

# **Business Rule Maintenance Tool**

## **Web Application Documentation**

Version 0.8

## Content

Overview.....	4
Ruleengine Use Cases.....	5
Prerequisites.....	5
Installation.....	6
Start Page.....	7
Menu.....	7
Start Page.....	8
Password.....	8
Login/Logout.....	8
Projects.....	8
History.....	8
Activity.....	8
Search.....	9
Export File.....	9
Import File.....	9
Checks.....	9
Actions.....	9
Files.....	11
Groups.....	11
Users.....	11
Configuration.....	11
Configure Database.....	12
Documentation.....	12
About.....	12
Concepts.....	13
Group (Role) Management.....	13
User Management.....	13
Project Management.....	14
Reference Fields.....	14
Checks.....	15
Configuration.....	17
Import Project.....	17
Export Project.....	18
History.....	18
User Avatar.....	18
Help.....	19
Projects.....	19
Rulegroups.....	20
Subgroups.....	20
Rules.....	21
Actions.....	21

Rulegroup Testing.....	21
Pentaho ETL tool integration.....	24
Workflow.....	25
Preparations.....	25
Create a project.....	25
Create a rulegroup.....	26
Create a rule.....	26
More rules.....	27
More subgroups.....	27
Create an action.....	27
More actions.....	28
Testing a Rulegroup.....	28
Export project.....	29
Executing rules and actions.....	29
Software Updates.....	30
Extending the Ruleengine.....	31
Extending the Web Application.....	32
Software components.....	33
Contact.....	34
Links.....	35
Defaults.....	36

## **Overview**

The Business Rules Maintenance web application allows to create, update and delete projects, rulegroups, subgroups, rules and actions. These are used to define and orchestrate a logic with the purpose to execute it against a given set of data.

The web application exports a project to a zip file which can directly be executed with JaRE – Java Rule Engine.

The ruleengine JaRE is written in Java and can run either standalone from the command line or may be embedded in other Java related projects. It may be embedded in other script languages or web applications.

The main goal of using a business rules and ruleengine approach is to separate the IT logic from the business logic. This way both types of logic can be managed individually and by the relevant domain experts: IT expert or business expert. This creates a proper division of responsibilities, makes IT code cleaner and easier to maintain and thus adds to the overall quality and agility of the system.

By separating IT and business logic, the business user is not confronted with business rules which are mixed with complex IT code or work flows. This enhances the transparency for the user. It supports the idea of implementing changes to IT code in an agile manner and overall enhances the quality of the system.

This document explains the installation, configuration, concepts and usage of the web application.

## **Ruleengine Use Cases**

The ruleengine can be used to:

- Filter (remove, route) data based on rule engine results
- Manipulate/update data based on rule engine results
- Calculate values based on rule engine results
- Create detailed logs of the results for further processing or debugging
- Create KPI calculations from data
- Quality checks of data
- Check results of test cases

## **Prerequisites**

To use the web application following prerequisites have to be met:

- Java: The Business Rules maintenance web application is written in Java. It comes in the form of a .war file (web archive).
- Apache Tomcat: Apache Tomcat allows to run Java code on the server. Other web servers with similar capabilities that allow to use .war files may also be used.
- MySQL/MariaDB: The web application stores configuration and all data entered by the user in a MySQL database. Other dialects of MySQL such as MariaDB may also be used.
- A web browser. Parts of the application use JavaScript and JQuery to enhance the user experience.

It is assumed that the user – before running the Business Rules Maintenance web application - has a running instance of Java, Tomcat and MySQL already in place.

The web application uses the JaRE Java library (jar archive file). In case you want to use/install a newer version of this library see the section “Software Updates”.

## Installation

### 1. MySQL/MariaDB: database setup

The database, tables and some default data is created from within the web application. It is not required to manually import the database schema.

But a running MySQL or MariaDB server is required. To setup the database, tables and some default data a user and user password is required who has sufficient privileges to create the database.

### 2. Apache Tomcat: install the web application

Make sure that Apache Tomcat is actually running. Download the Business Rules Maintenance web application .war file.

Locate where in your system Apache Tomcat is installed. It contains a folder named “webapps”. Copy the .war file into the “webapps” folder. Usually Tomcat automatically expands the war file (installs it) and the web application is immediately available. If this is not the case you might have to restart Tomcat.

Open a web browser window and enter following URL (address):

*[http://localhost:\[port\]/rule\\_maintenance/](http://localhost:[port]/rule_maintenance/)*

If Tomcat is running on a different server (not localhost), replace “localhost” with the server ip address, the hostname or a DNS anme for that server. Tomcat usually runs per default on port 8080. If unsure, check with your Tomcat configuration, which port is used.

Next you will be presented with the login page. If not, click on “Login” in the menu on the left-hand side. Now enter the userid and password for the web application. The default userid for the administrator is: *admin* and the password is also: *admin*.

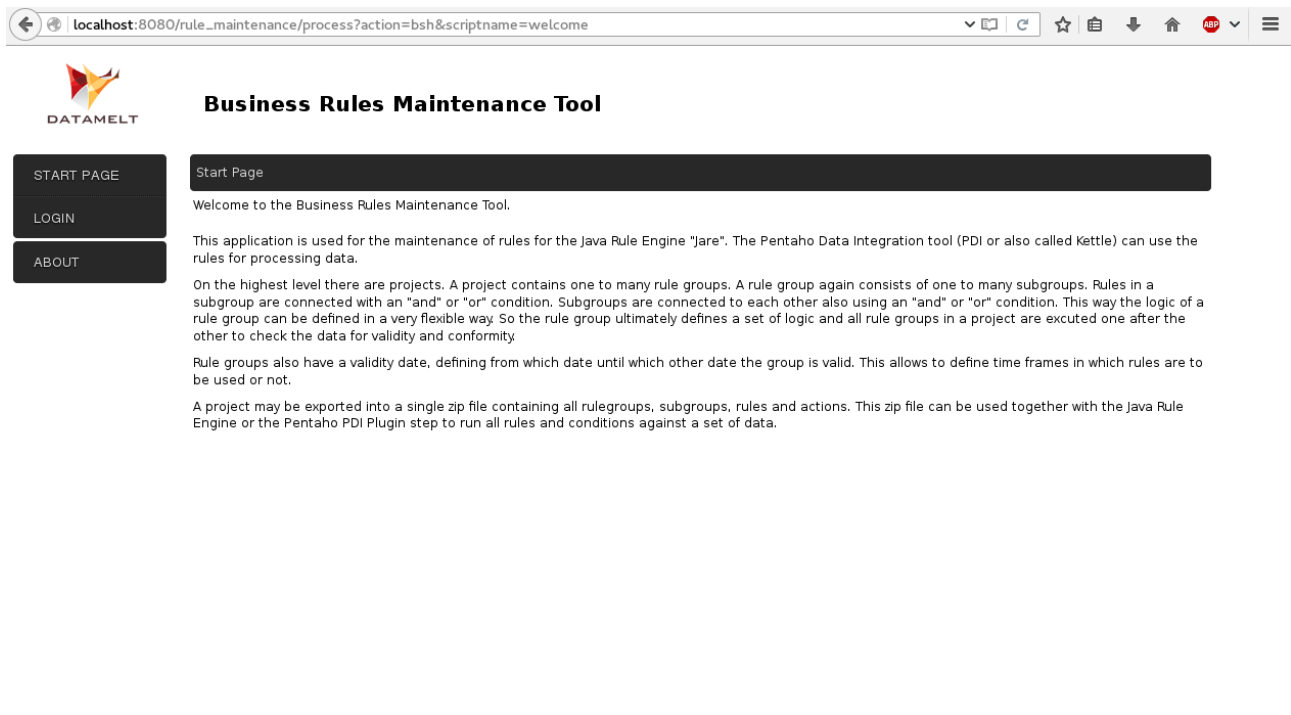
### 3. First run of the web application

When the application is run the first time, only the admin user can login. Enter the admin userid and password for the web application.

The database user that is used for all database access by the web application needs sufficient privileges to create the database and to write to and read from the database and it's tables.

## Start Page

Below is a screenshot and overview of the web application Start Page.



*Screenshot 1: Application Start Page*

## Menu

The menu is located on the left side of the web application and is always visible. It allows the user to navigate through the application.

The actual menu items displayed will vary depending on if the user is logged in or not and if the user is assigned to the "Admin" group or not – some of the menu items are only available to admin users.

Below find a description of the individual menu items

## Start Page

The Start Page is the first page a user is directed to after starting the web application or logging into the application. It shows a basic introduction to the purpose of the application.

## Password

Allows to change the password of the user. The user has to provide the current password and then twice a new password.

If the application is configured to use LDAP, then a change of the password from within the application is not possible.

## Login/Logout

The web application requires users to enter a userid and password to authenticate. On login it is determined to which groups (roles) the user is assigned. The user settings steer to which project users have access or which project they see.

Users may register with the application. The user will receive an email with a link. Once the user clicks the link, he will be forwarded to the web application and can finalize the registration process

## Projects

“Projects” is the central place to work with projects, to create, update or delete rulegroups, subgroups, rules and actions.

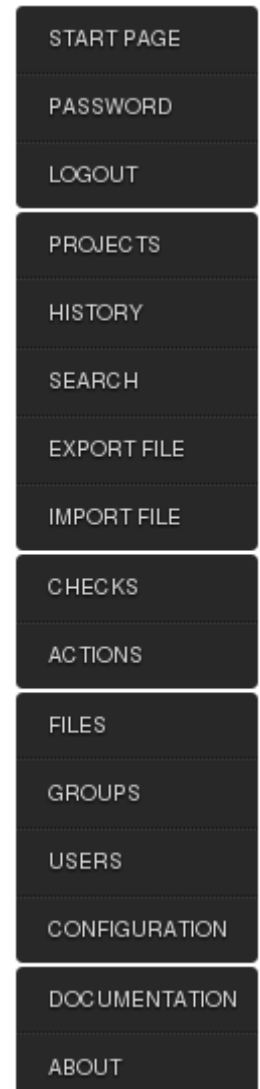
A Project is the highest element in the hierarchy. A project may contain zero, one or multiple rulegroups. Rulegroups may contain zero, one or multiple subgroups. Rulegroups also may contain zero, one or multiple actions. Subgroups contain zero, one or multiple business rules.

## History

The history displays all elements that have recently been changed by the user – projects, rulegroups, subgroups, rules and actions. It allows the user to quickly re-access these elements.

## Activity

Clicking on this menu entry will display a list of the recent activities. Activities include



Screenshot 1:  
Application Menu



adding/changing deleting projects, import/export of projects, user management and more.

The relevant activity, the user who triggered the activity and a time stamp is displayed in reverse chronological order.

## Search

The search page allows to search for rules and actions or to search for fields that have been used in rules or actions. The name and description of the rules and actions is searched for the given search term.

The results displayed allows to directly jump to the project and rulegroup where the rule or action is located.

## Export File

Allows to export a project. The file created is a regular zip file and contains all rulegroups, subgroups, rules and actions for a given project. The resulting zip file can be used directly with the ruleengine to execute the rules.

When a project is exported a validity date has to be specified. Only groups valid on or after the specified date will be included in the export.

## Import File

Allows to import a project that previously has been exported. The user has to specify the name and description for the new project and the owner group of the imported project.

## Checks

Rules use predefined checks to achieve their task. One rule uses exactly one check at a time. By combining rules with different checks a complex logic can be built.

An example for a check is „is equal to“, „is greater than“ or „is not null“. Currently there are 40+ different checks available.

The list of checks provides links to a list of methods that are defined for each check. There is also a link to the documentation of the checks.

## Actions

Actions allow to execute certain actions based on if the rule group - as a whole - failed or if it passed. Or to always execute the action independent of the result of the rulegroup.

Actions may update or calculate data or perform other tasks.

The list of actions provides links to a list of methods that are defined for each action and there is also a link to the documentation of the actions.

## Files

Visible only for users in the *Admin* group. The web application uses Beanshell scripts and Apache Velocity templates to separate the programming logic of the web application from its representation on the screen.

In the Files section a user with administrative rights can change the web application code or display on the fly.

## Groups

Visible only for users in the *Admin* group. Groups define a collection of users that have the same access rights to projects. Different groups can contain different users and these can have access to multiple projects.

## Users

Visible only for users in the *Admin* group. A user can be assigned to one or multiple groups. A user needs to authenticate to the application to be able to use it.

Users may be deactivated. In this state they can not log in to the web application, but all data is preserved. In a second step the user may be deleted. This will completely remove all data belonging to the user, such as e.g. the history.

## Configuration

Visible only for users in the *Admin* group. The configuration defines the details of the connection of the web application to the database and optional LDAP settings.

For the application to connect to the underlying database, it needs information on

- the database server host – specify the hostname or IP address
- the database server port used
- the database name
- the userid of a user with read and write privileges to the database
- the users password

You may use an LDAP server to verify the credentials of a user. Specify:

- LDAP server hostname or IP address
- LDAP server port

- LDAP Server domain name

The process will first try to authenticate the user with the specified userid to the LDAP server and if this fails it will try to authenticate the user to the database. If no LDAP server is defined it will authenticate the user to the database only.

## Configure Database

The configuration page is used to specify all required details to access the MySQL or MariaDB server.

It can also be used to switch to a different (existing) database, by specifying the name of the existing database.

It can be used to create a new database for the web application by specifying a different database name. The user is asked if the database, tables and the default data shall be created.

In any case, no existing data is overwritten or deleted.

## Documentation

A link to documents relevant to the web application and the ruleengine in general.

## About

The about page shows information about the web application, the web application version and the rule engine version and other details.

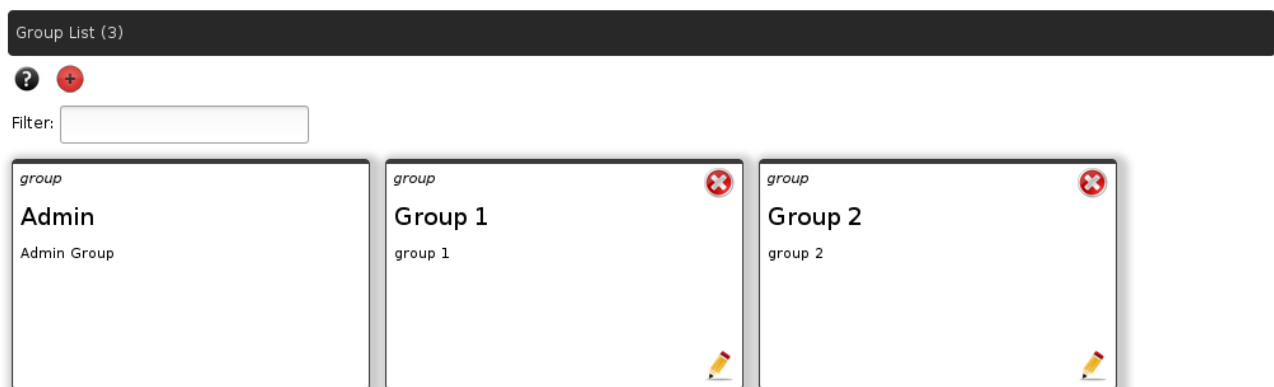
## Concepts

This section explains the concepts that build the base for this web application. It is also related to the concept of the ruleengine that is used when the rules are executed against a set of data.

### Group (Role) Management

The web application allows to define and modify groups. Users can be assigned to one or more groups. The settings for a project allow the definition to which group the project belongs. All users that are assigned to this group may view and update all details of the project, rulegroups, subgroups, rules and actions. Users not being part of the group have a read-only access to all the mentioned parts.

The group *Admin* can not be updated or deleted, as it is required by the system. Users that are assigned to this group have admin privileges for the web application.



Screenshot 2: List of groups

### User Management

The web application allows to define and update users. Users can be assigned to one or multiple groups. See „Group Management“ for more details.

Users are added or modified by the administrator or a person which is in the “Admin” group.

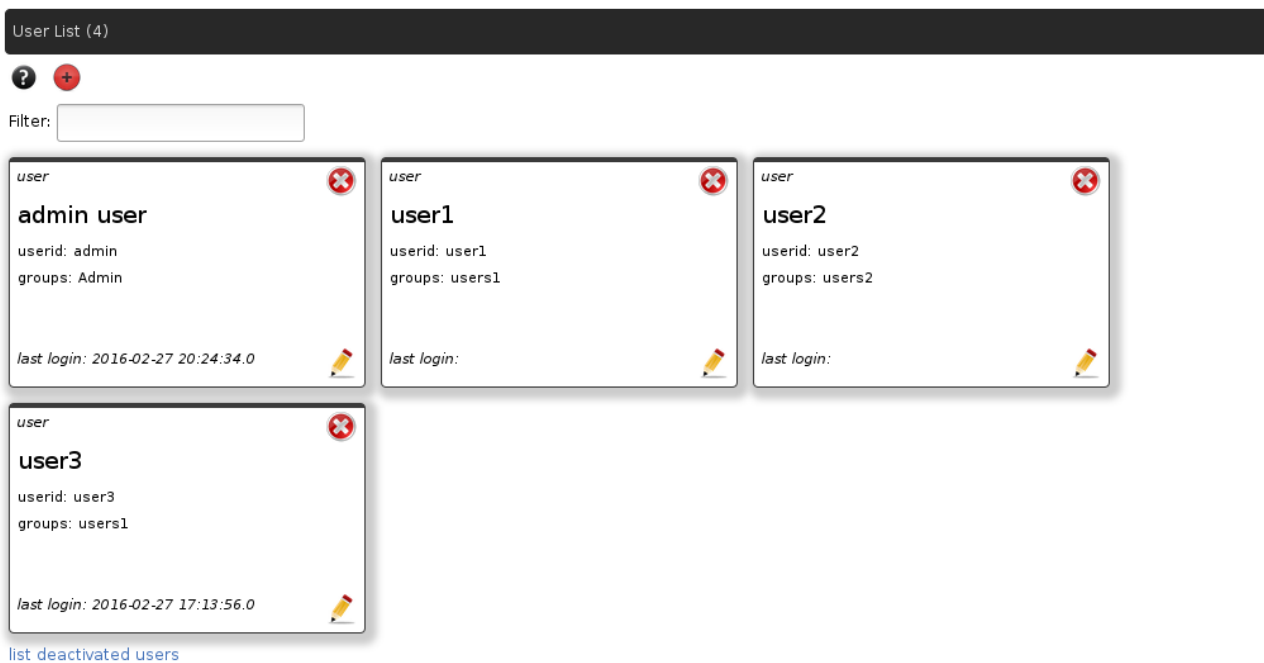
Each user that is added will have access to the web application and the groups and in turn projects that are assigned to her/him.

Users may be deactivated. A deactivated user can not login to the web application anymore

but all relevant data of the user such as the history is conserved. When the user is finally deleted, all user related data is completely deleted and cannot be restored.

The deactivated user can also be reactivated, so that he/she can again login the web application.

In the configuration dialog a user with administrative privileges may configure the settings of a LDAP host. On successful setup, users will be authenticated against LDAP. Therefore the userid in this web application and the userid of the user in LDAP have to be the same.



User List (4)

Filter:

user	userid	groups	last login
admin user	admin	Admin	2016-02-27 20:24:34.0
user1	user1	users1	
user2	user2	users2	
user3	user3	users1	2016-02-27 17:13:56.0

[list deactivated users](#)

Screenshot 3: List of users

## Project Management

Projects are containers for rulegroups, subgroups, rules and actions. They group all parts together. When a project is created it has to be assigned to a group. Users assigned to this group have read and write access to all parts of the project. Other users have read-only access to all parts of the project.

If a project is marked as private project, only users assigned to the group the project belongs to have access to the project. Other users don't see the project.

Projects may be copied. All content of the project is copied into a new project. Also, projects may be exported into a zip file and they also may be imported back into a new project.

## Reference Fields

Reference fields can be assigned to a project. A field is defined by its name, descriptive name, description and the type. They define the fields – and corresponding to source data fields- that are required for the rules and actions and serve following purposes:

- The user does not have to type in field names and the relevant field type
- It serves as a documentation for business users which fields are available
- It supports the testing of the rulegroups in the web application

Reference fields may be defined manually by entering the details of each field. Alternatively a list of fields may be imported from a CSV file. The CSV file needs to have four columns:

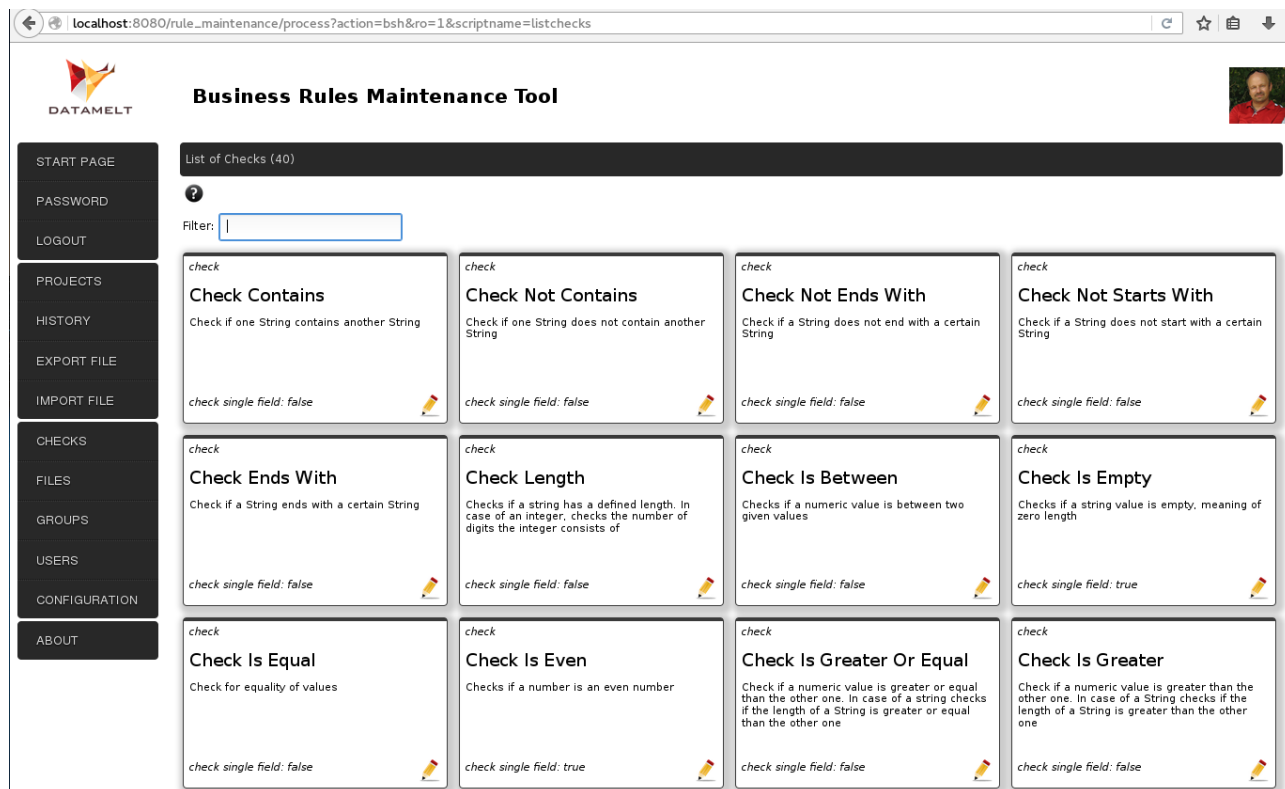
- Column 1: the name of the field. There shall not be duplicates of the same name in one CSV file
- Column 2: the descriptive name of the field
- Column 3: the description of the field
- Column 4: the type of the field. Valid types are: string, integer, float, double, boolean, long, bigdecimal or date

The fields in the CSV file have to be separated from each other by a semicolon. Blank rows or rows starting with the character “#” are ignored during the import of the data.

## Checks

A check is an individual module that is designed to fulfill one specific test. For example one check is “is equal to”. It tests, if the given data is equal to the expected result. One single check usually has the ability to use different types of data such as textual data, data of type date or numeric data.

There are currently about 40 checks available. They are referenced by the rules and by combining several checks a complex logic can be built to test for certain conditions of the data.



The screenshot displays the 'Business Rules Maintenance Tool' interface. On the left is a sidebar with navigation links: START PAGE, PASSWORD, LOGOUT, PROJECTS, HISTORY, EXPORT FILE, IMPORT FILE, CHECKS, FILES, GROUPS, USERS, CONFIGURATION, and ABOUT. The main area is titled 'Business Rules Maintenance Tool' and shows a 'List of Checks (40)'. A search filter is present. Below the filter, 12 check cards are displayed in a grid. Each card includes a title, a description, and a status indicator (e.g., 'check single field: false').

Check Name	Description	Status
Check Contains	Check if one String contains another String	check single field: false
Check Not Contains	Check if one String does not contain another String	check single field: false
Check Not Ends With	Check if a String does not end with a certain String	check single field: false
Check Not Starts With	Check if a String does not start with a certain String	check single field: false
Check Ends With	Check if a String ends with a certain String	check single field: false
Check Length	Checks if a string has a defined length. In case of an integer, checks the number of digits the integer consists of	check single field: false
Check Is Between	Checks if a numeric value is between two given values	check single field: false
Check Is Empty	Checks if a string value is empty, meaning of zero length	check single field: true
Check Is Equal	Check for equality of values	check single field: false
Check Is Even	Checks if a number is an even number	check single field: true
Check Is Greater Or Equal	Check if a numeric value is greater or equal than the other one. In case of a string checks if the length of a String is greater or equal than the other one	check single field: false
Check Is Greater	Check if a numeric value is greater than the other one. In case of a String checks if the length of a String is greater than the other one	check single field: false

Screenshot 4: List of checks

The ruleengine can be extended by implementing new checks or by extending the existing ones. Please read the appropriate documentation on how to create your own checks.



## Configuration

### Configuration of the Rules Maintenance Tool

The Business Rule Maintenance tool uses a MySQL database for data storage. Specify the database hostname/IP address and port, database name and a user who has read and write access privileges to the database.

Optionally you can specify the LDAP server details below, to authenticate a user login against the LDAP server. Provide the LDAP server hostname or IP address and the port and the domain name of the server (e.g. mydomain.com).

Database Server Hostname or IP Address:	<input type="text" value="localhost"/>
Database Server Port:	<input type="text" value="3306"/>
Database Name:	<input type="text" value="ruleengine_rules"/>
Database User:	<input type="text" value="root"/>
Database User Password:	<input type="password" value="••••••••"/>
Pentaho PDI Plugins Folder:	<input type="text"/>
LDAP Server Hostname or IP Address:	<input type="text"/>
LDAP Server Port:	<input type="text"/>
LDAP Server Domain Name:	<input type="text"/>
<input type="button" value="save"/> <input type="button" value="cancel"/>	

Screenshot 5: Configuration

The configuration page allows to define the basic settings for the application. An administrator may update the settings for accessing the database and the LDAP settings. Additionally an export folder and a backup folder can be defined where exported projects will be stored. For user self-registration the SMTP server details can be defined.

## Import Project

A project – in the form of a zip file – that has previously been exported using this web application can be imported. This allows the exchange of project files and definitions between different users or across different companies.

Import Rule Maintenance Zip file



Project Name:	<input type="text"/>
Project Description:	<input type="text"/>
Zip File to import:	<input type="button" value="Durchsuchen..."/> Keine Datei ausgewählt.
Owner group:	<input type="text" value="v"/>
Private project:	<input type="checkbox"/>
<input type="button" value="import"/>	



Screenshot 6: Import project

If a project is exported to a file, it can be safely deleted and then re-created by importing the project file. There is no data loss in this process.

## Export Project

A project may be exported. In the process a zip file is created that can directly (without modifications) be run with the JaRE ruleengine – either standalone, using Pentaho PDI or running the ruleengine serverside.

When exporting, the user has to select the project and enter a validity date for which he/she wants to do the export. The validity date steers which rulegroups are exported. Rulegroups have a start and end validity. Only those rulegroups are exported which are valid on or after the specified validity export date.

If in the configuration an export folder is defined, then projects will be exported to this location. In any case a backup file is created in case the same project file already exists.



Screenshot 7: Export project

## History

Projects can contain a large number of rulegroups, subgroups, rules and actions. This makes it time consuming to locate a certain item. The History lists elements that lately have been updated by the user that is logged in. The user won't see the history of other users.

An entry in the history can be clicked to directly jump to the project or rulegroup where the entry is located.

The user may clear his/her own history.

## User Avatar

The user may choose a picture for his/her representation. It is shown on the upper right-hand side of the web application. Clicking on the picture – or the default picture if undefined – will allow the user to upload a picture to be used as the users avatar.

If no specific picture is used, a default picture will be used instead.



Screenshot 8: Change User avatar

## Help

In various screens of the web application, there is help available identified by a little icon with a question mark inside:



Click on the icon to display the corresponding help text. Clicking on it again will hide the help text.

## Projects

Project is the highest elements in the hierarchy. A project may contain zero, one or multiple rulegroups. Rulegroups may contain zero, one or multiple subgroups. Rulegroups also may contain zero, one or multiple actions. Subgroups contain zero, one or multiple rules.

Projects belong to exactly one owner group. All members of this group can read, add, update and delete all content of the project. If the project is marked as being “private”, users that are not members of the group can not see the project.

Reference fields for the project may be setup manually or alternatively can be imported from a CSV file. This makes the process of writing rules and actions easier for the end user and also serves the purpose of documenting which fields are available and the data type of the fields.



Screenshot 9: List of projects

## Rulegroups

Rulegroups are the containers for subgroups, rules and actions. They have a validity start and end date defined. It defines in which time-frame a certain rulegroup and all its content is valid. This allows to define a logic that is automatically valid at a certain date - so you can plan ahead - and later on automatically expires and is not used anymore.

Rulegroups that are not valid on the date the ruleengine runs are not processed!

A rulegroup may depend on another rulegroup. The user may select if it depends on the passed or on the failed status of the group it depends on. When a rulegroup is dependent on another group and the result of the group does not correspond to the dependency setting (passed or failed), then the rulegroup, the rules and the actions are not executed.

The validity dates of a rulegroup that depends on another rulegroup needs to lie within the validity dates of the rulegroup it depends on.

On the one hand dependencies help to define a central rulegroup with general conditions that one or multiple other groups depend on. This allows a clearer design of the overall project. On the other hand it is a performance benefit. Dependent groups, rules and actions are not executed at all if the dependency is not correctly met.

Note that you can only establish a dependency of one group to another group, if this other group does not already have a dependency itself to another group. This avoids the problem of creating circular dependencies and the project and rulegroups do not get over-complicated and unmanageable.

## Subgroups

Subgroups are the containers for rules. Within each subgroup all rules are connected using

the same condition: either “and” or “or” - you can not use a mixture. Subgroups are connected to each other (to the previous subgroup) also either by “and” or “or”.

The combination of “and” or “or” in the subgroup plus the combination of “and” or “or” between the subgroups gives the user the flexibility to define any complex logic.

## Rules

The rules are the working horses of the ruleengine. On execution data is passed to the rule, the rule is executed and the result is a “passed” or “failed” state. All rules in a rulegroup form a group of logic using “and” or “or” conditions.

## Actions

Actions belong to the rulegroup they are defined in. When a rulegroup – and all its rules – is executed, the result is either a “passed” or “failed” state. An action can run based on this state and execute e.g. an update to the data, do calculations and modify the data, send messages, log information and more.

A rulegroup may have zero, one or many actions. The actions are executed one after the other in the order they have been defined.

## Rulegroup Testing

The web application allows the user to test the rules and actions of an individual rulegroup. Testing is based on a rulegroup and so one rulegroup can be tested at a time.

To enable testing of a rulegroup, reference fields for a project need to be added. This can be done either manually by typing them in or by loading the reference fields from a CSV file.

When running the test for a rulegroup, all rules of the group are executed against the data that the user enters and the result is presented to the user. A check mark or a red cross indicates if the rules passed the tests and if the rulegroup as a whole passed or failed. Additionally the resulting rule message is displayed.

Actions that have been executed or not executed are also displayed with the relevant check marks. Those actions that were executed display the value of the field, that they updated.

Note: If multiple actions manipulate the same field, then all actions will display the same resulting value (final value) of the field.

Below screen print marks the test icon. Click the icon to start the test run.

Rules List - Project: [calculation\\_sample](#) - Rule Group: rulegroup 1

? ◀ S A

<b>project</b> <b>calculation_sample</b> calculation_sample	<b>rulegroup</b> <b>rulegroup 1</b> rulegroup 1 valid: 2016-02-26/2020-12-31
<b>subgroup</b> <b>subgroup_1</b> subgroup 1 rules connector: <b>and</b>	<b>rule</b> <b>check financial group</b> field [financialgroup_code] is not equal to value [XYZ]
	<b>rule</b> <b>check insurance company</b> check insurance company field [insurance_company] is not equal to value [ABC]

Next the user is presented with a page with all fields which are used or updated by rules or actions. Fill in the fields with the appropriate values for the test and click on „test data“.

Note: Reference fields of the relevant project that are not used in any of the rules or actions of the rulegroup will not be available for the testing cycle.

Rules List - Project: [calculation\\_sample](#) - Rule Group: rulegroup 1

? ◀ S A

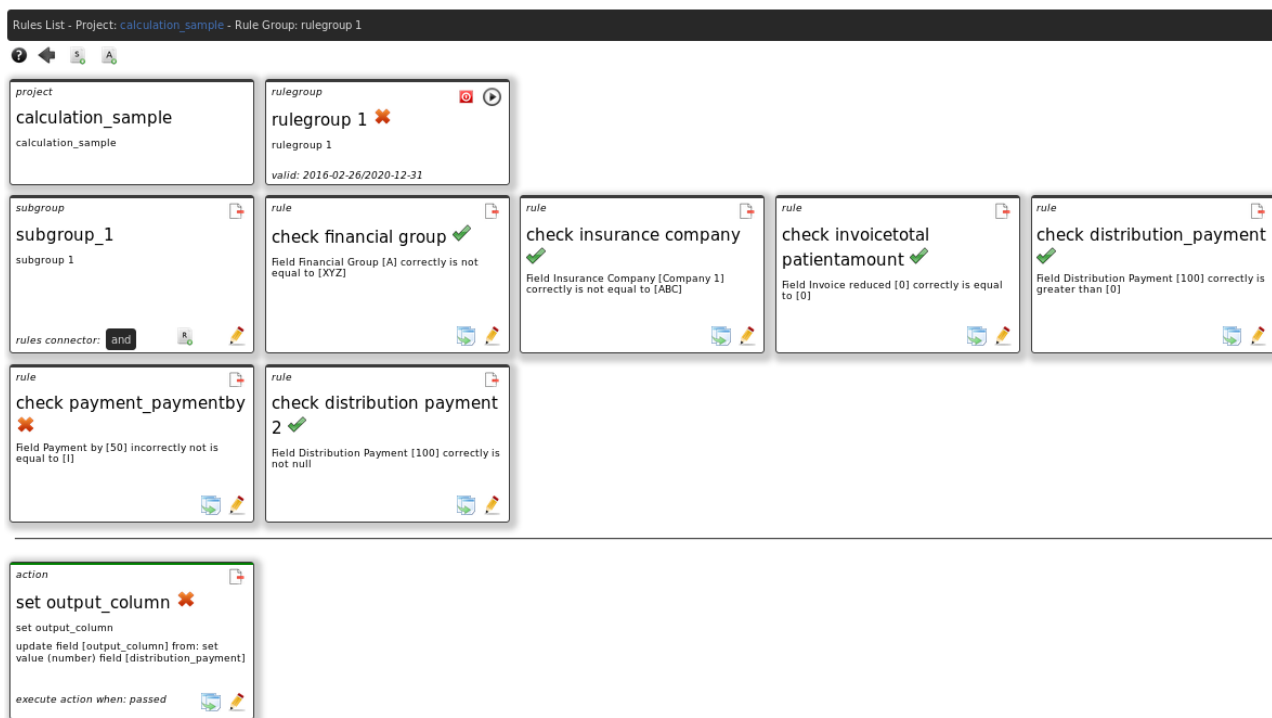
Below find a list of all fields that are referenced in rules or actions of this rule group. Fields not used by any of the rules and actions are not shown. To test the rule group, enter test values for all displayed fields.

distribution_payment (long):	<input type="text"/>
financialgroup_code (string):	<input type="text"/>
insurance_company (string):	<input type="text"/>
invoicetotal_patientamount (long):	<input type="text"/>
payment_paymentby (string):	<input type="text"/>

- Enter "string" values as regular text
- Enter "date" values in the format yyyy-MM-dd (e.g. 2016-05-25) or select them from the calendar
- Enter "integer" and "long" values as numbers without decimal point and precision (e.g. 12345)
- Enter "float" and "double" values as numbers with decimal point and precision (e.g. 567.89)
- Select "boolean" values from the dropdown list

Once the test run is completed the user is presented with the results: The rulegroup is marked as passed or failed and so are the rules. The rules show the corresponding message that was generated. Actions that were executed also show the relevant check mark.

Rules List - Project: calculation\_sample - Rule Group: rulegroup 1



The screenshot displays a hierarchical view of business rules and their test results. The interface is organized into a grid of cards. At the top, a header bar indicates the current context: 'Rules List - Project: calculation\_sample - Rule Group: rulegroup 1'. Below this, the first row shows the 'project' card for 'calculation\_sample' and the 'rulegroup' card for 'rulegroup 1', which is marked with a red 'X' indicating a failed test. The 'rulegroup' card also shows a validity period: 'valid: 2016-02-26/2020-12-31'. The second row contains five 'rule' cards. The first rule, 'check financial group', is marked with a green checkmark. The second rule, 'check insurance company', is also marked with a green checkmark. The third rule, 'check invoicetotal patientamount', is marked with a green checkmark. The fourth rule, 'check distribution\_payment', is marked with a green checkmark. The fifth rule, 'check payment\_paymentby', is marked with a red 'X'. The third row contains two 'rule' cards. The first rule, 'check payment\_paymentby', is marked with a red 'X'. The second rule, 'check distribution payment 2', is marked with a green checkmark. The bottom row contains one 'action' card, 'set output\_column', which is marked with a red 'X'. Each card displays the rule name, a status icon (green checkmark or red 'X'), and a detailed description of the rule or action. The 'rule' cards also show the specific test results, such as 'Field Financial Group [A] correctly is not equal to [XYZ]' or 'Field Insurance Company [Company 1] correctly is not equal to [ABC]'. The 'action' card shows the action description: 'set output\_column' and 'update field [output\_column] from: set value (number) field [distribution\_payment]'. The 'execute action when: passed' status is also visible.

project  
calculation\_sample  
calculation\_sample

rulegroup  
rulegroup 1  
rulegroup 1  
valid: 2016-02-26/2020-12-31

subgroup  
subgroup\_1  
subgroup 1  
rules connector: and

rule  
check financial group  
Field Financial Group [A] correctly is not equal to [XYZ]

rule  
check insurance company  
Field Insurance Company [Company 1] correctly is not equal to [ABC]

rule  
check invoicetotal patientamount  
Field Invoice reduced [0] correctly is equal to [0]

rule  
check distribution\_payment  
Field Distribution Payment [100] correctly is greater than [0]

rule  
check payment\_paymentby  
Field Payment by [50] incorrectly not is equal to [1]

rule  
check distribution payment 2  
Field Distribution Payment [100] correctly is not null

action  
set output\_column  
set output\_column  
update field [output\_column] from: set value (number) field [distribution\_payment]  
execute action when: passed

This testing feature allows the user to adjust the logic of the rulegroup, in case the test did not give the expected result. It also increases the overall quality of the rules and rulegroups if sufficient testing is done.

## Pentaho ETL tool integration

The Pentaho ETL tool named PDI (Pentaho Data Integration) is a tool to design ETL jobs and transformations.

In general the Business Rules Maintenance Tool produces/exports a project and all relevant details to a single zip file. This file can then be used with the JaRE ruleengine to run the rules and actions against a set of data.

Note that it is **not** required to use any Pentaho software to run the rules – the ruleengine can run standalone from the command line or can be integrated into any Java application.

There are two plugins available in PDI that connect the ETL to the external rules and actions (in the project file) and make the interactions of PDI with the ruleengine very easy. Having the rules and actions outside of the ETL code will separate the code from business rules and allows for cleaner design and code and - very important – and a separation of responsibilities between business and IT.

The ruleengine plugins for Pentaho PDI are available through the Pentaho marketplace. They can easily be installed from within the PDI application.

The plugins in PDI are what I call “puzzle pieces”. A certain number of these puzzle pieces form a complete picture (ETL) and the ruleengine plugin can be one of the pieces. The ruleengine plugin simply references the project zip file created with the web application and in turn executes the rules using the JaRe – Java Rule Engine.

The output is twofold: One incoming row will be evaluated and it will be passed to the next plugin or step as one row. It will contain additional fields from the ruleengine indicating how many rule groups and rules failed.

Optionally there is an additional second output stream, that shows the details of each row, compared to each rule that ran. So the number of rows in this optional stream is the product of the number of rows and the total number of rules defined in the project.

Within PDI the user can then decide how to continue in the workflow based on the results of the ruleengine. The number of failed/passed rulegroups can be used as an indicator. Or the number of failed or passed rules.



## **Workflow**

Below find some details and guidelines for a standard workflow that can be used.

### **Preparations**

Any user that is able to login to the application can create projects.

It is advisable that you think about following a naming convention for projects, rulegroups, subgroups, rules and actions. This will ensure that you or others are able to understand the purpose more easily. Also, when running the project against the ruleengine, some of the results will reference the names or descriptions you have assigned to the individual elements.

### **Create a project**

In general the name should be a rather short text string.- a sort of identifier for the element. The description may be longer.

Assign the project to a group. Users who are members in this group will be able to view and update the project content.

A name for the project export file may be defined. If undefined the name of the the project will be used when the project is exported.

If you do not want that users outside this group see this project, then mark the project as private.

Save your settings and click on the close button to go back to the previous page.

### **Create reference Fields**

Creating reference fields is not mandatory. If no reference fields are available, then all field names have to be entered manually and the appropriate field types (string, integer, date, etc) have to be selected from a list of possible types.

It is good practice anyway to define reference fields, as they help the end user to create rules and actions in a simpler and quicker way.

Reference fields also document the available fields and as such the interface of the target system. They also support the testing of the rulegroup and rules.

## Create a rulegroup

Back on the project list page, click on the name of the project. On the following page add a new rulegroup by clicking the relevant icon in the top part.

The rulegroup will contain subgroups, rules and actions. These form a logical unit for a specific condition or multiple ones you want to test.

Again, select a rather short name – according to your naming convention – and enter a more detailed description. Enter a valid from and a valid until date for the rulegroup. You may select a date in the past for the valid from date. If you think your group should be valid forever, simply add or select a date far in the future for the valid until date. Such as 2099-12-31.

Save your settings and click on the close button to go back to the previous page.

When you create a rulegroup, one subgroup inside the rulegroup is created automatically. It is named „subgroup\_1“. You can edit this subgroups details if necessary. Take special attention to the rules connector displayed in the lower left hand corner of the subgroup. It defines how the rules in this subgroup are connected to each other. The values can be „and“ or „or“.

Later in the process you can create more subgroups as required.

## Create a rule

Inside the subgroup displayed click to add a new rule.

Enter a name and description for the rule. Next, enter the name of the field that shall be checked (tested) and next to it the data type of the data. This references a field of the data source you will later run the rules against. It could be a CSV file or a database or some other data source. The field name should be entered exactly as it is in the data source; uppercase/lowercase may be relevant.

Next, select a check you want to use with the field defined before. E.g. a check for equality. This is a good moment to look at the documentation for the checks to find out if the desired check is available and if the logic you want to use is implemented. Mainly you want to make sure that the data types involved are usable for a certain check.

Next there are two possibilities: Either check the field against another field of the data source. Or check the field against a given fixed value. If you check against another field, enter the field name and select it's type. If not, leave this definition empty and enter in the next line the value you want to check against and it's type.

Some of the checks allow you to specify additional parameters. For example the check for equality of strings takes an optional parameter, if the comparison should be case sensitive. If you need to specify an additional parameter, enter it and its type.

Next comes the message that is generated in case the data passes the check and the message in case it fails. You may enter a free text to describe the two messages. Next to the text field to enter the message, there is a button. It auto-generates a message for you.

In the auto-generated message there will be placeholders: \$1 and \$0 (at times only one of them). The placeholder \$1 represents the actual value of the data. The placeholder \$0 represents the value to compare against – either the data from another field or the given fixed value.

At this point the rule is complete. Save it and click „close“ to go back to the rulegroup listing.

## More rules

If you want to implement more rules, repeat the last step. The displayed rules connector of the subgroup will indicate if the rules in a certain subgroup are connected by an „and“ or „or“ condition. You can change this by editing the subgroup.

## More subgroups

Subgroups can be connected to each other by an „and“ or „or“ condition. This condition always defines the connection to the **previous** subgroup. If you have two subgroups and „Connector to previous group“ (edit the subgroup to change) is set to „or“, then the logic is such that the subgroup 1 has to pass the checks or the subgroup 2 has to pass the checks. The other way around, if the subgroups are connected with an „and“ condition, then both subgroups have to pass the checks for the rulegroup to pass.

So all rules in the individual subgroups and the subgroups itself together form a logic. This collection of logic is captured inside a rulegroup. The ruleengine will execute the rules in the rulegroup and the rulegroup will pass or fail the tests.

## Create an action

Actions belong to the rulegroup they are defined in. Based on the fact if the rulegroup passes or fails or in both cases, one or multiple actions are executed.

Enter a name and description for the action. Next, select if the action should be executed when the rulegroup fails or when it passes or in both cases. Next, select a field and its type

that shall be updated. The next step is to select which action to execute – select one from the drop down box. Make sure you select the correct action for the data types involved.

Again, a good moment to consult the documentation for the actions to determine if the desired action is available, for which data types it is applicable and what (optional) additional parameters are available.

As with the rules, there are two possibilities. An action can use the data of a field (or two fields) or a given fixed value that will be used to execute the action. If the data comes from a field (or two), enter the field name and select it's type.

If you are using a fixed value, then leave the field to retrieve data from and it's type empty and enter the fixed value in the first text box for the parameters. And select the type of the parameter's data.

Specify any additional parameters in the following text boxes and select the data type for each parameter.

Finally save the action and click on „close“ to go back to the rulegroup listing.

## More actions

Continue to create more actions as required. Actions will be executed one after the other as defined in the Business Rule Maintenance web application.

## Testing a Rulegroup

Once the rules have been designed, the user can test a rulegroup to determine if the rules have been correctly put together and the logic is working as planned.

First, the user has to input data for the fields that are involved in the rules and actions. The rules are then executed against this data and the result is presented to the user. The user will get a visual indication which rules passed or failed and the relevant message that was produced for each rule. It is also indicated to the user if the rulegroup passed or failed as a whole.

On the page for a project where the subgroups, rules and actions are displayed, click on the relevant test icon for the rule group. You will be required to enter test data for all the fields that are displayed. These are all fields that are used in one of the rules of the rule group. Fields that are not used in one of the rules or actions of the rulegroup will not be available for testing. Make sure, that you enter the correct data corresponding to the types of the

fields. That is very important to receive correct test results.

Values for fields of type “Date” may be selected from a calendar but may also be typed in manually by hand. Values for fields of type “boolean” (yes/no or true/false) can be selected from a dropdown box.

Once you run the test the process will return to the page where the subgroups, rules and actions are displayed and display the results of the test run. You can then decide to run the test with other data or to stop the test.

## Export project

Once you have defined all rules and actions that are required to check a certain business logic, export the project. This will create a zip file that contains all your definitions in a single file. This file can directly be executed with the ruleengine.

In the menu click the „Export Project“ entry. Select the relevant project from the drop down list. Next, select a validity date. As described further above, rulegroups (which contain all rules and actions) have a defined validity in terms of time. These are valid from a given date and until a given end date. When you select the validity date for the project to be exported, only those rulegroups will be exported, which are valid at the given validity date. The other way around: rulegroups, that are not valid at the given date (expired or not yet active), will not be included in the project zip file.

This feature allows you to plan changes in your business logic ahead of time. You can implement new or changed logic well before it will become active.

## Executing rules and actions

The final step of the workflow is then to execute the defined rules and actions.

One way is to run them in the Pentaho ETL tool where you will get the results of the execution directly inside the ETL tool.

You may also include the ruleengine in your own software or web application and run it from there. The API (Application Programming Interface) for the ruleengine will list the possible methods to retrieve the results of the execution and the messages that were generated.

Another possibility is to run the ruleengine on a separate server. In this case the server waits for a (socket) connection from a client to execute the rules and actions. When executed in this way, all logging of the ruleengine and the results of the individual rules and

actions will be done on the server. Only a minimum of data is sent back to the client, indicating how many rulegroups and rules have passed or failed.

## **Software Updates**

The Business Rules Maintenance web application uses several software libraries. If a newer version of these libraries is available it might be desired to replace the existing ones. In this is possible depends on the compatibility of the newer libraries with this web application code.

The libraries are located in following folder:

*[Tomcat root folder]/webapps/rule\_maintenance/WEB-INF/lib*

Updates to the base components Java, Apache Tomcat and MySQL/MariaDB should in general be possible without interference to the web application. Otherwise check if a newer version of the web application is available.

Attention: Before you upgrade all or parts of the applications always make sure that you have a backup of the web application and – more importantly – of your data, in case something goes wrong.

## **Extending the Ruleengine**

The ruleengine can be extended for the two main concepts: checks and actions. By implementing a defined interface the developer can create new checks or actions. The resulting Java classes can then be added to the ruleengine jar file or to the Java classpath, so they can be located by Java.

It is also possible to add new checks and actions to the GitHub repository by issuing a pull request, so they will be reviewed and then included in a future release of the application.

This way other users and the community as such will benefit from it.

## **Extending the Web Application**

The web application mainly – except from the server side Java code – consists of Beanshell scripts and Apache Velocity templates. The scripts are used to handle the application logic and interface with the database. The templates are used for the visual representation in the browser to the user.

As an administrator you can modify the existing code in the scripts and templates to fix problems or to change or enhance the existing functionality.

You may also add new script or template files. If no server side code needs to be changed, then this can be done through the web application. This feature allows to created new pages with new functionality without the need of a direct programming and without the need to restart the web application.



## **Software components**

The web application uses following software components:

- Java code on the server side
- Beanshell scripts on the server side
- Apache Velocity templates on the server side
- JQuery and Javascript on the client side
- CSS (Cascading Style Sheets) on the client side

All current templates for the web application, as well as the the menu on the left side , are based on the English language.

The application is technically prepared to handle other languages as well. This can be achieved by translating the templates and menu to a different language of your choice.

## **Contact**

For questions or feedback please contact me.

- Email: [uwe.geercken@web.de](mailto:uwe.geercken@web.de)
- Twitter: @uweeegeee

## Links

Below is a list of links to software or web pages references in this document.

- MySQL/MariaDB database schema:

Link: [https://github.com/uwegeercken/rule\\_maintenance\\_db](https://github.com/uwegeercken/rule_maintenance_db)

Download file: ruleengine\_rules.sql

- Business Rules Maintenance Tool .war file

Link: [https://github.com/uwegeercken/rule\\_maintenance\\_war](https://github.com/uwegeercken/rule_maintenance_war)

Download file: rule\_maintenance.war

- Business Rules Maintenance Tool documentation

Link: [https://github.com/uwegeercken/rule\\_maintenance\\_documentation](https://github.com/uwegeercken/rule_maintenance_documentation)

Download file: various

- JaRE (Java Rule Engine) source code and JaRE java library

Link: <https://github.com/uwegeercken/jare>

Download file: various

- Apache Velocity Template engine

Link: <http://velocity.apache.org/>

- MySQL Database Server and tools

Link: <http://www.mysql.com/>

- MariaDB Database Server and tools

Link: <http://mariadb.org/>

- Oracle Java

Link: <https://www.java.com>

## Defaults

Listed below are defaults that the web application uses.

- Business Rules Maintenance application default login

userid: *admin*

password: *admin*

*Note: You should change the default password for security reasons. This can be done from inside the web application.*

- Database name: *ruleengine\_rules*

*Note: The user creating the database can choose to name the database differently. The configuration page in the web application allows to specify this name.*

- When handling dates in the application, the application always expects date values to be specified in the format: four digits for the year, then two digits for the month and finally two digits for the day of the month. The individual values must be separated by a minus sign. Format:

*yyyy-MM-dd*

Example: *2016-12-16*

## Subject Index

Actions.....	4, 8p., 13p., 18pp., 31
Activity.....	8
Apache Tomcat.....	5
Apache Velocity.....	5p.
Beanshell.....	5p.
Checks.....	9, 15p., 31
Configuration.....	4pp., 11, 14, 17
Deactivated user.....	13
Dependency.....	20
Dependent.....	20
Export.....	4, 9, 17p., 24
Github.....	35
GitHub.....	31
Groups.....	4, 8p., 11, 13p., 18pp.
History.....	8, 13, 18
Import.....	6, 9, 17p.
JaRE.....	4p., 18, 24
Java.....	4p., 24, 30, 35
Language.....	33
LDAP.....	4p., 8, 11, 14, 17
MariaDB.....	5p., 30, 35
MySQL.....	5p., 35
Password.....	6pp., 14, 36
Pentaho ETL tool integration.....	24
Pentaho PDI.....	5, 24
Projects.....	4, 8, 11, 14, 18p.
Reference fields.....	14p.
Reference Fields.....	14
Rulegroups.....	4, 8p., 13p., 18pp.
Rules.....	4pp., 8p., 13pp., 18pp., 24, 30, 35p.
Search.....	9
Subgroups.....	8p., 13p., 18pp.
Testing.....	15, 21, 28
Tomcat.....	5p.
Validity date.....	18, 29
Workflow.....	25
Zip file.....	4, 9, 17p., 24