Использование Decision Table в JBoss Drools

Правила. Мы узнаем это слово с самого детства. В начале родители говорят нам, как нужно поступать, а как лучше не стоит. Потом мы приходим в школу, и там учителя диктуют свои порядки. В университете мы опять же сталкиваемся с правилами, которые для нас устанавливают преподаватели и деканат. Взрослеем и начинаем уже следовать законам, которые разработаны государством. Что общего у всех этих правил? То, что они выверены годами, десятилетиями, а некоторые даже и поколениями. Представьте какова была бы наша жизнь, если бы правила менялись часто: каждую неделю или каждый день!  
  
У нас в компании Mediascope в ежедневной поставке данных мы тоже используем правила: для расчета демографических атрибутов респондентов и домохозяйств, для расчета характеристик категорий товаров или брендов. Все эти правила меняются достаточно часто, и каждый раз переделывать конвейер поставки данных было бы накладно. Поэтому мы решили вынести эти правила отдельно от кода и отдать их на поддержку бизнесу. В данной статье я хотел бы показать, как мы пользуемся этими правилами, а также  
продемонстрировать основные конструкции и практики использования правил.

Что это вообще такое?

Довольно часто в процессе разработки коммерческого софта мы сталкиваемся с проблемой, когда некоторую логику расчета необходимо передать на сторону заказчика. Зачастую представитель этого заказчика не хочет (или не может) использовать какой-либо язык программирования для описания необходимой логики работы приложения. В таких случаях на помощь приходят BRMS или Business Rule Management System. Это такая информационная система для создания, управления и исполнения той самой  
бизнес-логики приложения. Её еще называют бизнес-правилами. Обычно такие системы состоят из сервера, на котором происходит выполнение правил - это юрисдикция программистов, и средств ведения самих правил - это уже зона ответственности бизнеса.  
   
Практически каждая компания, где бизнес плотно взаимодействует с программной разработкой, пытается изобрести свой "велосипед". И мы не исключение. После первой итерации разработки системы управления правилами, было принято решение отказаться от внутреннего решения в пользу уже существующих. Причин было несколько: недостаточно быстрая работа системы, неудобное описание правил, постоянно возникающие ошибки расчета, и что самое страшное, непредсказуемо меняющийся   
выходной результат. Хорошо, что мы не внедрили это решение в продакшен!   
  
BRMS фреймворков на рынке достаточно много. Свои решения предлагают многие крупные компании: IBM, Red Hat, Agiloft, SAS и даже Bosch. Все они либо платные, либо не подходили нам по тем или иным критериям. Решили начать с уже зарекомендовавшей себя системы JBoss Drools. Она достаточно надежная, проверенная временем, используется в банковских решениях, ритейле и телекоме, а также предоставляет возможность ведения бизнес-правил, как на специальном языке DRL, так и с помощью Excel-таблиц. Существует также и несколько UI решений для ведения правил. Так как наши аналитики используют  
единую систему ведения справочников и нам удобнее хранить правила там, поэтому от использования UI мы отказались в пользу понятных бизнесу Excel-таблиц.

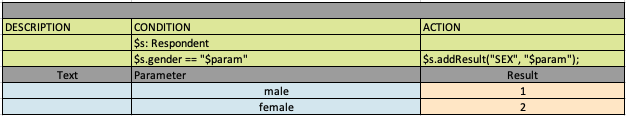
Что же такое бизнес-правило и как оно выглядит?  
  
Обычно бизнес-правило это набор инструкций или ограничений, которые позволяют принимать решение о выполнении того или иного действия. Другими словами, это некий набор логики, который позволяет в зависимости от вариации входных значений выполнять определенные действия, соответствующие входным параметрам. Давайте рассмотрим простой пример.  
  
У нас есть респондент, который обладает характеристикой **Гендерная идентичность (gender)**. В зависимости от значения этой характеристики респонденту выставляет **Пол (SEX)**. То есть, если gender = male, то значению свойства пол нужно будет поставить **1**. В противном случае это будет **2**. На языке DRL это правило будет выглядеть следующим образом:

*rule "Rule 1 Example 1"  
 when  
 $s: Respondent($s.gender == "male")  
 then  
 $s.addResult("SEX", "1");  
end  
  
 rule "Rule 2 Example 2"  
 when  
 $s: Respondent($s.gender == "female")  
 then  
 $s.addResult("SEX", "2");  
 end*

Таким образом мы имеем некий синтаксис очень похожий на язык Gherkin. В нашем случае у нас есть два правила, которые состоят из условия (указанного после ключевого слова **when**), и результата, которой будет получен в ходе выполнения этого условия (идет после слова **then)**. Условия в терминологии Drools принято называть Left Hand Side (или LHS), а действия Right Hand Side (или RHS). Стоит упомянуть еще об одном объекте: **Respondent.** Это **fact,** т.е. объект в текущей памяти Drools, над которым будут производиться те или иные преобразования. В нашем случае у этого объекта присутствуют свойства **gender** и **result**. Для того, чтобы было удобнее работать с объектами, framework предусматривает возможность введения переменных. Обычно переменные начинаются с символа **$**.

Возможно, вы подумали, что все это усложняет достаточно простую логику и можно обойтись обычной конструкцией if – else if – else. Но в случае большого правила и разных пересекающихся условий, использовать стандартные средства языка программирования уже становиться не удобно. В прочем при большом количестве условий использовать DRL-синтаксис также не представляется возможным, да и не всегда представители бизнес-аналитики хотят углубляться в изучение такого описания. В таком случае на помощь нам приходят всеми горячо любимые (а особенно аналитиками) Excel-таблицы. Drools в этом случае предоставляет нам достаточно широкий функционал их использования.

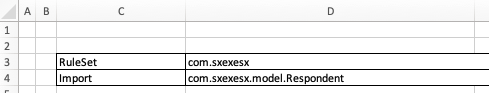
Вот так это правило будет выглядеть на языке Excel-таблиц:



Как же создать таблицу правил?

Прежде, чем мы перейдем к описанию основных конструкций, используемых в таблицах, необходимо разобраться, как же правильно формировать эту таблицу. Стоит отметить, что Drools умеет работать как с таблицами, созданными в табличных редакторах, Miscrosoft Excel или OpenOffice, так и с форматом CSV. Перед применением правил они будут сконвертированы из табличной формы в DRL формат.

В любой таблице правил можно выделить две области: область настройки правил и область описания правил. Первую обычно можно узнать по зарезервированному слову **RuleSet**, тогда как вторая – это **RuleTable**. Обратите внимание, что все зарезервированные чувствительны к регистру.



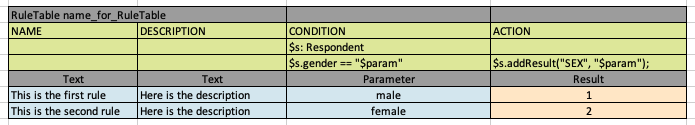
В области настройки правил обычно указываются основные конструкции DRL формата и атрибуты самих правил. Обычно это пара: зарезервированное слово в левой ячейке – значение в правой. Полный список зарезервированных слов можно найти в документации. Вот часть из них:

- **RuleSet** – здесь указывается имя пакета для сгенерированного файла DRL. Это параметр обязательно должен стоять первым.

- **Import** – через запятую указываются факты с указанием пакета, а также такие Java-классы, которые можно использовать в расчетах, например, java.lang.Math

- **Functions** – здесь можно описывать функции, которые будут работать в рамках данного RuleSet’а. Функции должны соответствовать DRL синтаксису.

В области описания правил тоже есть свои зарезервированные слова, при этом главным словом является **RuleTable**, которое указывает на то, что таблица ниже соответствует таблице правил, и которую движок Drools должен преобразовать в DRL-синтаксис. Опционально можно указать название для таблицы правил. В нашем случае это **name\_for\_RuleTable**.



Начиная со следующей строки идут колонки:

**NAME** – имя правила. Его можно не указывать.

**DESCRIPTION** – расшифровка для правила. Его тоже можно не указывать. Эти два параметра нужны для того, чтобы не потеряться в большом количестве правил.

**CONDITION** – это то Left Hand Side или указание условия на основании которого будет выполнен ACTION. Этот параметр обязателен.

**ACTION** – действие, которое необходимо применить к факту. В нашем случае это метод addResult, который добавляет в Map, значения результирующих переменных. Этот параметр обязателен. По факту в блоке ACTION выполняется Java код, поэтому наличие точки с запятой здесь также обязательно. Также, через точку с запятой можно указать сколько угодно методов.

Следует отметить, что количество CONDITION и ACTION может быть больше одного.

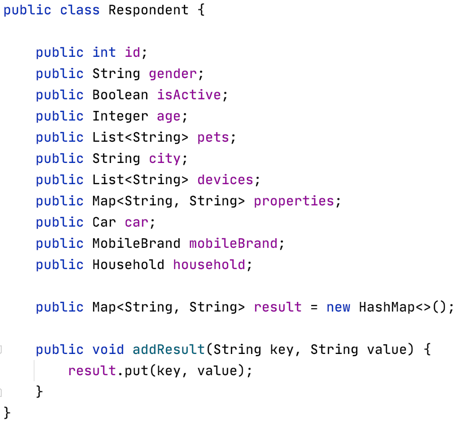
Как отмечалось ранее, строкой ниже идет присваивание переменной $s факта Respondent. Отмечу, что его полное имя с названием пакета в котором он находится обязательно указать в параметре **Import**. Если у нас несколько CONDITION относятся к одному факту, тогда ячейки с фактом необходимо объединить в одну. Также мы можем работать с разными фактами в разных CONDITION колонках, для этого мы просто указываем новый факт и новую переменную, не забыв добавить его в **Import**.

Далее, на следующей строке, у нас идет описание самих правил в колонке CONDITION и действий в колонке ACTION, которые необходимо выполнить. О том, какие могут быть условия и действия к выполнению, будет описано ниже. А пока перейдем к следующей строке. Это строка заголовков полей (Text-Parameter-Result на картинке). Ее указывать обязательно. В противном случае, те переменные/условия, которые будут указаны вместо этой строки Drools проигнорирует. Далее у нас идет указание самих условий.

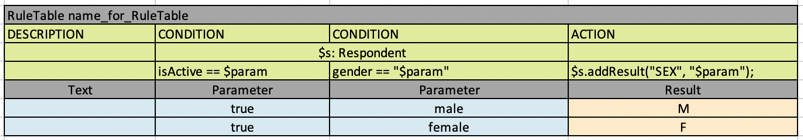
Какие условия могут быть?

В интернете достаточно много информации о том, как писать условия, но все она либо неочевидная, либо разрозненная. Главной задачей этой статьи было собрать воедино все часто встречающиеся варианты.

Для начала стоит упомянуть еще раз, что все манипуляции мы будем производить над объектом Respondent. В нашей терминологии респондент – это лицо, которое принимает участие в опросе. В каждого респондента есть свой набор свойств (например, гендерная принадлежность, рассмотренная ранее). Для такого, чтобы показать все многообразие условий, с которыми работает Drools, я каждое свойство вынес в отдельное поле класса Respondent. Для простоты понимания, в качестве еще одного поля я добавил *Map<String, String> result*, в котором я буду собирать все результирующие переменные. Таким образом, класс Respondent будет выглядеть следующим образом:



Самое простое правило по определению гендерной принадлежности мы уже рассмотрели. А что если нам нужно определить половой признак, только у тех респондентов, которые участвуют в исследовании? Для фильтрации такие респондентов мы будет использовать булеву переменную *isActive.* Значение *true* – респондент участвует в исследовании, *false –* нет.



На рисунке выше показано, как мы объединили 2 условия: активность респондента и его половой признак. Не трудно догадаться, что в таком случае происходит объединение правил, находящихся в одной строке, по логическому И. Два условия объединены одним фактов Respondent. Вот так выглядит описание правила на языке DRL:

*rule "name\_for\_RuleTable\_20"*

*when*

*$s: Respondent(isActive == true, gender == "male")*

*then*

*$s.addResult("SEX", "M");*

*end*

*rule "name\_for\_RuleTable\_21"*

*when*

*$s: Respondent(isActive == true, gender == "female")*

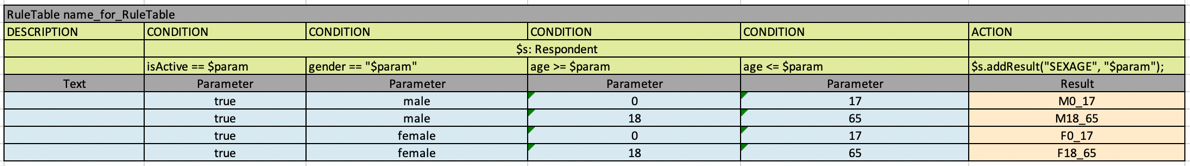
*then*

*$s.addResult("SEX", "F");*

*end*

Стоит отметить об использовании параметра *$param.* Этот параметр работает в рамках одного столбца и во время компиляции правил он будет заменен на конкретное значение из ячейки. Т.е. условие *isActive == $param* будет преобразовано в *isActive == true.* В случае работы с булевыми или целочисленными переменными экранировать его не надо. Движок Drools понимает, что это не строка. В случае работы со строками экранирование обязательно, как показано в примере со свойством gender.

В социологических исследования часто разбивают респондентов на половозрастные группы для того. Следующий пример как раз демонстрирует такое разбиение:



Данное правило производит разбивку активных респондентов по половому признаку, а также на две возрастные категории до 17 лет включительно и после 18 лет включительно. Повторюсь, в случае работы с числами экранирующие кавычки не требуются.