编辑位于软件包以及**全局区域（Global Area）**里的资产时需要很小心。在 **Global Area** 里编辑的资产将需要再次导入到使用它们的软件包里，而之前导入的资产将被删除。

5.2. JAR 模型

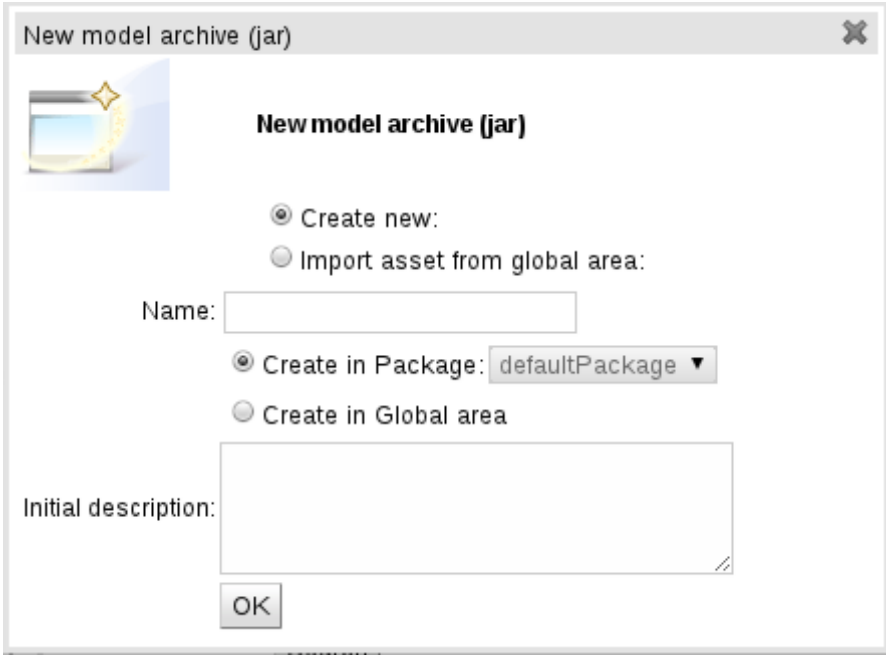
过程 5.1. 创建 JAR 模型

1. **打开 New model archive (jar) 菜单**
   从 **Knowledge Bases** 菜单里，选择 **Create New**，然后选择 **Upload POJO Model JAR**。



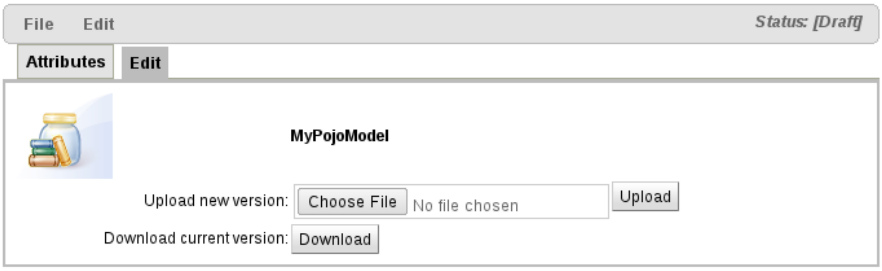
2. 创建 **JAR** 模型资产

   输入 JAR 模型的名字、类别和描述。选择模型所在的软件包或指定它应该添加到 **`Global Area`** 里。输入完毕后点击 **OK**。



3. 上传 **JAR** 到资产里

上传包含定义为常规 JAR 文件里的 Java 类和软件包的 JAR。



5.3. 声明式模型（Declarative Model）

使用声明式模型（Declarative Model）有如下好处：

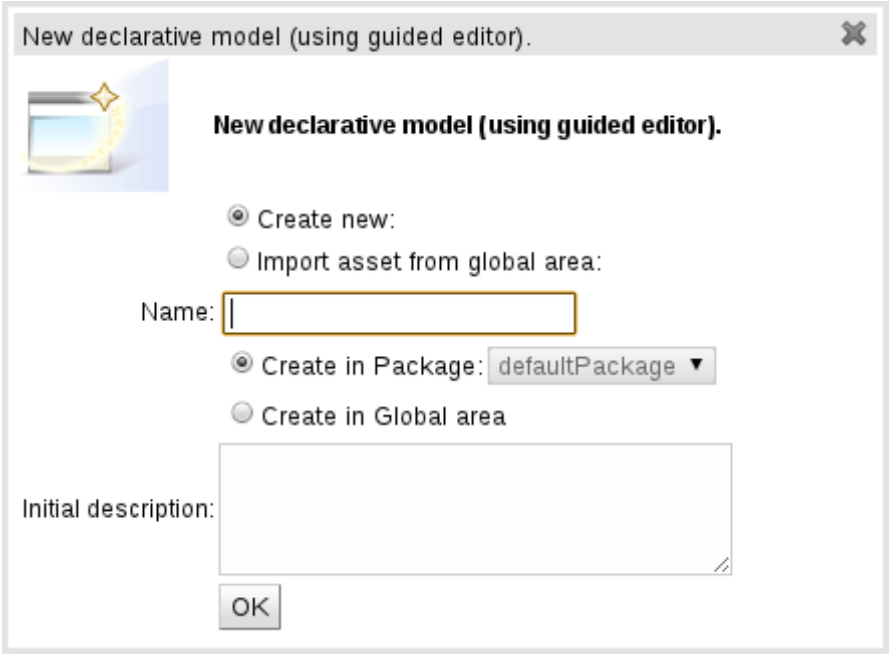
- 它强调模型属于知识库，标注应用程序。
- 这个模型可以有独立于应用程序的生命周期。
- 规则专有的注解可以丰富 Java 类型。
- JAR 文件必须在规则和使用它的应用程序之间保持同步，然而，声明式模型不需要保持同步。

声明式模型可以是：

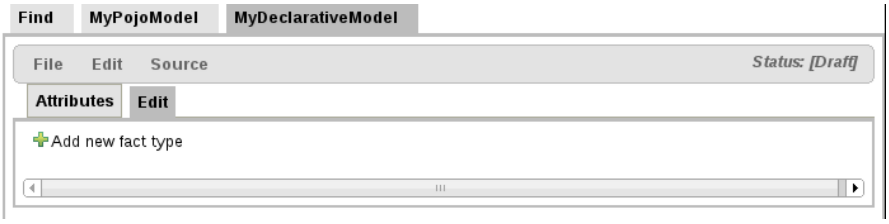
- 你的规则里使用的整个事实模型的独立定义。
- 支持 Java POJO 模型的补充性事实定义。

过程 **5.2.** 创建声明式模型

1. **打开 `New Declarative Model` 菜单**
   从 **`Knowledge Bases`** 菜单，选择 **`Create New`**，然后选择 **`New Declarative Model`**。
2. **创建一个新的声明式模型**
   指定新模型的名称。选择创建模型的软件包或者指定它是否应该添加到 **`Global Area`**。输入完毕后点击 **`OK`**。



3. **定义模型**
   点击 **`Add new fact type`** 并在弹出菜单的 **`name`** 字段输入事实的名称。

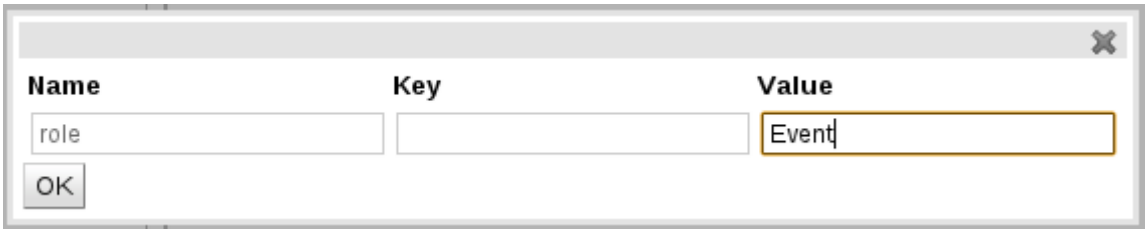


4. **添加事实字段**

   通过选择 **`Add field`** 按钮并在弹出菜单里输入信息来创建事实字段（Fact Field），

5. **添加注解**

   选择 **`Add annotation`** 按钮来创建事实注解。注解的**`Name`** 和 **`Value`** 字段都是强制的，但 **`Key`** 字段是可选的。如果没有指定 **`Key`** 值，它将被分配 ***`Value`*** 的一个缺省值。



5.3.1. 在 Java 里消费声明式模型

声明式类型是在知识库编译时生成的，因此应用程序只能在运行时访问它们。所以，应用程序不能直接引用这些类。

声明式类型可以象普通的事实对象一样使用，但你创建它们的方式会不一样（因为它们没有位于应用程序的 classpath 上）。要创建这些对象，它们可通过 KnowledgeBase 实例来进行。

例 5.1. 通过 API 处理声明的事实类型

```
// get a reference to a knowledge base with a declared type:
KnowledgeBase kbase = ...

// get the declared FactType
FactType personType = kbase.getFactType( "org.drools.examples",
                                         "Person" );

// handle the type as necessary:
// create instances:
Object bob = personType.newInstance();

// set attributes values
personType.set( bob,
                "name",
                "Bob" );
personType.set( bob,
                "age",
                42 );

// insert fact into a session
StatefulKnowledgeSession ksession = ...
ksession.insert( bob );
ksession.fireAllRules();

// read attributess
String name = personType.get( bob, "name" );
int age = personType.get( bob, "age" );
```

注意

声明的类型的命名空间是它所在的软件包的命名空间（在上面的例子里是 **`org.drools.examples`**）。

第 6 章 工作集（Working Set）

注意

工作集（Working set）是 BRMS 5.2 里的一个实验性的功能。

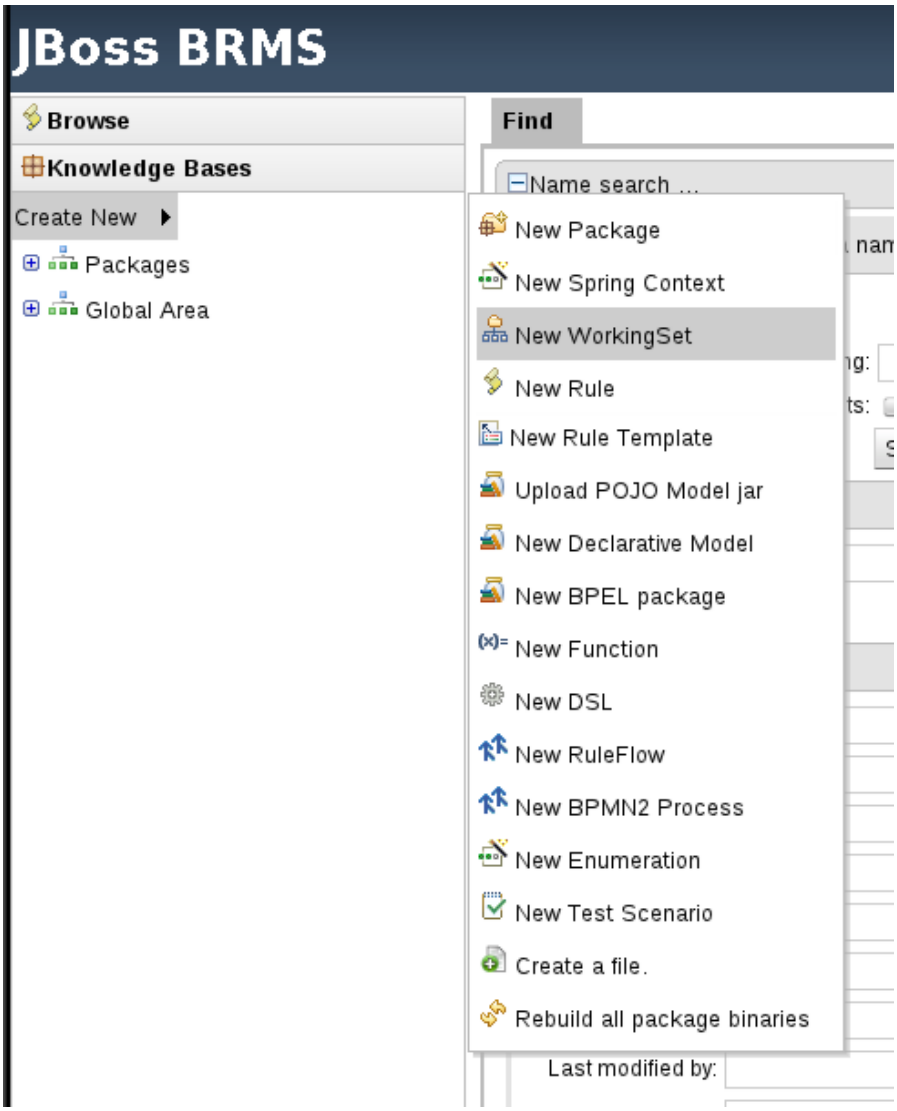
工作集（Working set）将事实（fact）进行分组并定义通用的约束。工作集也可以在对规则授权时限制 Guided Editor 里可见的规则。

工作集必须从 Guided Editor 里手动激活。

过程 6.1. 创建一个新的工作集

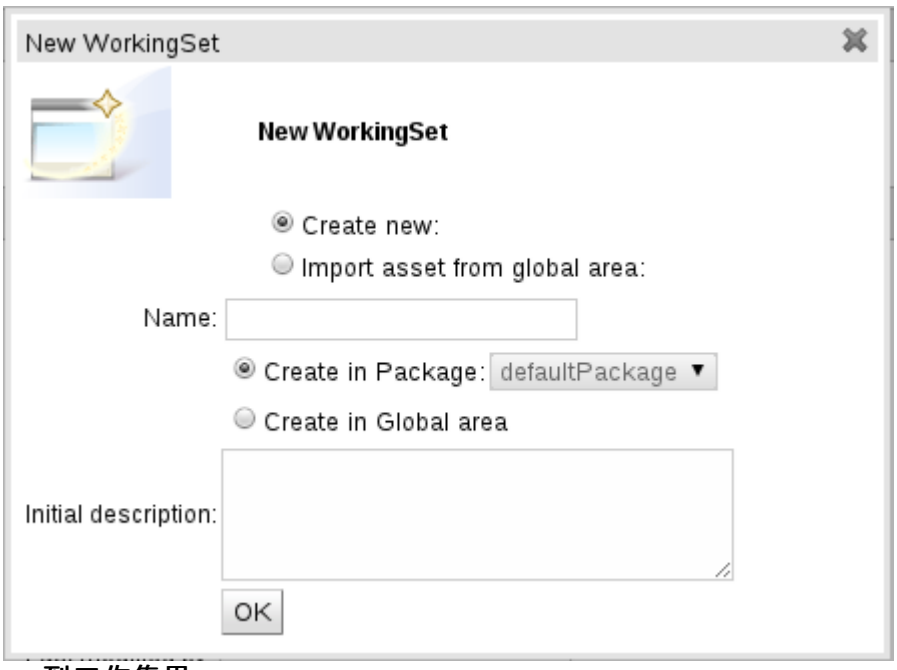
1. **打开 `New WorkingSet` 对话框**

   从 **`Knowledge Bases`** 菜单，选择 **`Create New`**，然后选择 **`New WorkingSet`**。



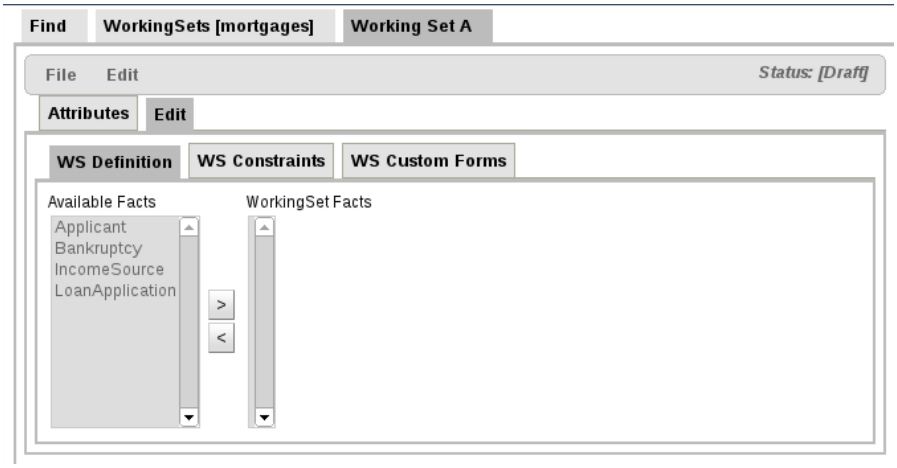
2. **创建新的工作集**

   输入工作集的名称和描述。选择创建模型的软件包或者指定它是否应该添加到 **`Global Area`**。输入完毕后点击 **`OK`**。



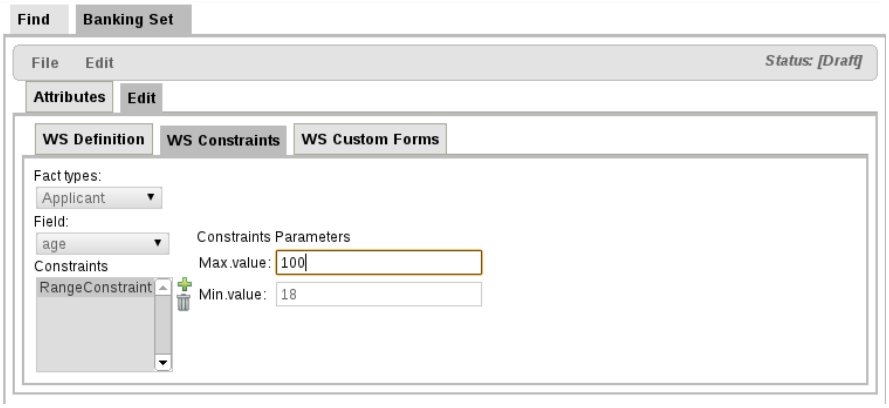
3. **添加 Fact Type 到工作集里**

   将 fact type 从左边列表移至右边可以添加 fact type 到工作集里。右边列表的 fact type 将对工作集可用。



4. **在工作集里的 Fact Type 里添加约束**

   从下列菜单里选择 **`Fact type`**，选择 **`Field`** 值并添加所需的约束。



6.1. 检验 Field 约束

你可以两种方式检验 Field 约束：

- On Demand 检验
- 实时检验

On Demand 检验是从 Guided Editor 工具栏里选择 **verify** 来执行的。检验结果包含一个检验报告。

实时检验会实时检查违反字段约束的情况，并在发生时进行标记。

要启用实时检验，选择 **Administration**，然后选择 **Rules Verification** 并在 **Enable** 复选框上打勾。

注意

这是 一个实验性的功能，在缺省情况下它是禁用的。

第 7 章 商业用户的视角

对于商业用户，最合适的规则格式是源于使用 Guided Editor、决策表和 DSL 规则的格式。你也可以在 Guided Editor 里使用一些 DSL 表达式（它提供用户可以输入值的“表单”）。

你也可以使用类别来隔离非技术型用户和规则/资产。只有以及分配了类别的资产将出现在“类别”视图里。

BRMS 平台的初始配置将由开发人员/技术人员来承担。这些人员将为所有的规则设置基础配置。它们也可以创建“模板”，也就是可以复制的规则（通常驻留在一个“傀儡”软件包里，位于“template"类别里）。模板是很有用处的。

部署也不应该由非技术人员来实施。如本*指南*里前面所提到的，部署是通过“Packages"系统来完成的。

第 8 章 集成规则和应用程序

到目前为止，本指南已经讨论了规则的管理。现在，你将学习在应用程序里使用规则。本节涵盖知识代理（Knowledge Agent）部署组件的用法，它可以自动化这个过程的一大部分。

8.1. 知识代理（Knowledge Agent）

重要

`jboss-brms-engine.zip` 里包括的 **`README_DEPENDENCIES.txt`** 包含了每个组件的依赖关系的细节。

知识代理（Knowledge Agent）是一个嵌入在 JBoss Rules 5.0 API 里的组件。使用知识代理不许要其他额外的组件。如果你在是以 JBoss 企业级 BRMS 平台，应用程序只需要在其 classpath 里包含 **`drools-core`** 依赖关系，也就是 **`drools`** 和 **`mvel`** JAR 文件。这里没有其他的规则专有的依赖关系。

这里也有一个名为 drools-ant 的 ant 任务，你可以用 ant 脚本构建规则并生成 .pkg 文件。最常见的情形是当规则在 IDE（JBoss Developer Studio）里编辑而无需使用 BRMS UI。

在软件包里构建了 BRMS 平台里的软件包后，你旧可以在目标应用程序里使用代理了。

下面的例子构造了一个代理，它将根据路径字符串里指定文件构建一个新的知识库。它每隔60秒（缺省设置）将轮询这些文件是否有了更新。如果找到了新文件，它将构造一个新的知识库。如果指定的资源是一个目录，它里面的内容也将被扫描。

```
KnowledgeAgent kagent = KnowledgeAgentFactory.newKnowledgeAgent( "MyAgent" );
kagent.applyChangeSet( ResourceFactory.newUrlResource( url ) );
KnowledgeBase kbase = kagent.getKnowledgeBase();
```

知识代理能接受的配置允许其中一些缺省值被修改。其中的一个例子是属性 "drools.agent.scanDirectories"，在缺省情况下任何指定的目录都将被扫描以检测是否有新的内容，你可以禁用这个功能。

```
KnowledgeBase kbase = KnowledgeBaseFactory.newKnowledgeBase();

KnowledgeAgentConfiguration kaconf =
KnowledgeAgentFactory.newKnowledgeAgentConfiguration();
kaconf.setProperty( "drools.agent.scanDirectories", "false" ); // we don't scan
directories, only files

KnowledgeAgent kagent = KnowledgeAgentFactory.newKnowledgeAgent( "test agent",
kaconf );
// resource to the change-set xml for the resources to add
kagent.applyChangeSet( ResourceFactory.newUrlResource( url ) );
```

这里是一个 **`change-set.xml`** 示例。

```
<change-set xmlns='http://drools.org/drools-5.0/change-set'";
    xmlns:xs='http://www.w3.org/2001/XMLSchema-instance'
    xs:schemaLocation='http://drools.org/drools-5.0/change-set drools-change-set-
5.0.xsd' >
    <add>
        <resource source='http://localhost:9000/TEST.pkg' type='PKG' />
    </add>
</change-set>
```

资源扫描在缺省情况下是启用的。这是一个服务且必须被启动，通知服务也是一样。这可以通过 ResourceFactory 来完成。

```
ResourceFactory.getResourceChangeNotifierService().start();
ResourceFactory.getResourceChangeScannerService().start();
```

下面是 BRMS UI 的部署界面，它提供了 URL 和软件包的下载地址。

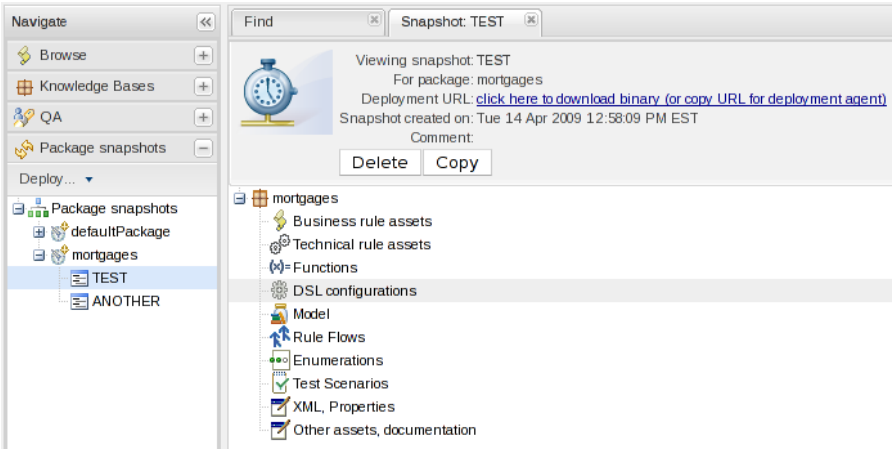


图 **8.1. 快照部署**

你可以看到 **`Package URI`**。这是一个你需要包含在 **`change-set.xml`** 文件里的 URL，以指定你需要这个软件包。它指定了一个额外的版本，在这里是一个快照。每个快照都有自己的 URL。如果你需要最新的版本，你可用 **`LATEST`** 替换 **`NewSnapshot`**。

你也可以从这里下载一个软件包文件（**`PKG`**）。将这个文件放入一个目录里并使用知识代理的 **`file`** 或 **`dir`** 功能。这将自动联系 JBoss 企业版 BRMS 平台服务器以获取在其他情形下可能不需要的更新。

8.2. 手动部署

只有将部署集成到自己的机制里的高级用户才需要阅读本节。通常来说你应该使用知识代理。

对于不希望使用知识代理自动部署的用户来说，手动部署是相当简单的。JBoss 企业版 BRMS 平台生成的二进制软件包会序列化为 Package 对象。你可以解序列化并将它们添加到任何知识库里。

在 JBoss 企业版 BRMS 平台里，二进制软件包是由最新的软件包（一旦软件包被成功检验并构建）或部署快照提供的。JBoss 企业版 BRMS 平台开放的 URL 提供了使用 HTTP 的二进制软件包。你也可以执行 "HEAD" 命令来获得软件包最后一次被更新的时间。

8.3. WebDAV

"back-end" 库也可以通过 **`WebDav`** 访问。WebDav 是一个基于 HTTP 的文件系统应用程序编程接口。大多

数的操作系统，包括 Windows、Apple MacOS X 和 Linux 都提供访问 WebDAV 共享目录的集成支持。请参考操作系统供应商提供的配置说明文档。对于多数平台而言，也有许多第三方的 WebDAV 客户端工具。

用 WebDAV 访问库的 URL 和 web 接口的

```
http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav/
```

象平常一样这也需要验证。WebDAV 提供了一个软件包和快照目录。**Snapshots** 是一个只读目录，它基本上是已创建的知识软件包的快照的视图。packages 目录包含作为目录的知识软件包列表，而目录则包含作为文件的单个资产。

8.3.1. WebDav 和特殊字符

BRMS 支持规则名称里含有 UTF-8 字符。然而，当规则通过 WebDav 复制，多字节字符将解码为 ISO-8859-1。

红帽不推荐使用在规则名称里使用特殊字符，然而如果使用了特殊字符，Web 连接器必须进行修改以支持 Unicode。

要添加对 Unicode 的支持，请完成下列步骤。

过程 8.1. 添加对 Unicode 的支持

1. 停止应用服务器。
2. 打开 **server.xml** 文件。这个文件位于 **jbossweb.sar** 目录里。
3. 在 web 连接器里添加 ***URIEncoding="UTF-8"***。例如，对于 HTTP，这个代码应该是：

   ```
   <Connector protocol="HTTP/1.1" port="8080" address="${jboss.bind.address}"
   connectionTimeout="20000" redirectPort="8443" URIEncoding="UTF-8" />
   ```

4. 启动应用服务器。

8.4. URL

本节里提及的关于知识代理的软件包部署 URL 也有一些其他特征。

要获得这个软件包而不是二进制软件包的 DRL，附加 **.drl** 到 URL 的结尾，如 /package/testPDSGetPackage/LATEST.drl，附加 **/assetName.drl**，它将显示该条目的 DRL，即使它不是一个 DRL 文件，如 **/package/testPDSGetPackage/LATEST/SomeFile.drl**。

第 9 章 收件箱和评论

重要

只存在于 JBoss 企业级 BRMS 平台 5.1.0 和之后的版本。

通过 artifact 文档的额外源以及用于 artifact 变动的通知系统，评论和收件箱功能帮助用户管理 artifact 的变动。

9.1. 评论（Comment）

任何用户都可以在 artifact 的 documentation box 下的评论部分添加评论。每条评论都是和进行评论的用户的标识符、评论的日期和时间一起记录的。管理员可以清除某个 artifact 的评论，但其他用户只能附加评论。

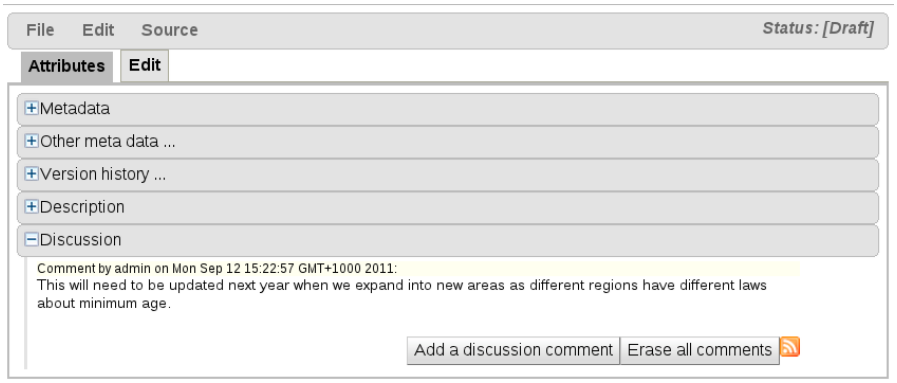


图 **9.1.** 评论（**Comment**）

9.2. 收件箱（Inbox）

“Inbox“位于导航面板的 Browse 区域。它提供对用户最近操作的 artifact 的快速访问以及对用户过去曾编辑的 artifact 的变动的通知。

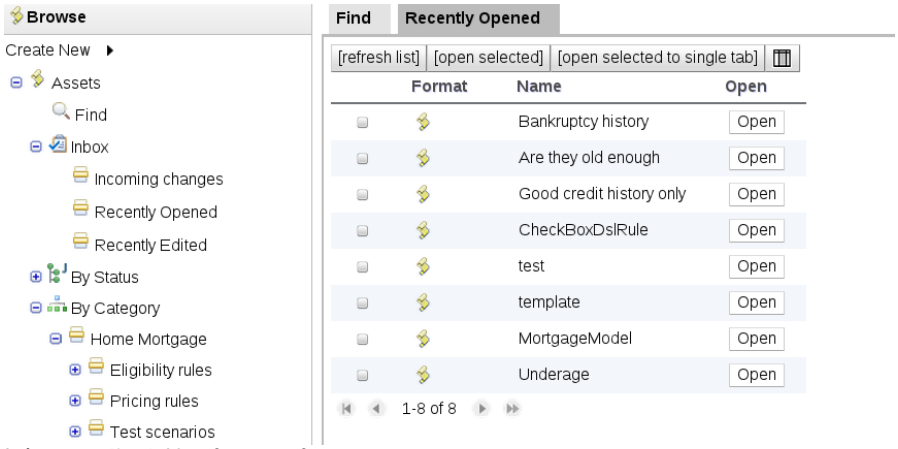


图 **9.2.** 收件箱（**Inbox**）

收件箱包含三个“子收件箱”：

Incoming changes

这里会列出对用户之前编辑或评论的 artifact 的修改。

Recently opened

这里会列出用户最近打开的 100 个 artifact 以供快速访问。

Recently edited

提供用户编辑的最近 100 个 artifact 以供快速访问。

第 10 章 JBoss Developer Studio 集成

Eclipse Guvnor 工具（EGT）为开发人员提供了使用 JBoss Developer Studio 4 读、写、添加和从 JBoss 企业级 BRMS 平台服务器删除资产的界面。EGT 也为开发人员提供了一个和传统的源码控制系统（如 Subversion）类似的界面。

重要

JBoss 只对 JBoss Developer Studio 4 的这个功能提供支持。

BRMS 平台库和 EGT 的目的不是要替换源码控制系统，它为开发人员提供了一种方便的访问方法。

重要

Guvnor 是构建 JBoss 企业版 BRMS 平台的开源项目。Eclipse Guvnor Tools 就是基于此点而开发的。EGT 在任何地方提及的 “Guvnor” 都是指的 "BRMS 平台"。为了避免混淆，本章以后的部分我们将统称 “Guvnor”。

10.1. 功能概述

EGT 包含两个视图 - 库浏览器（Repository Explorer）和版本历史（Version History）- 它们是和 Guvnor 库交互的中心点。

"Guvnor Repository Exploring" 视角是我们建议的格式。它可以通过 **Open Perspective** 对话框（**Window** -> **Open Perspective** -> **Other...**）来访问。

左侧是 **Guvnor Repository Explorer** 和 **Eclipse Properties** 视图，**Guvnor Resource History** 视图位于底部，而 **Eclipse Resource Navigator** 位于右侧。**Guvnor Repository Explorer** 为访问 Guvnor 库资源提供了可浏览的树型格式。**Guvnor Resource History** 视图显示了库里可用资源的修订记录。

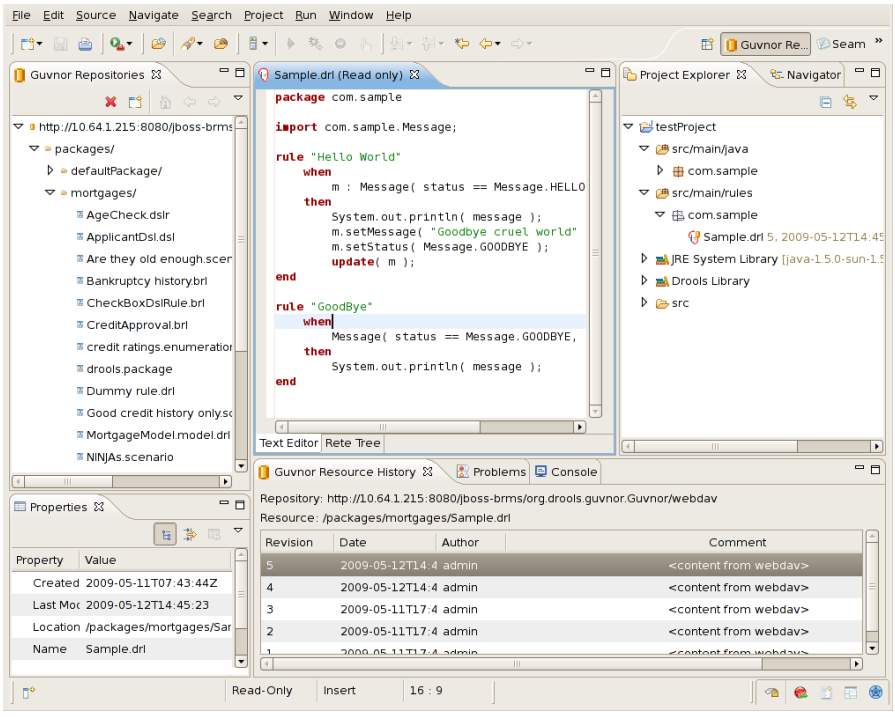


图 **10.1. "Guvnor Repository Exploring"** 视角

10.2. Guvnor 连接向导

打开 Guvnor 视角后，第一个任务是连接连接到 Guvnor 库。这是由 Guvnor 连接向导处理的。这个向导出现在 EGT 里的多个地方（参见下面的详情），但在本节我们只介绍两个最基本的入口点。Guvnor 连接向导可以用 Eclipse 菜单启动：File -> New -> Other -> Guvnor -> Guvnor repository location 或通过 Guvnor 浏览器的下拉菜单：

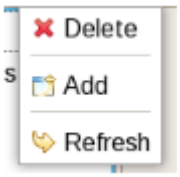


图 **10.2.** 连接向导

或者使用菜单按钮：

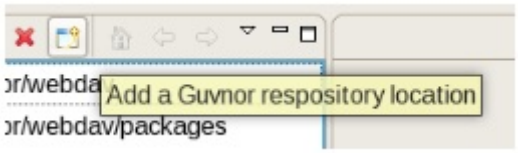


图 **10.3.** 连接向导

选择其中一个都将启动 Guvnor 连接向导：

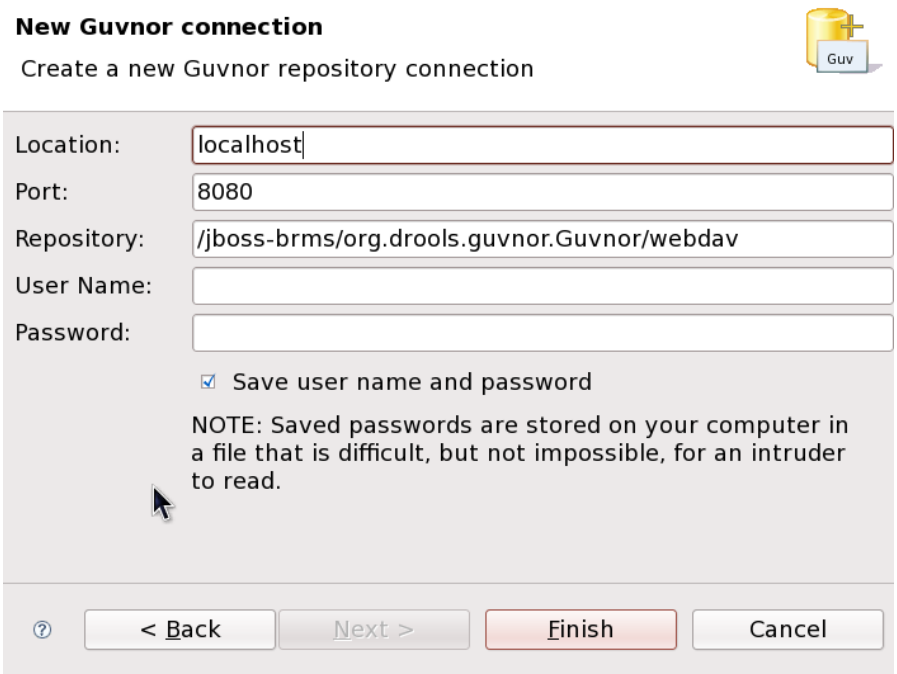


图 **10.4.** 连接向导

缺省值会出现在 Location、Port 和 Repository 字段。（关于如何修改这些缺省值，请参考下面的 "Guvnor 首选项" 章节。）

当然，你可以在对应的文本框里输入来编辑这些字段。将典型的 Guvnor 库 URL 拖放或粘贴到 Location 字段，如：**`http://localhost:8080/jboss-brms/org.drools.guvnor.Guvnor/webdav`**。

URL 里的结果也会被粘贴到对应的字段。选择 ”Save user name and password“ 可以将验证信息（用户名和密码）存储到 Eclipse 工作台的 key-ring 文件里。如果验证信息没有存储在 key-ring 文件里，EGT 将使用会话验证。这表示提供的凭证只能用于 Eclipse 工作台实例的生命周期内。

如果验证信息没有存储在 key-ring 文件里或者它是无效的，当EGT 需要访问 Guvnor 库时它将提示你：

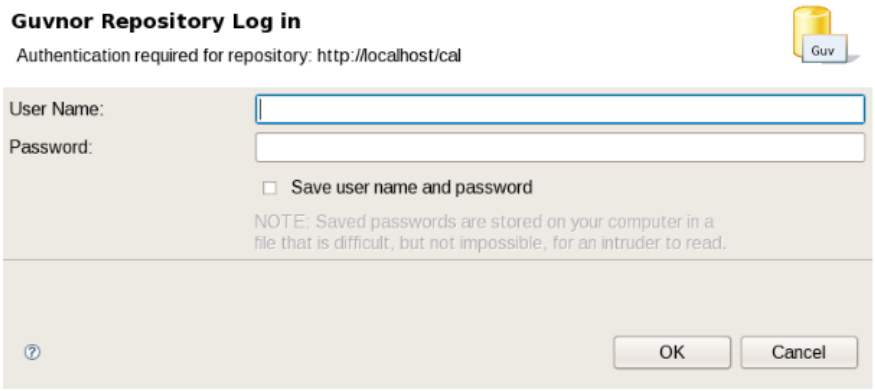


图 **10.5.** 登录

如果验证失败，EGT 将重试并显示验证失败错误。如果验证失败，你可以重试相同的操作并使用不同的凭证。EGT 在不同的情况下会调用 Guvnor 库，如当决定资源更新是否可用时，所以，如果你使用会话验证方式，根据你执行的行为，验证对话框将在 Eclipse 工作台会话期间多次出现。为了便于使用，我们推荐你将验证信息保存在 key-ring 里。Eclipse 的 key-ring 文件和其他平台如 Mac OS X 和许多 Linux 的 key-ring 文件不同。因此，有时候如果你在 EGT 之外访问 Guvnor 库，key-ring 文件可能无法同步，从而导致验证失败。这令人讨厌，但在这种情况下你应该使用平时的凭证。

Guvnor 连接向导执行完毕后，新的库连接将出现在 **`Guvnor Repository Explorer`** 里。然后你就可以扩展树来参看库的内容。

10.3. Guvnor 库浏览器

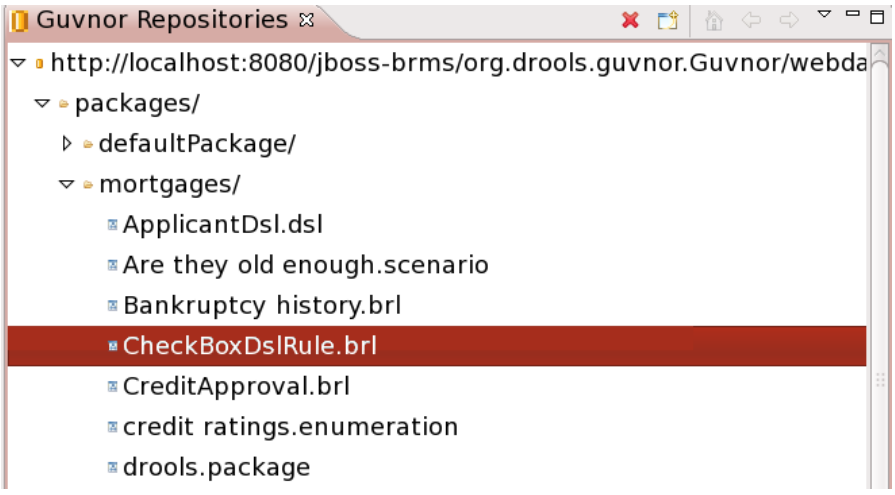


图 **10.6.** 浏览器

Guvnor 库浏览器视图显示 Guvnor 库内容的树型结构。如上所述，它具有创建 Guvnor 库连接的菜单和工具栏行为。工具栏里的红色 "X" 和菜单条里的 “Delete“ 都可以删除 Guvnor 库连接，而 “Refresh" 菜单则为所选节点加载树型内容。最后，它有大量的工具栏/菜单项支持 “drill-into" 功能：工具栏里是房子（“return to top level/home”）和箭头（go into/back）。当用于深度嵌套的树型结构且你希望专注于树的分支时，Drill-down 是很有用的。

对于 Guvnor 库文件，有大量的操作可以执行。在 Guvnor 库里选择一个文件将导致 Eclipse 属性视图更新该文件的细节：

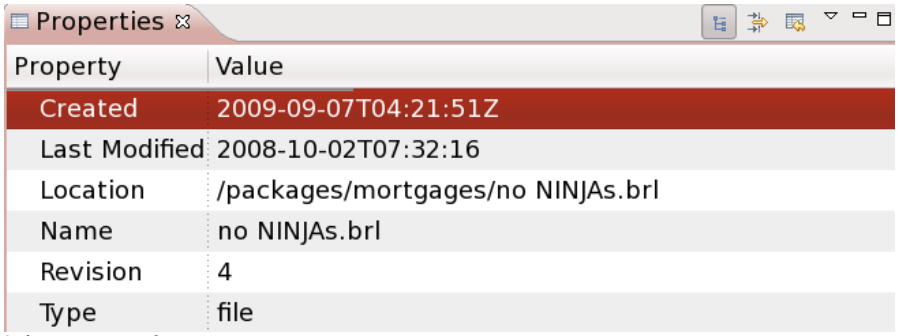


图 **10.7.** 属性

双击树里到文件夹（目录）将扩展或收缩该文件夹。双击文件夹里的文件将在 Eclipse 里打开一个只读编辑器，显示该文件的内容：

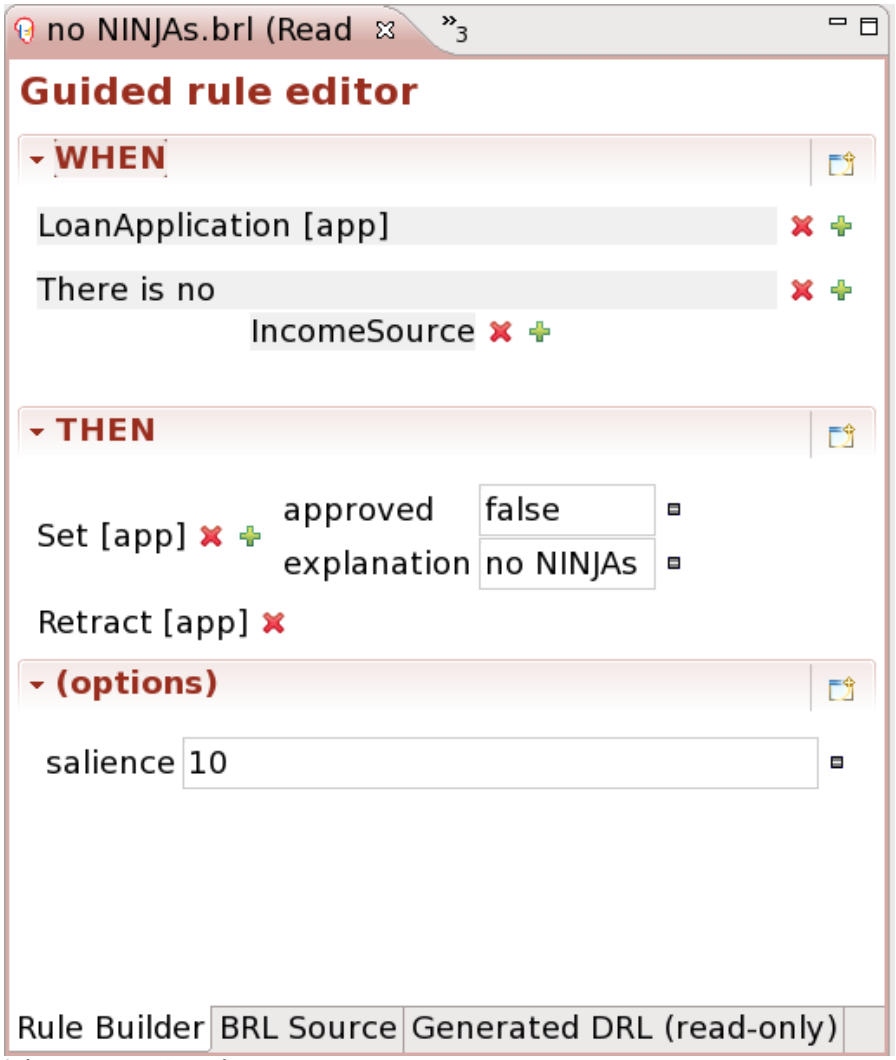

no NINJAs.brl (Read
Guided rule editor
WHEN
LoanApplication [app]
There is no
IncomeSource
THEN
Set [app]
approved
false
explanation
no NINJAs
Retract [app]
(options)
salience
10
Rule Builder
BRL Source
Generated DRL (read-only)

图 **10.8. 文件内容**

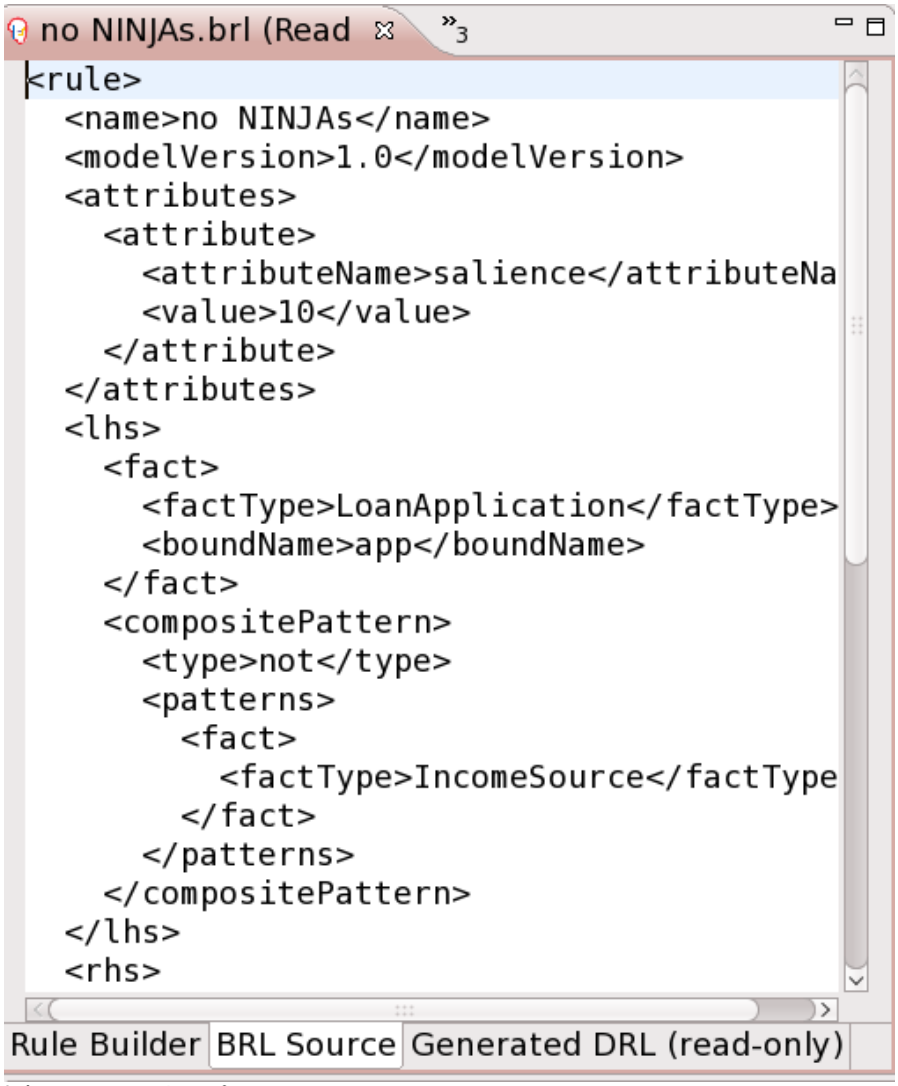


图 **10.9. 文件内容**

从 Guvnor 库树里拖动一个文件到 Eclipse 本地项目（如 Eclipse 资源导航器视图）里的文件夹里将在本地 Eclipse 工作区里创建该文件的拷贝。（请注意：在只读编辑器里打开文件时你也可以用 “Save As…” 保存一个本地到可写版本，但这样并不能关联所创建的文件和它的 Guvnor 源。）最后，你可以用 "Show History" 上下文菜单查看所选文件的修订历史记录。（下面将讨论资源历史记录的细节。）

10.4. Guvnor 文件的本地拷贝

如“简介”章节里所提及的，EGT 的主要目的是允许使用 Guvnor 库里的资源进行开发。有两个方法可获取 Guvnor 库资源的本地版本：

1. 如上所述，从 Guvnor 库浏览器里拖-放。

2. 如 第 10.6 节 “导入 Guvnor 库资源” 所解释的，使用 “import from Guvnor” 向导。

创建了 Gunvor 库文件的本地拷贝后，EGT 将设置本地拷贝和库里的主文件间的关联。（这个信息保存在本地项目的 “.guvnorinfo” 文件夹里，象所有的元数据，最终用户不应该修改它。）这个关联使得一些操作（如更新和提交）可与 Guvnor 库里的主文件进行同步。EGT 会装饰和主文件关联的本地资源。这种装饰出现在遵循通用导航器框架的 Eclipse 视图里，如 Eclipse 资源导航器和 Java 软件包浏览器。下图显示了 Eclipse 资源导航器里的装饰：

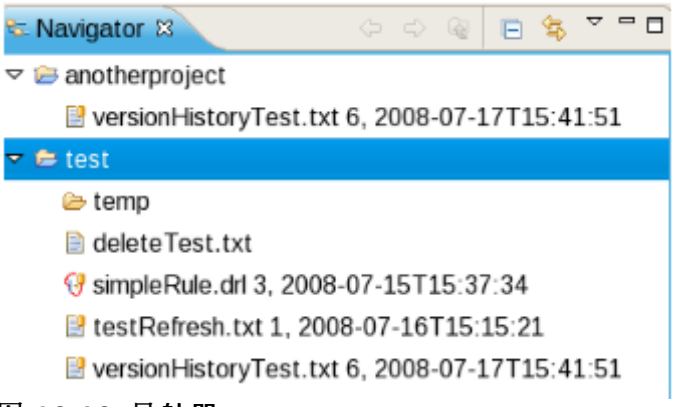


图 **10.10.** 导航器

注意文件图像右上角的 Guvnor 图标装饰以及附加在文件名后的 Guvnor 修订记录细节。（你可以修改这些东西的存在与否和位置。细节请参考下面的 “Guvnor 首选项” ）这里我们可以看到，例如，“simpleRule.drl” 和 Guvnor 库资源关联而本地拷贝基于版本 3，并带有 “7-15-2008, 15:37:34“ 时间戳。而文件 “ deleteTest.txt” 没有和 Guvnor 库文件关联。通过上下文菜单到 ”Properties“ 部分，你可以在标准的 Eclihttps://translate.jboss.org/webtrans/clear.cache.gifpse 属性页面找到关于关联的更多细节：

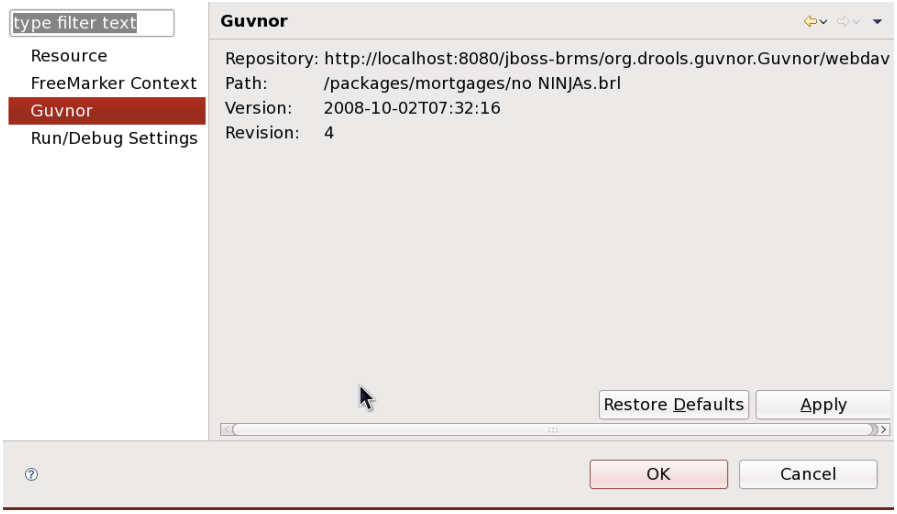


图 **10.11.** 属性

EGT 为标准的 Eclipse 属性对话框贡献了一个属性页面，上面显示了它的内容。其中包括专门的 Guvnor 库、库里的位置、版本（日期/时间戳）和版本号码。

10.5. 用于本地 Guvnor 资源的行为

EGT 提供大量的行为（通过文件的“Guvnor”上下文菜单）以用于处理文件，包括那些和 Guvnor 库主拷贝相关和不相关联的。这些行为是：

- 更新（Update）
- 添加（Add）
- 评论（Commit）
- 显示历史（Show History）
- 比较版本（Compare with Version）
- 切换版本（Switch to Version）

- 删除（Delete）
- 断开连接（Disconnect）

每种行为都将在下面进行描述。

更新（Update）行为：

更新（Update）行为可用于一个或多个没有和 Guvnor 库主拷贝同步的 Guvnor 资源。这些资源将不会进行同步，因为（1）发生了本地的修改，或（2）Guvnor 库里的主拷贝已经发生了变化。执行更新动作将用当前的主拷贝替换本地文件内容（相当于“切换版本（Switch to version）”到最新的版本）。

添加（Add）行为：

添加（Add）行为可用于和 Guvnor 库主拷贝没有关联的一个或多个本地文件。选择 Add 行为将启动 “Add to Guvnor” 向导：

图 **10.12. 添加（Add）行为**

向导的第一页将请求你选择目标 Guvnor 库以及创建一个新的 Guvnor 库连接（第二页和上面描述的 Guvnor 连接向导的相同）。选择了目标 Guvnor 库以后，向导将要求你选择添加所选文件的文件夹位置：

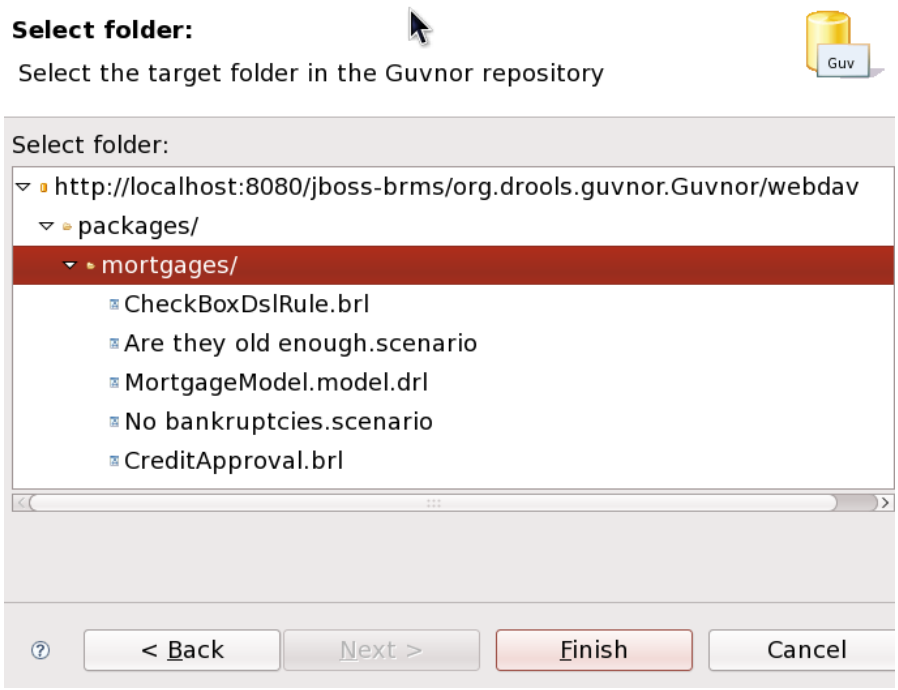


图 **10.13. 添加（Add）行为**

在这里我已经选择了 **`mortages`** 文件夹为目的地。点击 “Finish” 添加所选文件到 Guvnor 库并创建本地文件和 Guvnor 库文件间的关联。这个向导不会允许你在添加新文件时覆盖现有的 Guvnor 库文件。

比较版本（Compare with Version）行为：

Compare with Version 行为对于和 Guvnor 库相关联的文件是启用的。它首先启动一个向导并要求选择需要比较的版本（和本地文件内容）：

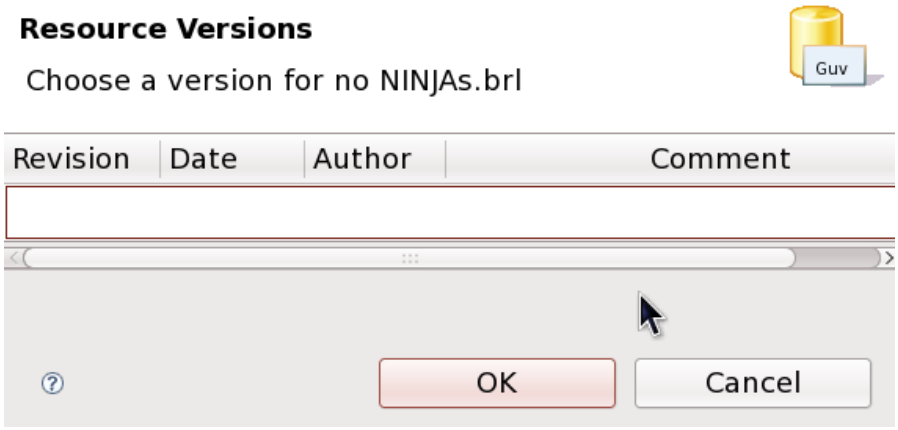


图 **10.14. 比较**

选择了版本后，这个行为打开 Eclipse 比较编辑器（只读）：

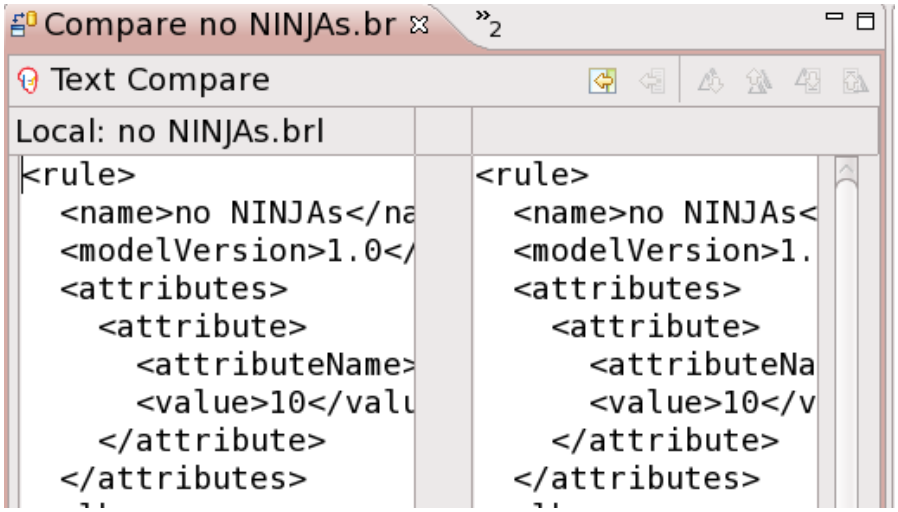


图 **10.15.** 比较

这个编辑器使用 Eclipse 标准的比较技术来显示两个版本的不同。如果没有不同，编辑器将不会被打开而是弹出一个对话框说明没有任何不同。

切换版本（Switch to Version）行为：

Switch to Version 行为对于和 Guvnor 库相关联的文件是启用的。它首先提示选择版本：

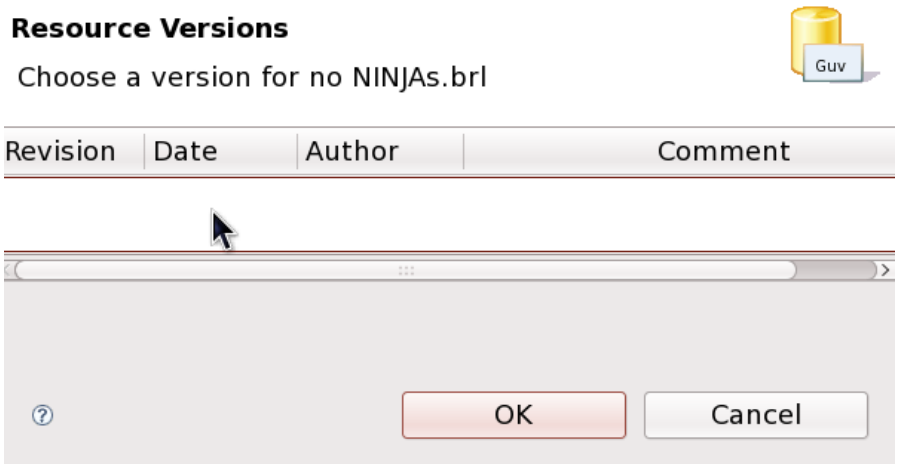


图 **10.16.** 版本

选择了版本后，Switch to Version 行为将用所选版本替换本地文件的内容。

删除（Delete）行为：

删除（Delete）行为对于和 Guvnor 库相关联的文件是启用的。通过对话框进行确认后，Delete 行为删除 Guvnor 库里的文件并和它相关联的本地元数据。

断开连接（Disconnect）行为：

断开连接（Disconnect）行为对于一个或多个和 Guvnor 库相关联的文件是启用的它删除和 Guvnor 库相关联的本地元数据。

Guvnor 资源历史视图：

Guvnor 资源历史视图应该提供所选文件的修订历史记录，对于本地文件和 Guvnor 库都是一样。这个视图的初始状态是：

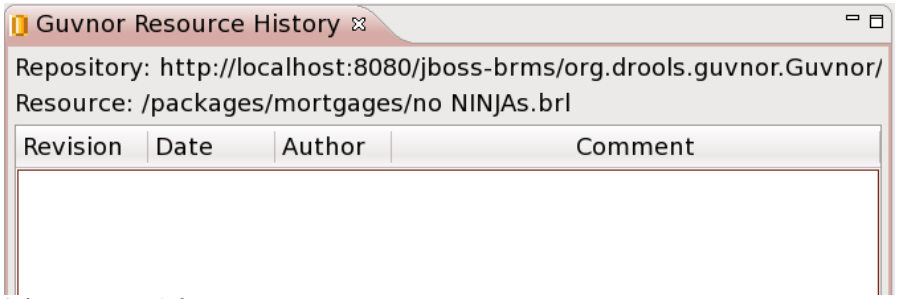


图 **10.17.** 历史

Guvnor 资源历史视图是由本地的 “Guvnor” 上下文菜单或 Guvnor 库浏览器里的文件的上下文菜单里的 “Show History” 选项决定的。执行了这个动作后，Guvnor 资源历史视图将进行更新以显示修订历史记录：

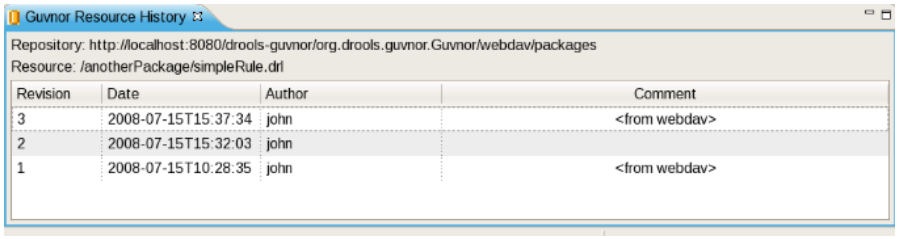


图 **10.18.** 历史

在这里我们看到文件 “simpleRule.drl” 有三个修订版本。双击某个版本（或选择上下文菜单的『打开（只读）』）将打开一个带有版本内容的 Eclipse 只读编辑器。（请注意：在只读编辑器里打开文件时你也可以用 “Save As…” 保存一个本地到可写版本，但这样并不能关联所创建的文件和它的 Guvnor 源。）

10.6. 导入 Guvnor 库资源

除了从 Guvnor 库浏览器视图里拖放单个文件以外，EGT 也包括一个可复制多个文件到本地工作区的向导（以及设置和 Guvnor 库的关联）。在 Eclipse Import 、Guvnor、Resource from Guvnor 和 Eclipse File、New、Other、Guvnor、Resource from Guvnor 菜单选项里，这个向导都是可用的。（请注意：在这两个位置这个向导是相同的，只是为了方便用户在不同的类别里使用这个功能。）向导的第一页将要求选择源 Guvnor 库并创建新的 Guvnor 库连接（第二页和上面描述的 Guvnor 连接向导相同）。

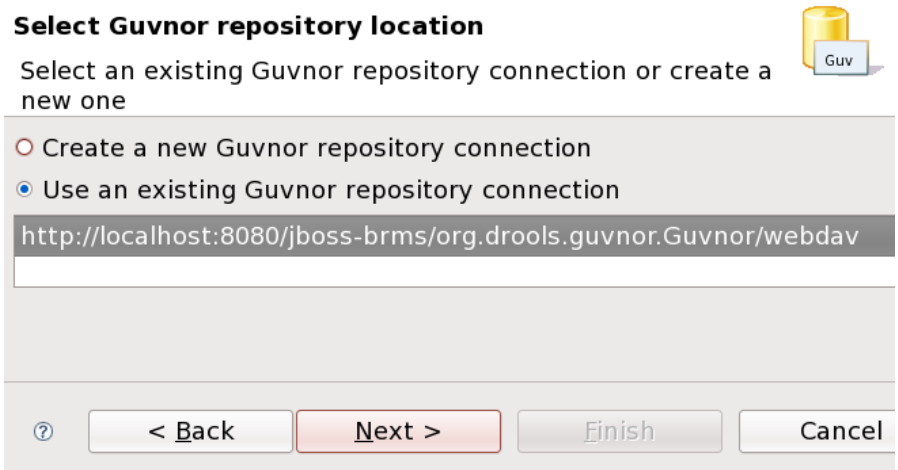


图 **10.19.** 导入

选择了源 Guvnor 库后，向导会提示你选择资源：

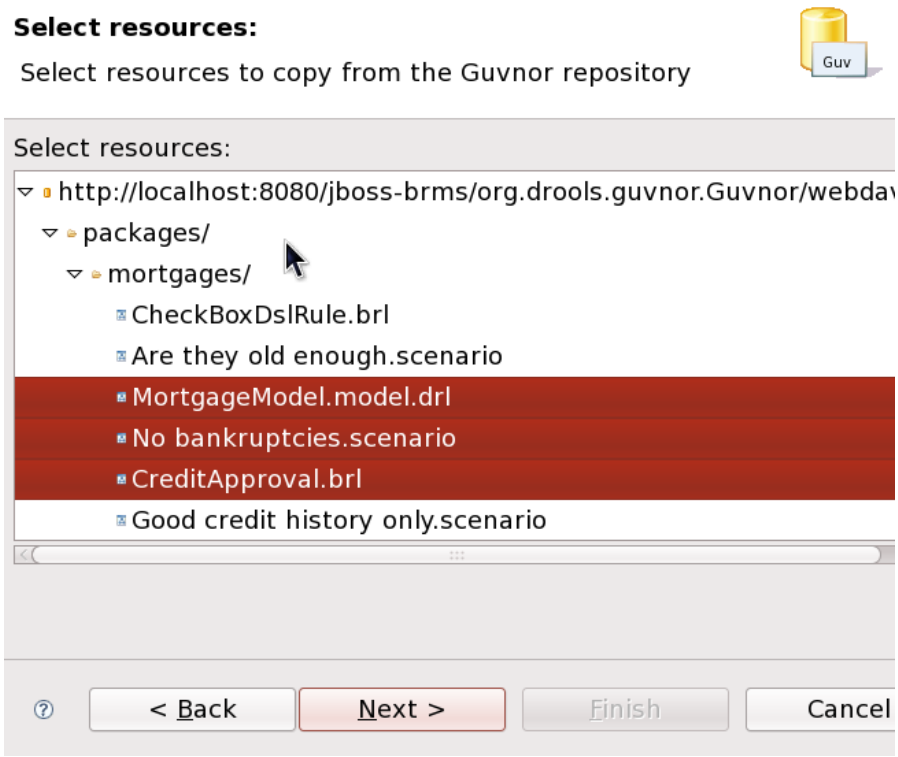


图 **10.20.** 导入

最后，选择本地工作区里的目标位置：

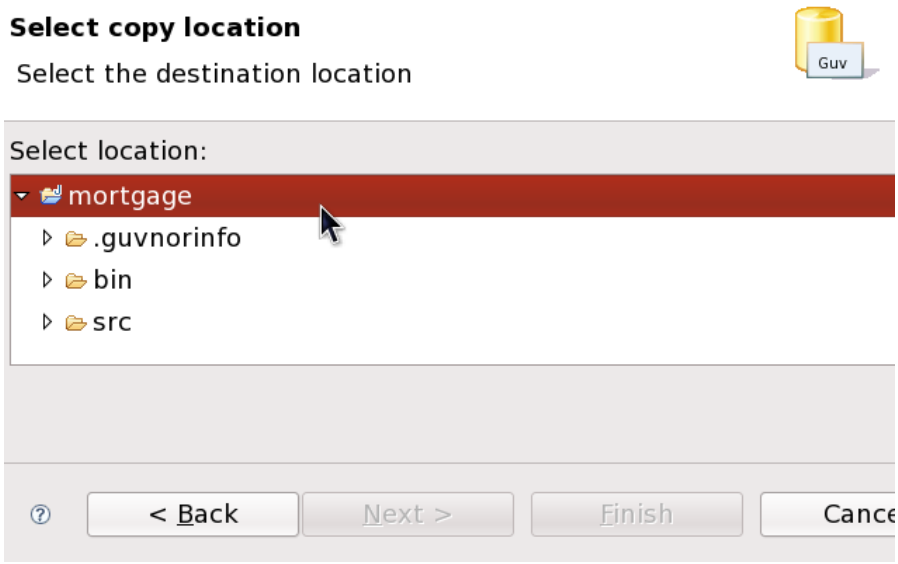


图 **10.21.** 导入

之后向导将从 Guvnor 库复制所选的文件到本地工作区。如果目的地已经存在相同名字的文件，向导将使用 Eclipse 的标准 “prompt for rename” 对话框：

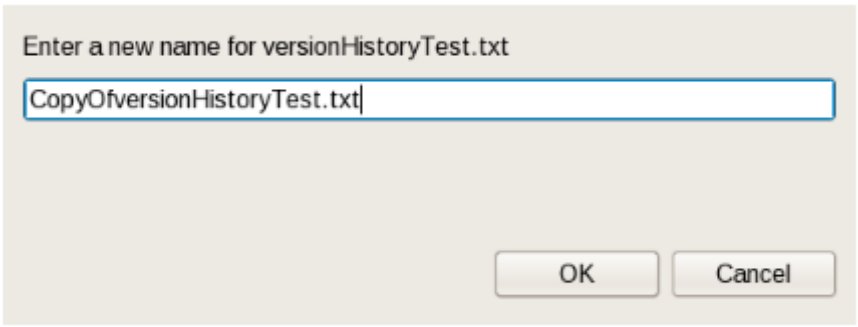


图 **10.22. 复制**

10.7. Guvnor 插件的首选项

EGT 在提供

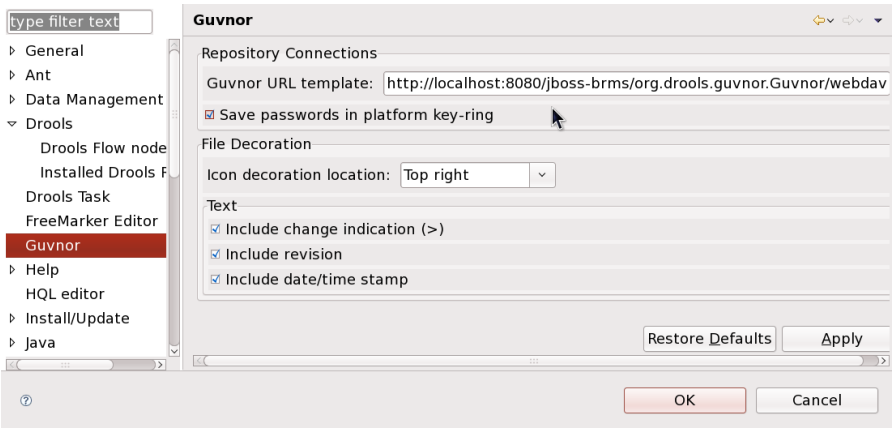


图 **10.23. 首选项**

首选项有两个类别：Guvnor 库连接和本地 Guvnor 库资源装饰。

Guvnor 库连接的首选项

对于 Guvnor 库连接，你可以设置两个首选项，它们是在创建新连接时使用的。第一个是缺省的 Guvnor 库 URL 模板，通过简单地修改部分字段（如主机名），它使得创建多个类似的连接更为容易。第二个是是否在 Eclipse 平台 Key-ring 里保存验证信息，在缺省情况下你应该启用它。使用 Guvnor 库 URL 模板时，是否在 Eclipse 平台 Key-ring 里保存特定的验证信息实例是可以在创建连接时决定的。也就是说，这两个首选项都已经简单设置为了合理的缺省值。

本地 Guvnor 库资源装饰的首选项

EGT 提供的首选项的第二个类别处理和 Guvnor 库资源相关联的本地资源的外观。既然 Guvnor 库不是 SCM 的替代物，且 Eclipse 里的 SCM 工具倾向于装饰本地资源，能够控制 EGT 装饰其本地资源以避免和 SCM 软件包冲突是很有用的。在首选项页面的 “File Decoration” 部分里，你可以选择装饰图标的位置（top right, bottom right, top left, bottom left），或者你可以选择不显示它。在 "Text" 部分，你可以格式化附加到文件名上的 Guvnor 元数据：当本地文件已经发生变动但还未提交会 Guvnor 库时是否显示一个标记(>)。是否显示修订版本号码。是否显示日期/时间戳。在点击了 “Apply” 或 “Ok” 后，对这些首选项的任何修改都是立即生效的。

第 11 章 规则软件包签名（Rule Package Signing）的客户端配置

阅读本节以学习规则软件包签名和密钥库的配置。

*规则软件包签名（Rule Package Signing）*确保了规则软件包在从 JBoss 企业版 BRMS 平台服务器下载到客户端应用程序时不会损坏或更改。在缺省情况下，规则软件包签名是禁用的。

重要

红帽强烈推荐在产品环境里启用规则软件包签名。

规则软件包签名是用*公共密钥加密*来实现的。JDK 命令 **keytool** 用于创建私有密钥和对应的公共数字证书。用私有密钥签名的软件包只可以用匹配的证书来检验。私有密钥存储在一个名为 *keystore* 的文件里，服务器使用它来自动为每个软件包签名。密钥里为每个客户端应用程序可用的公共证书被称为*信任库（truststore）*。信任库里的证书用于检验签名软件包的真实性。在下载过程中损坏或被修改的规则软件包将被客户端拒绝，因为其签名不再和证书匹配。

下面的过程描述了如何为规则软件包签名配置客户端应用程序。这涉及到设置客户端的几个属性。这些属性可以在程序里通过 **System.setProperty** 设置。**org.drools.core.util.KeyStoreHelper** 类包含几个代表这些属性的常量。

在执行这个任务之前，你需要：

- 已经安装了 JBoss 企业级 Web 平台并针对规则软件包签名进行了正确的配置。
- JBoss 企业级 Web 平台服务器所使用的包含数字证书的信任库的 URL。
- 信任库的密码（如果设置了的话）。

过程 11.1. 规则软件包签名（Rule Package Signing）的客户端配置

1. **启用签名**

   将 **drools.serialization.sign** 属性设置为 **true**。

   ```
   System.setProperty( KeyStoreHelper.PROP_SIGN, "true" );
   ```

2. **设置 TrustStore URL**

   设置 **drools.serialization.public.keyStoreURL** 属性为信任库所在的 URL。如果信任库位于客户端的 classpath 里，这可以通过 **getClass().getResource()** 方法来完成。

   例 11.1. 当信任库位于客户端的 classpath 里

   ```
   URL trustStoreURL = getClass().getResource( "BRMSTrustStore.keystore" );
   System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
   trustStoreURL.toExternalForm() );
   ```

   例 11.2. 当信任库位于 web 服务器上

   ```
   URL trustStoreURL = new
   URL("http://brms.intranet/resources/BRMSTrustStore.keystore" );
   System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
   trustStoreURL.toExternalForm() );
   ```

例 11.3. 当信任库位于本地文件系统里

```
URL trustStoreURL = new URL("file:///mnt/fileserve/rules-
server/BRMSTrustStore.keystore" );
System.setProperty( KeyStoreHelper.PROP_PUB_KS_URL,
trustStoreURL.toExternalForm() );
```

3. **可选：设置密钥库的密码**

   通过 **`drools.serialization.public.keyStorePwd`** 设置信任库的密码。只有访问信任库需要密码时才要设置这个属性。

   ```
   System.setProperty( KeyStoreHelper.PROP_PUB_KS_PWD, "sekretPasswordHere" );
   ```

- 关于如何针对规则软件包签名配置服务器的说明，请参考『http://docs.redhat.com/docs/en-US/JBoss_Enterprise_BRMS_Platform/5/html/BRMS_Administrator_Guide』上的《BRMS 管理员指南》。
- 关于公共密钥加密的更多信息，请参考『http://en.wikipedia.org/wiki/Public-key_cryptography』。

修订历史记录

修订 5.2.0-2.400	**2013-10-31**	**Rüdiger Landmann**

Rebuild with publican 4.0.0

修订 5.2.0-2	**2012-07-18**	**Anthony Towns**

Rebuild for Publican 3.0

修订 5.2.0-0	**August 4 2011**	**L Carlon**

针对 5.2.0 进行更新
将第 3 章（用户指南）分成了 3、4、5 章。
添加了关于 Guide Decision Table 的其他信息（4.4.4 章节）。
添加了关于 Fact Model 的其他信息（第 5 章）。
添加了关于 working set 的章节（第 6 章）

修订 5.1.0-0	**Mon December 13 2010**	**David Le Sage, Darrin Mison**

针对 5.1.0 进行更新
将安装和管理方面的内容移至各自的手册里
添加新收件箱和评论功能的内容（3.9 章节）
添加针对规则软件包签名功能的客户端配置的内容（第 5 章）

修订 5.0.2-0	**Wed May 5 2010**	**Darrin Mison**

针对 5.0.2 进行更新
BRMS-262 添加配置日志的内容（2.7 章节）
BRMS-233 更新几个截屏 (第 4 章)

修订 5.0.1-0	**Fri Oct 3 2009**	**David Le Sage, Darrin Mison**

针对 5.0.1 进行更新
BRMS-227 - 添加 2.3 本地化章节

修订 5.0.0-1	**Thu 16 Jul 2009**	**Darrin Mison**

BRMS-203 - 更新所支持的数据库的细节

修订 5.0.0-0	**Mon 18 May 2009**	**Darrin Mison**

创建 5.0.0