|  |
| --- |
|  |
| 规则引擎设计 |
| 基于Drools的规则引擎设计说明 |

|  |
| --- |
|  |

修改历史

|  |  |  |
| --- | --- | --- |
| **版本** | **时间** | **内容描述** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

目录

[1. 概要 4](#_Toc353365221)

[2. 总体设计 4](#_Toc353365222)

[3. 功能设计 4](#_Toc353365223)

[3.1. 流程设计 4](#_Toc353365224)

[3.2. 数据结构 5](#_Toc353365225)

[3.3. 功能设计 5](#_Toc353365226)

[4. 页面设计 6](#_Toc353365227)

[4.1. 场景维护 6](#_Toc353365228)

[4.2. 模型维护 6](#_Toc353365229)

[4.3. 规则维护 7](#_Toc353365230)

[4.4. 条件维护 7](#_Toc353365231)

[4.5. 动作维护 8](#_Toc353365232)

[4.6. 变量维护 8](#_Toc353365233)

[4.7. 规则记录 9](#_Toc353365234)

[5. 核心设计 9](#_Toc353365235)

[5.1. 流程设计 9](#_Toc353365236)

[5.2. 详细设计 10](#_Toc353365237)

[6. 对外接口 10](#_Toc353365238)

[7. 关键技术 10](#_Toc353365239)

[7.1. Drools 10](#_Toc353365240)

[7.2. 单点登录 10](#_Toc353365241)

[8. 参考文档 10](#_Toc353365242)

# 概要

规则引擎是一种嵌入在应用程序中的组件，实现了将业务规则从应用程序代码中分离出来，并使用预定义的规则语义来编写业务规则。规则引擎接受数据输入，解释业务规则，并根据规则执行相应的业务逻辑。

一个业务规则包含一组条件和在此条件下执行的操作，它们表示业务规则应用程序的一段业务逻辑。我们在业务中设置一个或者多个条件，当满足这些条件时触发相应的操作，规则引擎设计的初衷是可以将复杂多变的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者数据库中，使得规则的变更不需要修改代码即可使用，做到最大程度的灵活。

# 总体设计

规则引擎的设计主要分为两部分：一部分是规则的维护，包括规则的创建、修改、删除；一部分是规则的处理，这部分需要结合Drools的特点来设计。规则的维护部分侧重点是页面，我们需要将用户在页面上的操作转换为Drools的规则语法并保存到数据库中。在规则处理部分，我们首先根据场景信息得到场景对应的规则信息，然后准备好场景需要使用的数据，一起扔给Drools，在Drools的动作部分会执行我们预定义好的接口。

# 功能设计

## 规则执行流程

时序图：



## 数据结构

参见 [规则引擎数据结构设计.xls](http://pm.bobomi.com.cn/index.php?m=file&f=download&fileID=377&sid=lm8tt0o001uv0orr70t14do4d1)x 表结构设计文档。

## 功能设计

结合整体需求，规则引擎应该有可扩展、易维护的特点，先将规则引擎的功能需要实现的功能点总结如下：

1. 场景维护（场景的定义）
2. 模型维护（模型即实体，包括模型的创建以及模型属性的维护）
3. 规则维护（包括对规则的增删改查）
4. 条件维护（对规则条件的增删改查）
5. 动作维护（对规则动作的增删改查）
6. 变量维护（包括静态变量和动态变量部分）
7. 规则记录（查看规则的执行记录）

# 页面设计

## 场景维护

#### 整体功能

场景维护部分需要实现场景的管理。

#### 页面设计

场景维护部分需要录入的数据包括场景名称、场景标识、场景对应的实体。场景标识为对外调用时使用。

#### 关键点

* 场景与模型的关系决定最后Drools语法生成。

## 模型维护

#### 整体功能

模型维护部分需要实现模型的管理以及模型属性的管理，这部分是整个规则引擎的最基础部分。

#### 页面设计

模型维护部分需要录入模型名称、标识名(相当于java的变量，Drools条件中使用的别名)、包名。通过这三个信息的录入可以确定模型的类型，供Drools使用。在规则属性部分，我们需要录入属性的字段名、显示名供页面部分使用。

#### 关键点

* 模型定义中的标识名和包名都是为之后Drools规则表达式中使用，一定要录入正确，包括大小写。
* 模型属性部分的字段名和显示名在条件的创建部分会使用到。

## 规则维护

#### 整体功能

规则维护部分需要实现规则的管理，规则的管理包括条件的管理、动作管理以及条件之间的关系、动作之间的关系。

#### 页面设计

规则维护部分需要录入规则开始时间、结束时间以及规则条件、规则动作信息，规则引擎会根据这里定义的规则去生成Drools表达式。

#### 关键点

* 条件部分可以对条件的关系进行管理，比如与、或。
* 动作部分目前不考虑动作的或关系。

## 条件维护

#### 整体功能

条件维护部分需要实现条件的管理，条件的创建依赖于模型中的属性。这里限定为一个条件中只能包含一个模型(实体)的属性，如果要包含多个模型，需要在规则的条件部分进行配置。

#### 页面设计

条件维护部分需要录入条件名称以及条件。条件的编辑是最为重要的一部分，这里需要支持动态变量，并且需要有高度灵活性，所以考虑将编辑框设置为可以人工编辑，并提供条件检查功能。

#### 关键点

* 一个条件只能包含一个实体属性，多实体之间的关系放到规则中维护。
* 对条件的正确性检查。

## 动作维护

#### 整体功能

动作维护部分需要实现动作的管理，系统会提前预置一些常用动作，比如送积分。这里侧重点还是对动作的配置，比如送多少积分。

#### 页面设计

条件维护部分需要提供对动作变量的维护功能。

#### 关键点

* 动作定义部分需要支持动态变量
* 暂不考虑动作的自定义（系统采用动作类型预置功能）

## 变量维护

#### 整体功能

变量维护部分需要实现变量的管理，变量包括静态变量和动态变量，静态变量也就是固定的变量值，动态变量是指变量会随着条件值的改变而改变，比如消费1元送1个积分。

#### 页面设计

变量维护部分可以创建静态变量和动态变量，静态变量的值包括字符型、数字型、日期型。动态变量需要结合模型(实体)属性来确定，也就是说动态变量的来源是模型属性。

#### 关键点

* 支持动态变量

## 规则记录

#### 整体功能

我们需要查看规则的执行记录，包括规则执行时间、规则执行是否成功以及规则执行后的动作结果。

#### 页面设计

规则的记录是包括在引擎的核心执行之中的，在规则执行后会向记录表中插入相应的信息。我们需要提供一个页面让用户来查看规则的执行记录，比如某次规则的执行结果。

#### 关键点

* 规则结果查看。

# 核心设计

## 流程设计

时序图：



## 详细设计

外部系统通过统一的接口来调用规则引擎，需要向引擎提供场景信息以及fact对象信息，为了提高规则引擎的高可用性，我们这里虚拟化为两个参数，一个是统一封装的ruleObject对象，存放Fact对象，一个是全局变量。

在引擎接收到外部系统的参数后，引擎会去查找规则中对应此场景的可用规则信息，加载到规则后，得到规则的条件信息和动作信息。条件部分可能是一串表达式，表达式中包含条件的id信息，我们需要提取出这些id，然后根据id去条件表中查找相应的条件，找到条件后，得到的仍是一些id信息，当然这些id是由运算符连接在一起的，他们可能是属性id、变量id，我将这些id用特殊符号隔开，以便区分，其中属性用$隔开，静态变量用@隔开，动态变量用#隔开。

根据不同的标识拿到条件信息后，需要根据id来重新组装条件，以符合Drools语法。

动作部分同样需要根据动作id来拼接动作语法，这块需要结合接口中的具体方法来实现，特别需要注意拼接出动态变量部分。

在这一连串的拼接之后，我们会得到规则的Drools表达式，把表达式扔给Drools处理，并做一部分预处理，以满足Drools要求。

# 对外接口

其它第三方系统调用引擎时，需要调用引擎对外提供的接口，接口中包括Fact对象ruleObject以及场景标识。方法名称如下：

void execute(RuleExecutionObject fact, RuleExecutionResult result);

其中fact 对象里包含了drools需要的fact业务对象以及规则执行的一些参数，比如，是否匹配全部规则并执行，或者根据名称匹配对应的规则然后执行。Result 里面会有一个map，用来存放结果需要的信息，比如，规则调用记录等等。

# 关键技术

## Drools

# 参考文档

1. <http://www.jboss.org/drools/>
2. <http://lalalabs.blog.163.com/blog/static/94325202201112510340870/>
3. <http://www.docin.com/p-65570981.html>
4. <http://my.oschina.net/ITBoy/blog/74679>