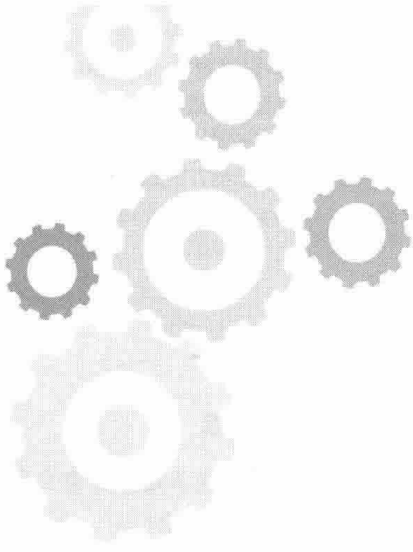
用于反欺诈平台、决策平台、促销平台、风险控制平台等业务的规则管理系统

Drools规则引擎技术指南

来志辉◎著

入门知识+中高级语法+源码分析+扩展说明+实战案例

北京大学出版社
PEKING UNIVERSITY PRESS



Drools规则引擎
技术指南

来志辉◎著

北京大学出版社
PEKING UNIVERSITY PRESS

内 容 提 要

Drools 规则引擎已经有几年的发展史了，但由于学习成本较高，且国内并没有详细的中文文档，导致 Drools 规则引擎在国内市场推行缓慢。本书将对 Drools 规则引擎进行一个详细说明，共分为六篇，基石篇主要介绍规则引擎的入门知识，基础篇详细介绍规则引擎的基础语法、规则属性、关键字及错误信息，中级篇介绍规则中级语法等，高级篇介绍 Workbench、Kie-Server、动态规则、多线程中的 Drools 等高级用法，源码篇为 Drools 源码分析，扩展篇为 Drools 扩展说明。除了讲解 Drools 规则引擎的思维方式外，还在每一个知识点上辅以大量的代码案例，并且有很多实战经验及思想在里面。本书作为国内第一本 Drools 规则引擎的中文教程，Java 开发者、对 Drools 规则引擎有兴趣的软件开发人员或系统架构师都可以阅读。

图书在版编目(CIP)数据

Drools规则引擎技术指南 / 来志辉著. -- 北京 :北京大学出版社, 2019.7
ISBN 978-7-301-30549-2

Ⅰ. ①D… Ⅱ. ①来… Ⅲ. ①搜索引擎—程序设计—指南 Ⅳ. ①TP391.3-62

中国版本图书馆CIP数据核字(2019)第103274号

书　　名	Drools规则引擎技术指南 DROOLS GUIZE YINQING JISHU ZHINAN
著作责任者	来志辉　著
责任编辑	吴晓月　刘沈君
标准书号	ISBN 978-7-301-30549-2
出版发行	北京大学出版社
地　　址	北京市海淀区成府路205 号　100871
网　　址	http://www.pup.cn　　新浪微博: @ 北京大学出版社
电子信箱	pup7@ pup.cn
电　　话	邮购部 010-62752015　发行部 010-62750672　编辑部 010-62570390
印 刷 者	山东百润本色印刷有限公司
经 销 者	新华书店
	787毫米×1092毫米　16开本　29.25印张　667千字
	2019年7月第1版　2019年7月第1次印刷
印　　数	1-3000册
定　　价	99.00 元

前言

Preface

在科技急速发展的今天，各种高端技术不断涌出，常见的有大数据、人工智能、机器学习、深度学习、区块链等，这类技术已经成为各大公司的首选。如今，金融行业、医疗行业、教育行业、保险行业及传统行业都在发生着巨大的改变，都在向互联网、物联网方向进行转型，从而突出了 IT（信息技术）的重要性。科技决定发展、科技改变人生已经不再遥不可及。

目前流行的行业，以金融类项目为例，风险控制系统、反欺诈系统、决策引擎等也成为常用且经常变化的业务。这类经常变更的业务让公司运营和程序员都非常头痛。按照产品开发的传统逻辑思维，基本写法就是添加 If 判断，或者通过 SQL 查询条件中的动态添加判断，如 Mybatis 配置文件中的 <if>。这应该是大部分程序员第一时间想到的解决方案，但是依照这类传统的思维方式考虑问题会出现以下弊端。

（1）增加开发人员与测试人员的工作量。

（2）部门间需要更加频繁地进行业务沟通，时间成本增加。

（3）增加用人成本，造成公司损失。

如何解决这些弊端成了技术部门的重要任务之一。规则引擎的出现完美地解决了这个问题。我刚开始研究规则引擎时有些迷茫，所面临的问题很多，在网上搜索规则引擎时，不知道选择学习哪一个规则引擎，当时网上规则引擎有 IBM ILog、URule、Rule Engine、Visual Rules、Drools。分析后，我发现只有 Drools 是开源项目，于是就开始了学习 Drools 的苦行。为什么说是苦行呢？原因很简单，Drools 有一定的学习成本，而且当时（2016 年）网上可以找到的教程实在是少之又少，很多人都是会用一些但并不知其道理与知识点，遇到问题连解决方案都没有，只能通过自己不断测试实践和钻研英文资料进行学习，鉴于我英文水平实在有限，就在一些 QQ 群和国外的论坛里找寻志同道合的人。经过一年的努力，我编写了《Drools 技术指南中文教程》（电子文档），创建了有关 Drools 6.4 版本的博客[①] 和 Drools 技术讨论群[②] 及微信公众号[③]，为国内 Drools 技术填补了空白。

本教程并非翻译官方文档，虽然有些地方看起来很相似，但知识点的说明比官方更好理解，下面跟着我的思路开始学习 Drools 吧！

①作者的博客地址：https://me.csdn.net/u013115157。
②作者的技术讨论及教学视频 QQ 群：676219749。
③作者的微信公众号：程序猿之塞伯坦。

本书提供以下源代码下载地址。

Drools 测试用例：https://github.com/projectLzh/DroolsTest。

Drools Spring：https://github.com/projectLzh/DroolsSpring。

Drools Spring Boot：https://github.com/projectLzh/DroolsSpringBoot。

JavaWorkbench 交互：https://github.com/projectLzh/WorkbenchJava。

读者也可以扫描下方二维码，关注微信公众号，输入书中 77 页的资源下载码，获取源代码。

目录

Contents

第一篇 基 石 篇

第二篇 基 础 篇

第四篇 高 级 篇

第五篇 源 码 篇

第六篇 扩 展 篇

第一篇

基石篇

第1章

Drools概述

1.1 程序来源于生活

程序来源于生活，这一点从 Java 语言中就能深刻体会到。万事万物皆对象，各种设计模式、框架技术、核心思想都源于生活。在日常生活中处处都有规则，用特定的规则来约束人们的行为，按照这些规则人们就知道哪些应该做、哪些不应该做，甚至哪些是必须要去做的。例如，当一个路口红灯亮起时，所有人和车都是不能通过的，否则就会有安全隐患，这里的红灯就是一个规则，它使人或车停下。Drools 是一个开源的规则引擎技术，下面就针对 Drools 规则引擎进行详细说明。

1.2 Drools是什么

Drools 是开源的项目，是具有一定学习成本的规则引擎技术，通过 Drools 特定的语法，将固定的业务、经常变的业务统一管理。它以特定的文件或数据库方式将规则内容进行存储（存放在哪里都可以，可以理解为它只需一个存储介质），以 Drools 包提供的接口对规则内容进行处理，并返回公司运营所规定的业务结果，当然这个结果也是正确的。

通过更简单的规则结构，预判业务是否正确，及时对业务进行测试，如果需要了解规则内部的执行过程，可以添加 Drools 规则引擎对外提供的事件监听功能。

1.3 Drools简要概述

Drools 是一款基于 Java 语言的开源规则引擎，可以将复杂且多变的业务规则从硬编码中解放出来，以规则脚本的形式存放在文件或特定的存储介质中（这里可以是数据库表），使得业务规则的变更不需要修正项目代码、重启服务器就可以在线上环境立即生效。这里可以理解为动态代码（动态业务）。

规则引擎的核心目的之一是将业务决策从程序代码中分离出来，使其代码与业务解耦合。通过特定的语法内容编写业务模块，由 API 进行解析并对外提供执行接口，再接收输入数据、进行业务逻辑处理并返回执行结果。引用规则引擎后的效果如图 1-1 所示。

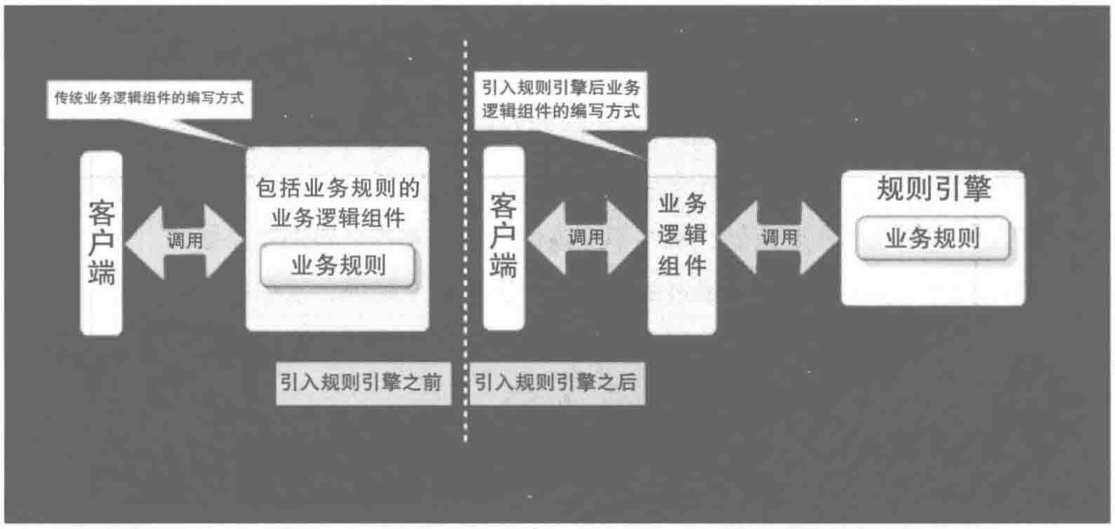


图1-1　引用规则引擎后的效果

1.4 Drools发展趋势

Drools 规则引擎是为开发人员提供的 KRR 功能程序，KRR 是人工智能的核心，所以 Drools 也是人工智能的一个分支，是专家系统的另一种展现方式。Drools 的核心之一是操作业务，与传统项目不同的是，业务由业务操作员管理，不再需要程序员去改代码，极大地减少了开发周期，使得业务员不再依赖 IT 部门就可以完成业务的变更。正是因为规则引擎有这样的优势，所以受到了风险控制类、反欺诈类、促销类项目的青睐。目前，不仅阿里巴巴、京东、智联招聘、智业软件（国内知名医疗研发公司）等公司在使用，而且国内一些商业产品的规则引擎底层也是通过 Drools 实现的。

1.5 Drools版本

在开始学习 Drools 规则引擎之前，先制订学习计划，我刚学习 Drools 时的版本是 Drools 6.4，本书主讲的版本是 Drools 7.10。从我研究 Drools 规则引擎开始到本书中主讲的 7.10 版本，我发现规则引擎的脚本并没有太多的变化，无论是 5.x 版本还是 7.x 版本，或者是 6.4 版本，在规则引擎的语法上都相差无几。发生变化的应该是性能的优化和 Drools 提供的 API 了，自 6.0 版本后 Drools 的 API 发生了巨大的变化，其相比 5.x 版本，更加简洁、容易理解。所以读者在学习和使用过程中，无须担心 Drools 规则引擎语法的使用问题。

1.6 Drools新特性

自 6.0 版本开始，Drools 推出了一个全新的概念，基于 KIE 的全新 API，其目的是为了更简单地操作规则引擎，用过 Drools 5.x 版本甚至更早版本 Drools 的人都知道，要执行一个规则文件是相当烦琐的事情。

KIE（Knowledge Is Everything）是“知识是一切”的缩写，是 Jboss 一系列项目的总称。如图 1-2 所示，KIE 的主要模块有 OptaPlanner、Drools、UberFire、jBPM。下面分别讲述这些模块的用途。

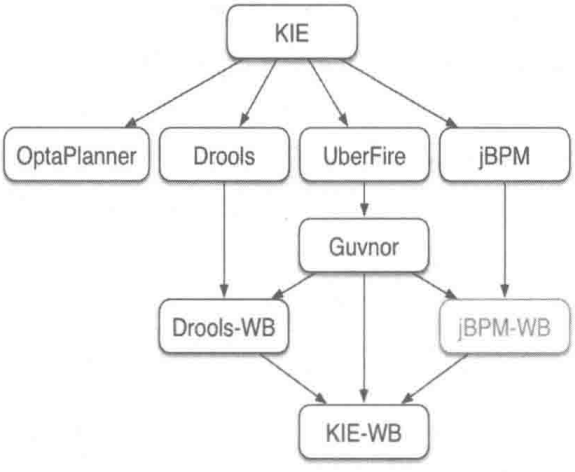


图1-2 KIE结构

OptaPlanner：一个本地搜索和优化工具，按官方的说法，OptaPlanner（以下简称为 OP）原本是 Drools 平台的组件，但由于发展的趋势，现在独立于 Drools（虽说已经独立了，但还是拥有良好的整合功能），成为与 Drools、jBPM 等同级的 KIE 组件。

Drools：规则引擎，自 6.0 版本后，Drools 提供了 Web IDE，从图 1-2 中可以看出，Drools 组件包含了 Drools-WB，Drools-WB 又指向了 KIE-WB。按照官方的说法，KIE-WB 是一个强大的 Web IDE，它结合了 Drools、Guvnor、jBPM 插件成为一个超级平台，KIE-WB 的作用将在后面进行详细说明。

UberFire：一个全新的组件，其功能类似于 Eclipse，包括插件中的样式和页面。UberFire（以下简称为 UF）独立于 Drools 和 jBPM，可以进行独立部署，对外提供功能服务。

jBPM：一个业务流管理组件，用于实现自动化业务流程和决策的工具包。jBPM 源于 BPM（业务流程管理），但它已经发展到使用户能够在业务自动化中选择自己的路径。jBPM 提供了各种功能，可以将业务逻辑简化和外部化为可重用的资产，如案例、流程、决策表等。为什么 jBPM-WB 是灰色的？按官方的说法，jBPM-WB 对于 KIE-WB 是多余的，因为 KIE-WB 完美地结合了工作流，所以将其制成灰色也不为过。

1.7 KIE生命周期

通过上述的说明，相信读者对 KIE 已经有一个大致的了解了，下面来介绍 KIE 的生命周期。

（1）创建：通过 KIE-WB 创建知识库，如 DRL、BPMN2、决策表、决策树、实体等。

（2）构建：构建一个可提供 KIE 部署的组件，简单地说，就是生成一个包含知识库的 jar 包，通过 Java 代码或 KIE-WB 提供的服务器（Kie-Server）来操作业务规则。

（3）测试：在构建部署前，对整体知识库进行测试。通常使用的测试场景将在 4.1 节中讲到。

（4）部署：KIE 使用 Maven 将其组件部署到应用程序上。

（5）使用：通过 KieContainer 创建 Kie 会话（KieSession），为执行提供前提条件。

（6）执行：通过执行 KieSession 与 Drools 系统进行交互，执行规则、流程、决策表等。

（7）交互：用户与 KieSession 的交互，通过代码或页面进行操作。

（8）管理：管理 KieSession、KieContainer 等 Drools 提供的相关对象。

Drools 是很多年前就有的规则引擎技术，但随着新技术的涌现，为了适用于各式各样的场景，Drools 进行了模块化分类，使其功能更加独立、内容更加丰富、分工也更加明确。下面通过 3 种业务建模技术来实现不同应用级的业务场景，其中较为核心的是业务规则管理。

① 业务规则管理：主要以规则管理为核心进行详细的业务介绍。

② 流程管理：指规则流部分。

③ 复杂事件处理：负责事件处理功能。

1.8 为什么要用规则引擎

一般的项目中没有引用规则引擎之前，通常的做法都是使用一个接口进行业务工作。首先要传进去参数，通过 if…else 或其他方式进行业务逻辑判断，其次要获取到接口执行完毕后的结果。引用规则引擎后就截然不同了，原有的 if…else 不复存在，代替它们的是规则引擎脚本，通过规则引擎实现可动态变化的“if …else”。

规则引擎可以给项目带来什么？规则引擎的应用场景是什么？使用规则引擎的好处是什么？下面将进行详细介绍。

1. 规则引擎可以给项目带来什么

（1）给公司运营人员带来了什么？

① 将业务规则交于业务员来处理。

② 提高业务灵活性，业务员可以随时对公司业务进行修改（设计时要加权限）。

③ 增加业务处理的透明度，业务规则可以被管理。

④ 修改业务将不再通过开发人员，极大地减少了对 IT 人员的依赖。

⑤ 减少各部门之间的矛盾，各司其职。

（2）给公司 IT 部门带来了什么？

① 简化了系统的复杂度，使系统间变得简单、透明。

② 提高了系统的可维护性。

③ 减少了维护成本。

④ 规则引擎是相对独立的，只关心业务规则，并不关心与谁交接。

⑤ 减少了“硬代码[①]”业务规则的成本和风险。

⑥ 减少了与业务员的冲突。

2. 规则引擎的应用场景

（1）适用的行业分类。

① 金融行业——黑名单、白名单、风险投保。

② 医疗行业——合理输血、合理用药。

③ 电商行业——促销平台。

（2）适用的系统分类。

① 风险控制系统——风险贷款、风险评估系统。

② 反欺诈项目——银行贷款、征信验证。

③ 决策平台系统——财务计算。

④ 促销平台系统——满减、打折、加价购。

3. 使用规则引擎的好处

（1）应用概述说明。

① 应对复杂多变的业务场景。

② 快速且低成本地进行业务规则变更。

③ 业务员直接管理，不需要程序员进行干预，减少风险。

④ 平台独立化，系统迁移、系统升级都极为方便。

（2）作用与优点。

① 业务规则与系统代码分离，实现代码与业务的解耦合。

② 提供领域语言（自然语言），使业务人员更容易理解。

③ 提供了可视化页面[②]操作，使用更简单。

④ 大大提高了对复杂逻辑代码的可维护性。

⑤ 可随时对业务进行扩展和维护。

⑥ 符合公司对敏捷性或迭代性开发的策略。

①硬代码指将业务逻辑写到项目代码中，风险是维护成本高、开发难度加大、服务器重启等。
②可视化页面指 KIE-WB，有一定的学习成本，比较适合 IT 人员使用。