Drools Technical Document

Table of Contents

Last Published March 21, 2016

Purpose

Drools is a Business Rules Management Solution (BRMS) that serves as an alternative rules engine to Kuali Rules Management System (KRMS) for Circulation policies in OLE. This was necessitated owing to KRMS not being able to perform at optimal speeds as data scaled up. Also authoring circulation rules and updating them involved significant work and lacked ease in KRMS.

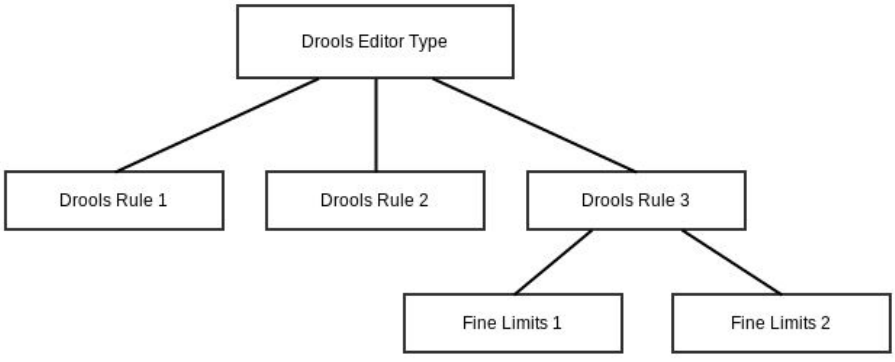
Drools, an open source BRMS, is expected to ease the process of authoring and updating circulation rules and deliver better performance than KRMS. While KRMS would still be available it will not be supported going forward.

Dependencies

The Drools BRMS, unlike the KRMS, doesn't use the database to store rules. However, a few tables are used by OLE to store and retrieve rule related information during the process of generating .drl files. They are listed below

ole_drl_editor_t	Drools Editor
ole_drl_rule_t	Drools Rule
ole_drl_fine_limits_t	Drools Fine Limits

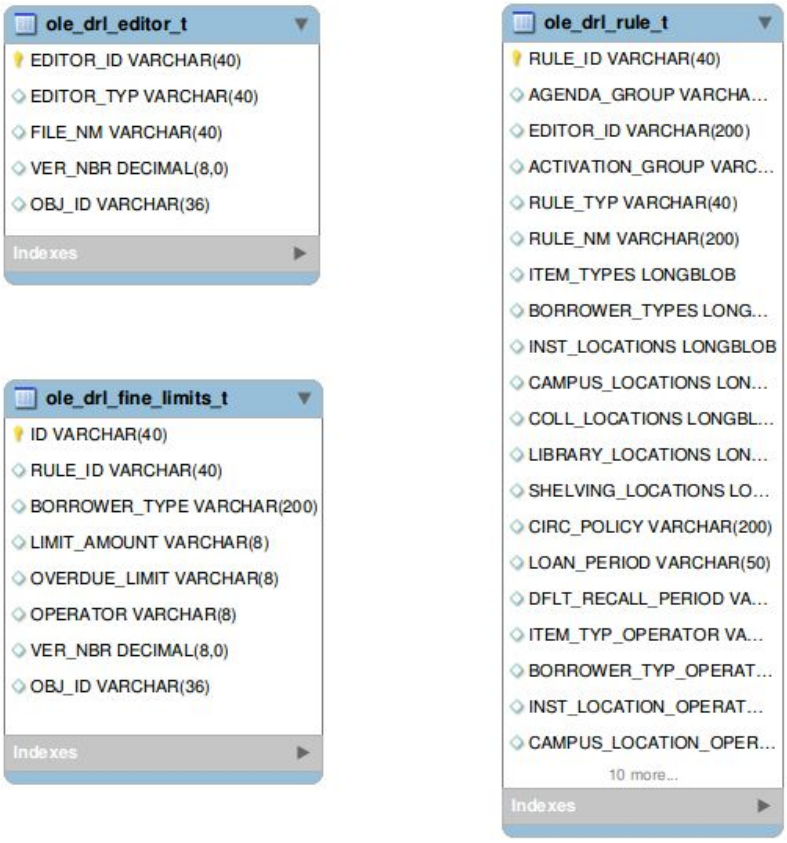
Logical Data Model



The Drools Editor Type table is at the top of the hierarchy and holds the Editor Types. Under each of the Editor Type, multiple rules can be added. Multiple fine limits can reside under a rule. The editor type contains fixed values. This can be equated to the *agenda* in KRMS. Under Editor are rules. Rules would contain various attributes and these attributes are used in making

conditions. Any number of rules can be added under Editor. Fine limits are specified under certain rules, especially, under check-in rules where fines are calculated based on library's overdue policies.

Physical Data Model



The *ole_drl_editor_t* table contains the editor related information. The Editor Type is restricted[1] in OLE to General Check, Check-out, Check-in, Renew, Request and Notice. The *ole_drl_rule_t* table contains rule related information. There can be multiple rules under an editor and there is no restriction on the number of rules that can reside under an editor. The EDITOR_ID field is a foreign key in *ole_drl_rule_t* table. Certain rules, such as those associated with Check-in operation, involve calculation of Fines. The fine related information is stored under the *ole_drl_fine_limits_t* table. RULE_ID field is a foreign key in *ole_drl_fine_limits_t* table.

[1] Drools implementation is ongoing and may incorporate more Editor Types. This needs to be updated appropriately.

Last Published March 21, 2016

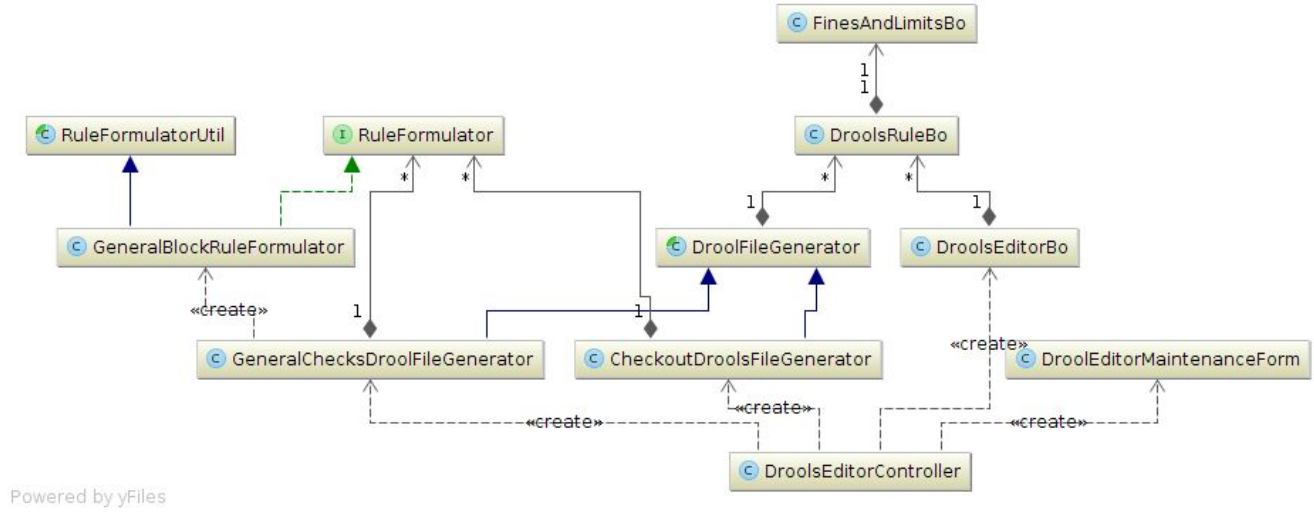
Service Interface Design

Generating a Drools Rule File

Drools BRMS uses a native rule file with a *.drl* extension. Each DRL file can have multiple rules, queries and functions. Apart from this it also holds imports, globals and attributes that are assigned and used by the rules. A typical DRL file follows the below structure

```
package

imports

globals

functions

queries

rules
	rule "<rule-name>"
	when
		<something>
	then
		<something>
	end
```

However, to ease the process of creating the DRL files, OLE, provides the end user with a Graphical User Interface (GUI) where the circulation related policies are fed. This is stored in the database tables and are used to generate DRL files in the background for use by the Drools engine.



The *DroolsEditorController* class controls the whole generation of the DRL file. It uses the *DroolEditorMaintenanceForm* class to collect the rule information from the GUI. This information is used by the *DroolFileGenerator* class which is the abstract class that helps in generating DRL file. The *CheckoutDroolsFileGenerator* and *GeneralChecksDroolFileGenerator* classes use the

RuleFormulator to generate rules[2]. The *DroolsRuleBo, DroolsEditorBo and FinesAndLimitsBo* are the Business Object classes. The Object Relational Mapping to the Database tables is done in *ojb-deliver.xml* file.

The *RuleFormulator* is an interface and each rule in Drools has a Rule Formulator class which extends *RuleFormulatorUtil* and implements *RuleFormulator.* Each Rule Formulator class has a template which sometimes have placeholders which will be populated with data from the User Interface.

Loading functions as part of the Drools Rule File

As already detailed, a DRL file also allows functions and queries other than rules. Since functions are fairly standard across institutions and typically aren't subject to customizations, they reside in the template files which are used by the Rule Formulator classes. For example, the function *today*() resides in the *patron-expiration-date.txt* file used by the Rule Formulator class, *PatronExpiredRuleFormulator.*

NOTE:

- While the DRL file is the most important file for the Drools Engine to work, without data in the database tables, there is no way OLE can pull DRL data for modifications later in GUI.
- Implementers would be better off inserting data both into the database tables and creating DRL files, simultaneously. To maintain data integrity it is advised to use the GUI to load all circulation rules.
- Another alternate way is to enter data into database tables and submitting them from the GUI which would generate DRL files.

Loading the Drools Rules

The *DroolsKieEngine* class takes care of loading the rules from the DRL files. The *initKnowledgeBase* method of the *DroolsKieEngine* class is called from the *OLEInitializeListener* class. This fires up the *populateKnowledgeBase* method where the system parameter, LOAD_CIRC_POLICIES_IND, is checked.

This parameter needs to be set to 'Y' if OLE is expected to ingest the circulation policies. Once the rules are read from the files, the parameter is updated to 'N'. Not just during implementation, whenever the institution makes changes to the rules and wants it to be reloaded, the parameter will need to be set to 'Y'. Whenever a KIESession is established using the *getSession* method, the *populateKnowledgeBase* method is called and the rules are reloaded depending on the parameter.

[2] As Drools evolves more such File Generators would be used as required.

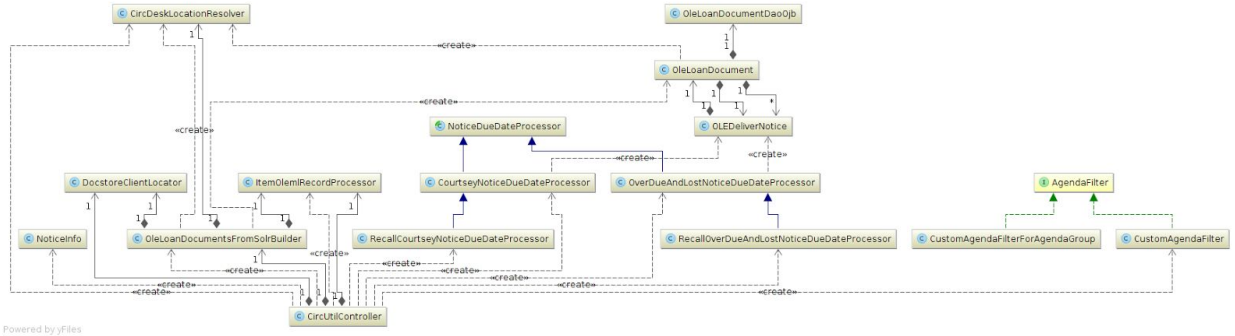
The location in which the DRL files are kept is configured in the *olefs-config-defaults.xml* file under the *rules.directory* parameter. This is usually a folder named rules. It may contain .drl files or subfolder with the .drl files.

```
<!--Drools rule file directory-->
  <param name="rules.directory">${project.home}/rules</param>
```

The *loadRules* method of the *DroolsKieEngine* class looks for files with DRL extension (.drl) in the directory and loads the rules. Following this *readRules* method creates a new KIEContainer. The *updateParameter* method is then called to update the parameter to 'N' to prevent repeated loading of circulation rules.
When the rules are to be fired, a KIESession is created from the KIEContainer and a *fireAllRules* method is called.

Notice Generation in Drools

Notice generation is vital in OLE and was handled in the erstwhile KRMS. Drools had allowed for a more comprehensive notice generation rules, easier to configure and faster to process. The various notices that can be configured to be sent in OLE includes Courtesy Notice, Overdue Notice, Lost Notice, Hold Courtesy Notice, Recall Notice, Pickup Notice and On Hold Notice.

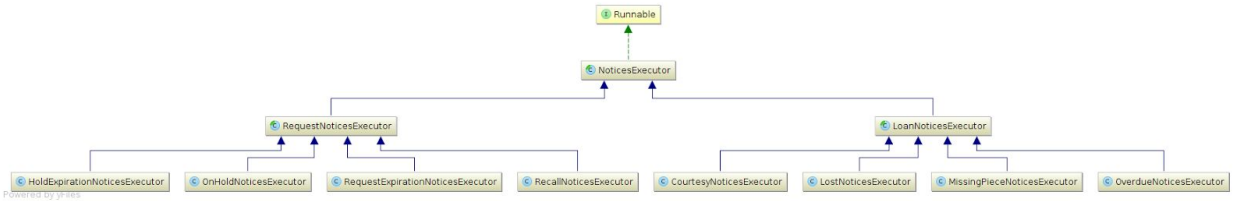


The *processNotices* method of the *CircUtilController* class is being called across classes to set in motion the generation of notices. The method retrieves the notice type of the loan document and loads them into the notice info business object. Similarly, the item record details are also retrieved. Both notice info and item record are made as a List and the *fireRules* method is called.

The overdue notices and lost notices are handled by the *OverDueAndLostNoticeDueDateProcessor* class in their *generateNotices* method. The courtesy notice is handled in the *CourtseyNoticeDueDateProcessor* [sic] class, the recall courtesy notice in the *RecallCourtseyNoticeDueDateProcessor* [sic] class and the recall overdue notice in the *RecallOverDueAndLostNoticeDueDateProcessor* class. The classes extend the *NoticeDueDateProcessor* abstract class.

Notice Content Configuration

With Drools, notices became more customizable. This led to maintaining various notice content configurations as maintenance documents. Earlier the notice contents were configured as only System Parameters.



The system now looks for content data from both the System Parameter and the Maintenance document. If the maintenance record data is empty it will build the content from data in the system parameters. the *setOleNoticeContentConfigurationBo* method of the various NoticesExecutor classes is involved in building the notice content business object either from the maintenance data or the system parameter.

Service Interface Design (REST/SOAP)

Not applicable.

User Interface Design

The Drools Editor uses KRAD's UIF (User Interface Framework). A very good guide on this can be found here.

Data Importing

Drools doesn't use the database, hence data importing is not applicable. However, OLE maintains tables to hold circulation rules related data for generating DRL files. This is held in a RDBMS and can be loaded through SQL or other batch uploads supported by the database. However, to generate DRL files, the data from the database needs to be retrieved through the GUI and submitted.

Data Exporting

The circulation policy rules are maintained as DRLs. However, OLE maintains the rules in database tables which can be exported through SQL queries.

Last Published March 21, 2016

Workflow

Not applicable.

System Parameters

Namespace Code	Parameter Name	Description
OLE-DLVR	LOAD_CIRC_POLICIES_IND	The parameter value is set to 'Y' to have OLE ingest the default circulation policies upon next policy evaluation.
OLE-DLVR	CANCELLATION_NOTICE	It just holds the notice type as cancellation notice.
OLE-DLVR	CHUNK_SIZE_FOR_NOTICE_CONTENT_REINDEX	This parameter value is used to specify the chunk size for notice content reindexer
OLE-DLVR	COURTESY_NOTICE_CONTENT	The following item(s) on loan to you are about to become due. Please return by or before the due date in order to avoid any penalties
OLE-DLVR	COURTESY_NOTICE_INTER	Setting interval for courtesy notice
OLE-DLVR	COURTESY_NOTICE_TO_DATE	The courtesy notices will be send to the patron who are having the courtesy date falling on or before the date specified in this parameter while running the notice job - date format mm/dd/yyyy
OLE-DLVR	COURTESY_NOTICE_TYPE	It just holds the notice type as Courtesy notice.
OLE-DLVR	DELIVER_NOTICE_FROM_ADDRESS	This is for setting the from mail address for the notices related deliver module
OLE-DLVR	EXP_HOLD_NOTICE_CONTENT	Expired Hold Notice Body
OLE-DLVR	HOLDCOURTESY_NOTICE_TYPE	Holds the notice type.
OLE-DLVR	LOST_NOTICE_TO_DATE	The replacement fee will be generated and item status is updated to lost for the items have the lost

		date falling on or before the date specified in this parameter - date format mm/dd/yyyy
OLE-DLVR	NCIP_ACCEPT_ITEM_NOTICE_INDICATOR	This parameter decides whether a pick up notice need to be send to the patron who requested for that item at the time of accept item service.Allowed values are Y and N .By default it will send pickup notice
OLE-DLVR	NOTICE_FROM_ADDRESS	This is for setting the from mail address for the notices related deliver module
OLE-DLVR	NOTICE_PERIOD	This is for configuring the default notice period while changing the configuration in admin tab
OLE-DLVR	NOTICE_THREAD_POOL_SIZE	This parameter provides the number of parallel execution to be applied on the notice generation process
OLE-DLVR	ON_HOLD_NOTICE_ITEM_STATUS	This parameter defines the item status code to which the notice to be send to the patron when the on hold notice job runs.';' act as record separator for each item status
OLE-DLVR	ON_HOLD_NOTICE_REQUEST_TYPE	This parameter defines the request type codes to which the notice to be send to the patron when the on hold notice job runs.';' act as record separator for each request type
OLE-DLVR	ONHOLD_NOTICE_TYPE	Holds the notice type
OLE-DLVR	OVERDUE_NOTICE_CONTENT	Overdue Notice Body
OLE-DLVR	OVERDUE_NOTICE_INTER	Setting interval for overdue notice
OLE-DLVR	OVERDUE_NOTICE_TO_DATE	The overdue notices will be send to the patron who are having the overdue date falling on or before the date specified in this parameter while running the notice job - date format mm/dd/yyyy
OLE-DLVR	OVERDUE_NOTICE_TYPE	Holds the notice type
OLE-DLVR	RECALL_NOTICE_TYPE	Holds the notice type

OLE-DLVR	RQST_EXPR_NOTICE_TYPE	Holds the notice type
OLE-DLVR	SEND_ONHOLD_NOTICE_WHILE_CHECKIN	This parameter value is used for configuring the system to send onhold notice while checkin an item
OLE-DLVR	DEFAULT_TIME_FOR_DUE_DATE	This parameter is for providing the default time whenever the time field is left blank when altering the due date of an item. Give the time in the format HH:MM:SS (24 Hour Format)
OLE-DLVR	GRACE_PERIOD_FOR_NON_WORKING_HOURS	This parameter value is used for configuring the grace time for the patron to return item.
OLE-DLVR	INCLUDE_NON_WORKING_HRS	This parameter value is used for configuring non working hours to be excluded.

Roles and Permissions

Not applicable.

Debugging Help

Adding the following snippet to the log4j.xml file (under /src/main/resources) will output details to the log which can be useful when trying to debug rules.

```
<category name="org.drools">
<priority value="TRACE"/>
</category>
```

Last Published March 21, 2016