

# Drools Tools Reference Guide



Version: 5.1.0.trunk

---

<b>1. Introduction</b>	1
1.1. What is Drools?	1
1.2. Drools Tools Key Features	1
1.3. Other relevant resources on the topic	2
<b>2. Creating a New Drools Project</b>	3
2.1. Creating a Sample Drools Project	3
2.2. Drools Project Structure Overview	8
2.3. Creating a New Rule	9
<b>3. Debugging rules</b>	13
3.1. Creating Breakpoints	13
3.2. Debugging	14
<b>4. Editors</b>	17
4.1. The Rule Editor	17
4.1.1. Content Assist	17
4.1.2. Code Folding	19
4.1.3. Synchronization with Outline View	20
4.1.4. The Rete Tree View	21

# Introduction

## 1.1. What is Drools?

**Drools** is a business rule management system (BRMS) with a forward chaining inference based rules engine, more correctly known as a production rule system, using an enhanced implementation of the Rete algorithm.

In this guide we are going to get you familiar with Drools Eclipse plugin which provides development tools for creating, executing and debugging Drools processes and rules from within Eclipse.



### Note:

It is assumed that you has some familiarity with rule engines and Drools in particular. If no, we suggest that you look carefully through the [Drools Documentation](http://downloads.jboss.com/drools/docs/4.0.7.19894.GA/html_single/index.html) [http://downloads.jboss.com/drools/docs/4.0.7.19894.GA/html\_single/index.html].

**Drools Tools** come bundled with **JBoss Tools** set of Eclipse plugins. How to install **JBoss Tools** you can find in the [Getting Started Guide](#) [../GettingStartedGuide/html\_single/index.html#JBossToolsInstall].

## 1.2. Drools Tools Key Features

The following table lists all valuable features of the **Drools Tools**.

**Table 1.1. Key Functionality of Drools Tools**

Feature	Benefit	Chapter
Wizard for creating a new Drools Project	The wizard allows to create a sample project to easy get started with Drools	<a href="#">Creating a Sample Drools Project</a>
Wizards for creation new Drools resources	A set of wizards are provided with the Drools Eclipse tools to quickly create a new Rule resource, a new Domain Specific language, Decision Table and Business rule	<a href="#">Creating a New Rule</a>
The Rule editor	An editor that is aware of DRL syntax and provides content assistance and synchronizing with the Outline view	<a href="#">The Rule Editor</a>
The Domain	The editor allows to create and manage mappings from users language to the rule language	

Feature	Benefit	Chapter
Specific Language editor		
The Guided editor	This guided editor for rules allows you to build rules in a GUI driven fashion based on your object model	
The Rule Flow graphical editor	The editor is meant for editing visual graphs which represent a process (a rule flow)	

### 1.3. Other relevant resources on the topic

- Drools on [JBoss.org](http://www.jboss.org/drools/) [http://www.jboss.org/drools/]
- [JBoss Tools Home Page](http://www.jboss.org/tools/) [http://www.jboss.org/tools/]
- [The latest JBossTools/JBDS documentation builds](http://download.jboss.org/jbosstools/nightly-docs/) [http://download.jboss.org/jbosstools/nightly-docs/]
- All [JBoss Tools/JBDS](http://docs.jboss.org/tools/) documentation you can find on the [documentation release page](http://docs.jboss.org/tools/) [http://docs.jboss.org/tools/].

# Creating a New Drools Project

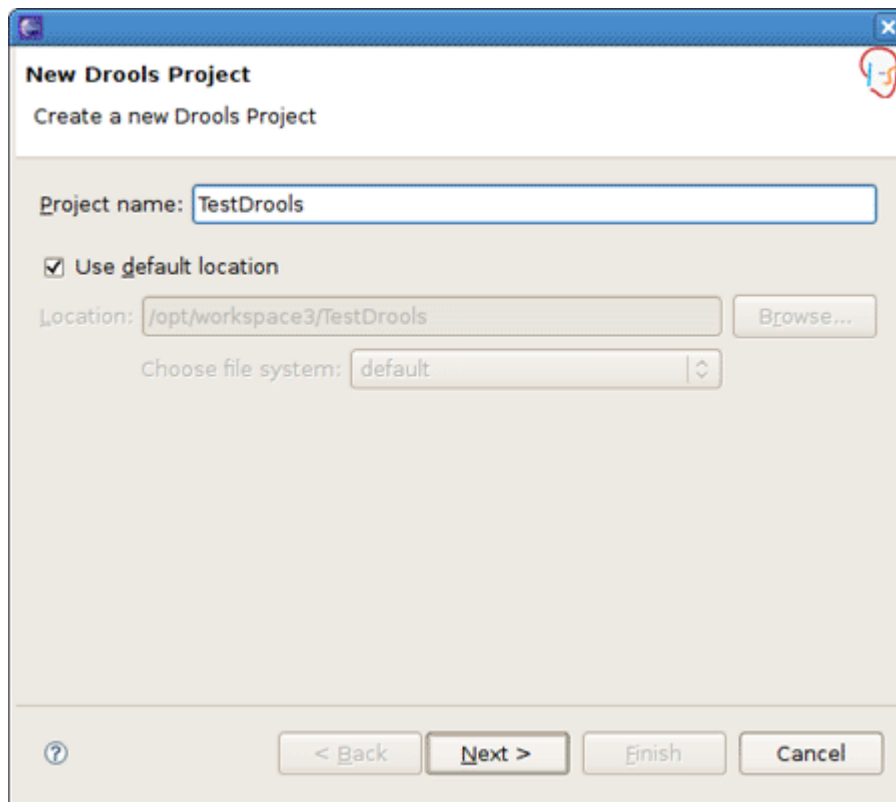
In this chapter we are going to show you how to setup an executable sample Drools project to start using rules immediately.

## 2.1. Creating a Sample Drools Project

First, we suggest that you use [Drools perspective](#) which is aimed at work with Drools specific resources.

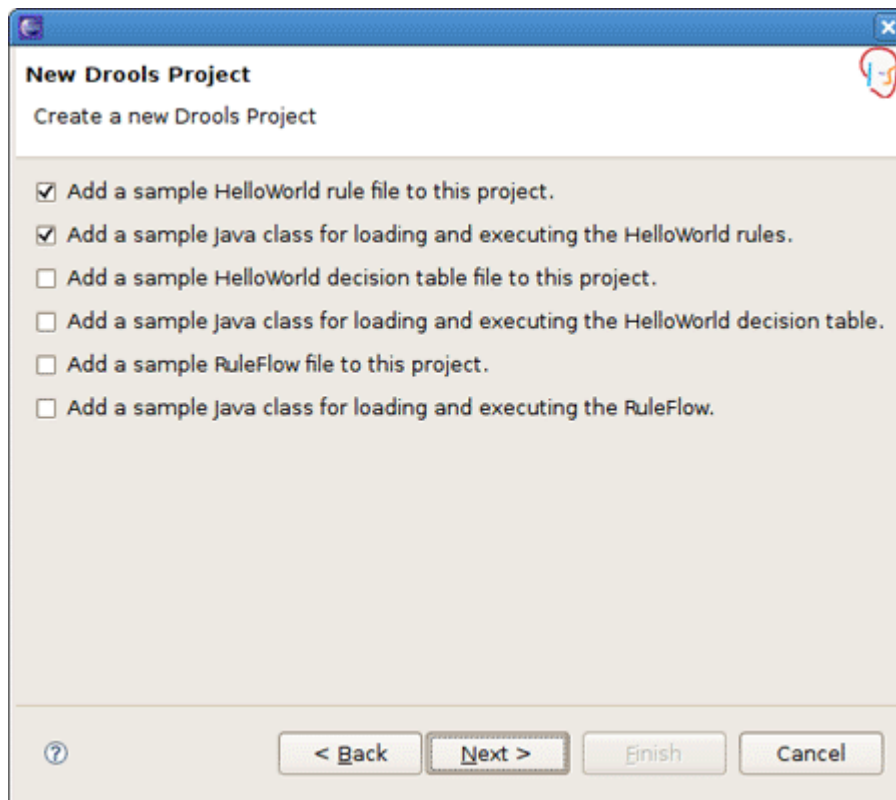
To create a new Drools project follow to [File > New > Drools Project](#). This will open [New Drools Project wizard](#) like on the figure below.

On the first page type the project name and click [Next](#).



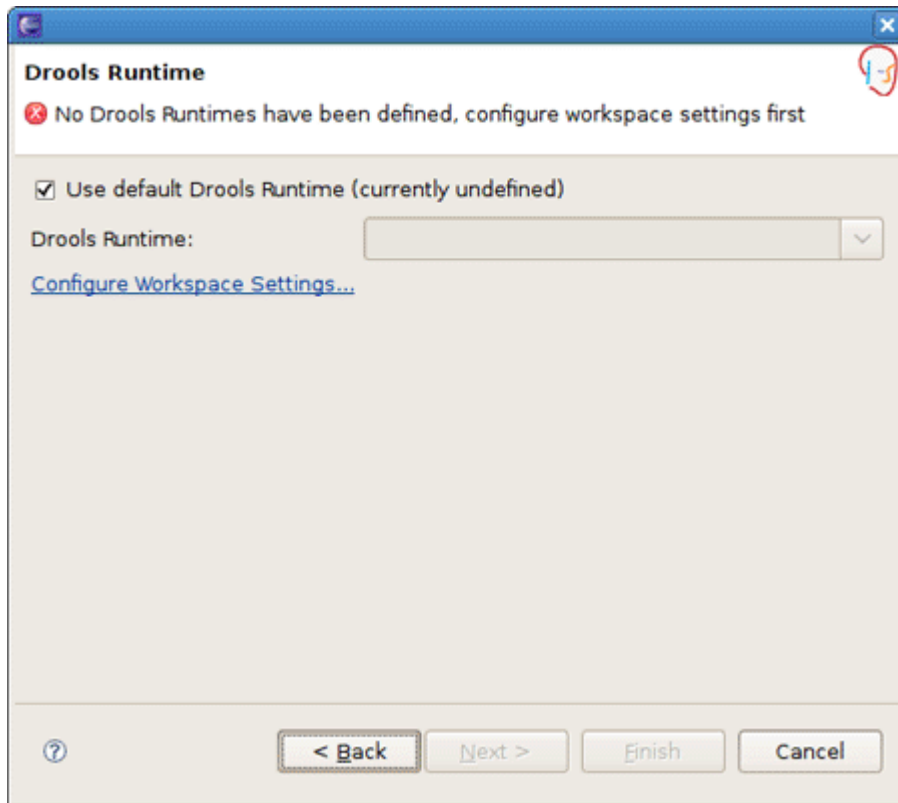
**Figure 2.1. Creating a New Drools Project**

Next you have a choice to add some default artifacts to it like sample rules, decision tables or ruleflows and Java classes for them. Let's select first two check boxes and press [Next](#).



**Figure 2.2. Selecting Drools Project Elements**

Next page asks you to specify a Drools runtime. If you have not yet set it up, you should do this now by clicking the [Configure Workspace Settings](#) link.



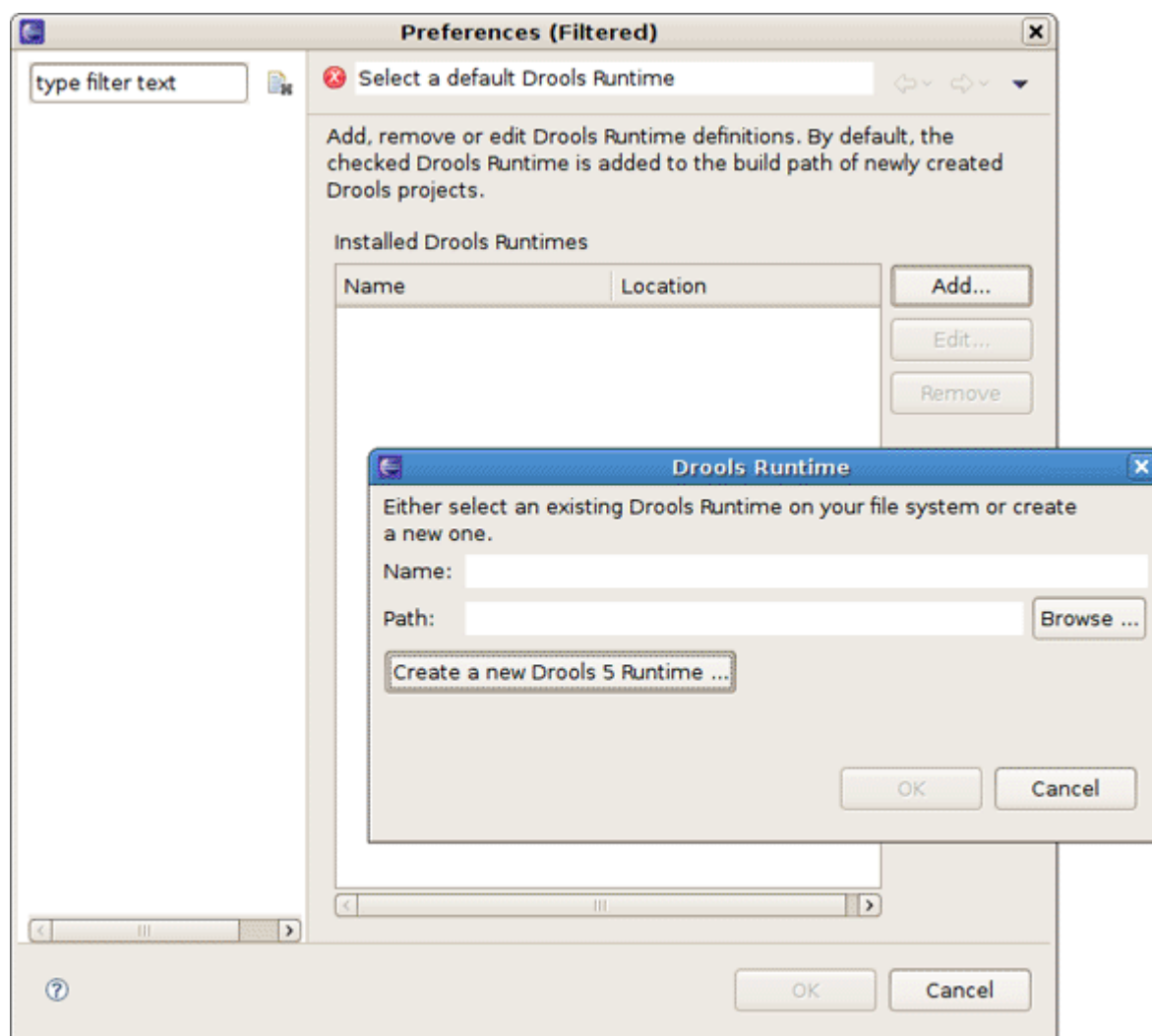
**Figure 2.3. Configuring Drools Runtime**

You should see the [Preferences window](#) where you can configure the workspace settings for Drools runtimes. To create a new runtime, press the [Add](#) button. The appeared dialog prompts you to enter a name for a new runtime and a path to the Drools runtime on your file system.



**Note:**

A Drools runtime is a collection of jars on your file system that represent one specific release of the Drools project jars. While creating a new runtime, you must either point to the release of your choice, or you can simply create a new runtime on your file system from the jars included in the Drools Eclipse plugin.

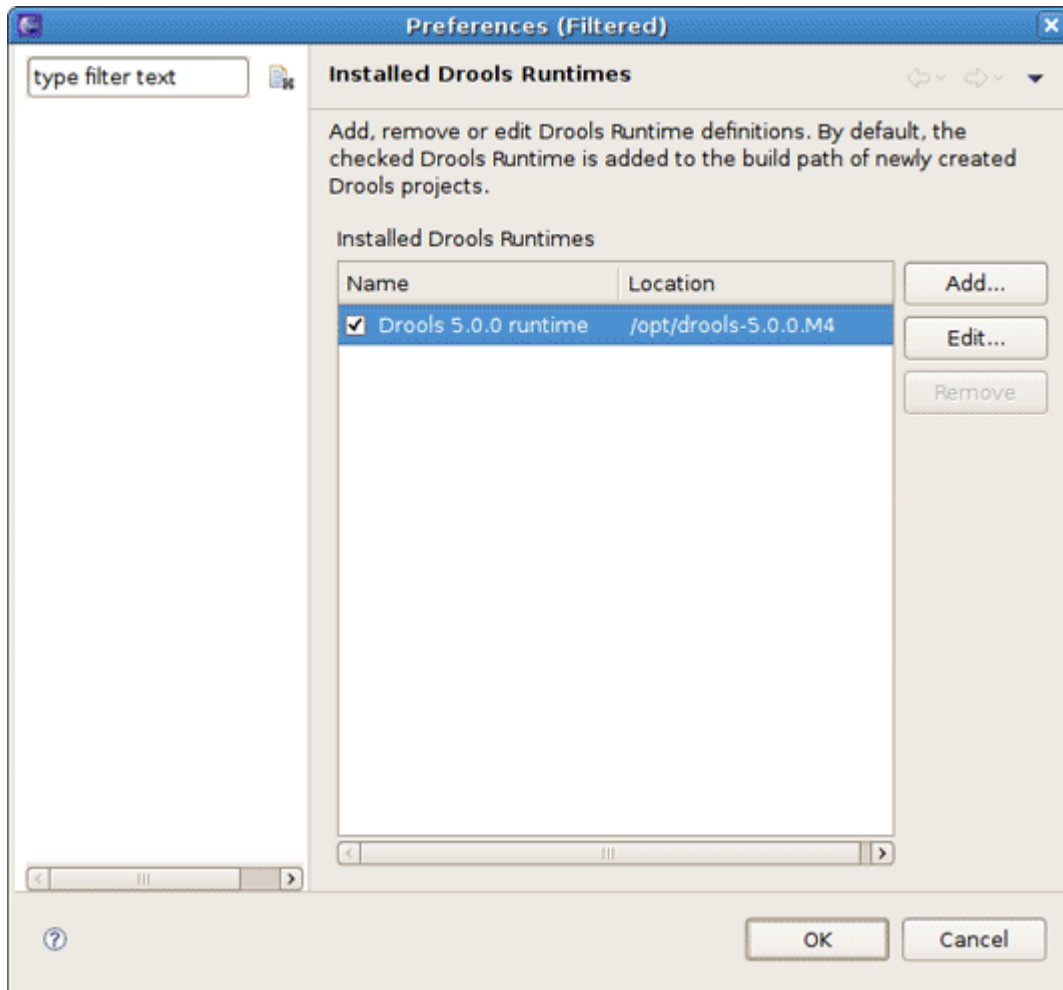


**Figure 2.4. Adding a New Drools Runtime**

Let's simply create a new Drools 5 runtime from the jars embedded in the Drools Eclipse plugin. Thus, you should press [Create a new Drools 5 runtime](#) button and select the folder where you want this runtime to be created and hit [OK](#).

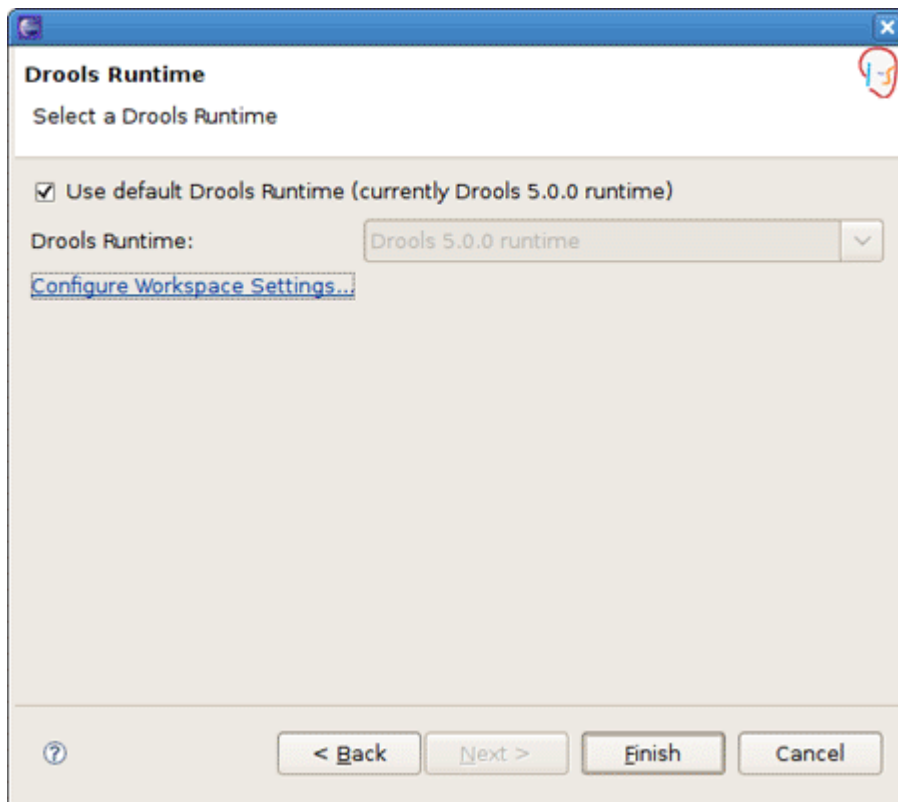
You will see the newly created runtime show up in your list of Drools runtimes. Check it and press [OK](#).





**Figure 2.5. Selecting a Drools Runtime**

Now press *Finish* to complete the project creation.

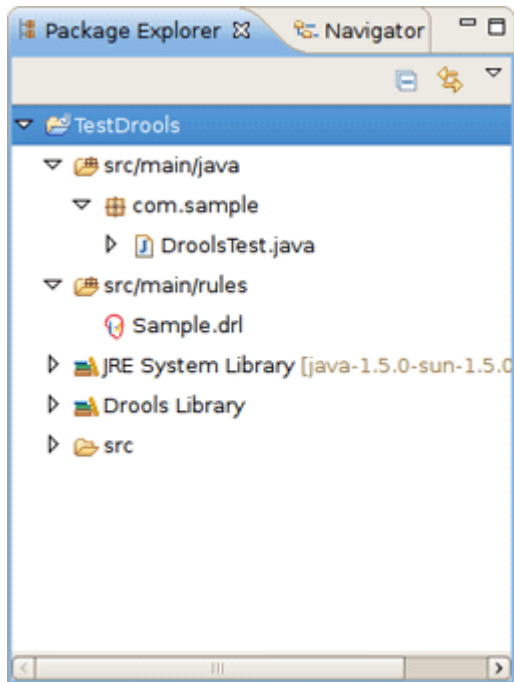


**Figure 2.6. Completing the Drools Project Creation**

This will setup a basic structure, classpath and sample rules and test case to get you started.

## 2.2. Drools Project Structure Overview

Now let's look at the structure of the organized project. In the [Package Explorer](#) you should see the following:



**Figure 2.7. Drools Project in the Package Explorer**

The newly created project contains an example rule file [Sample.drl](#) in the [src/main/rules](#) directory and an example java file [DroolsTest.java](#) that can be used to execute the rules in a Drools engine in the folder [src/main/java](#), in the [com.sample](#) package. All the others jar's that are necessary during execution are also added to the classpath in a custom classpath container called [Drools Library](#).



#### Tip:

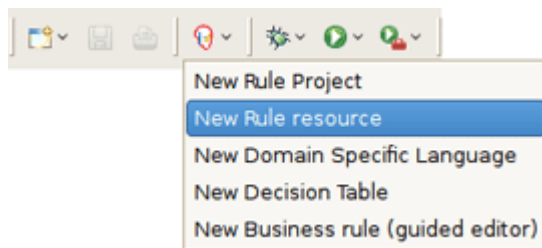
Rules do not have to be kept in Java projects at all, this is just a convenience for people who are already using eclipse as their Java IDE.

## 2.3. Creating a New Rule

Now we are going to add a new Rule resource to the project.

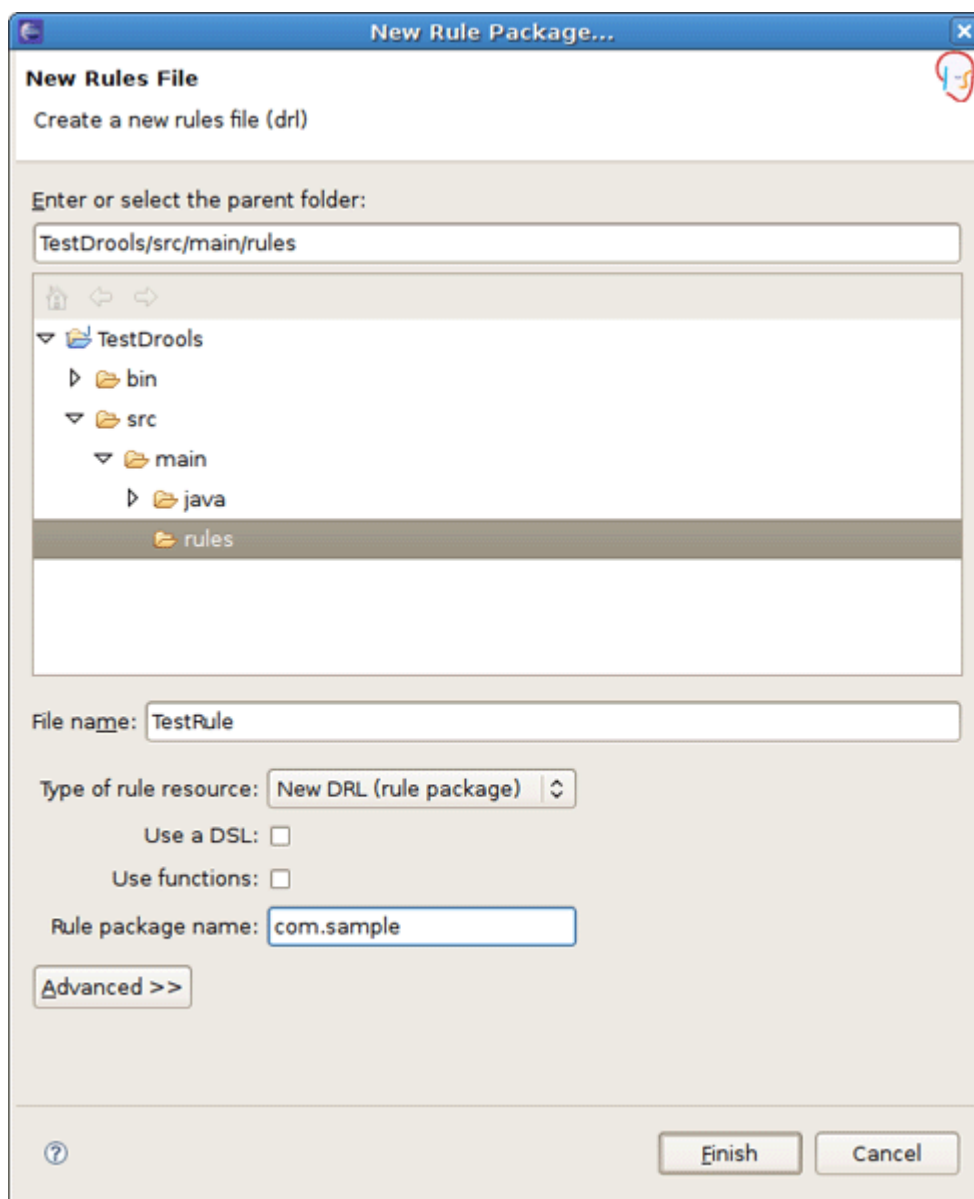
You can either create an empty text [.drl](#) file or make use of the special [New Rule Resource wizard](#) to do it.

To open the wizard follow to [File > New > Rule Resource](#) or use the menu with the JBoss Drools icon on the toolbar.



**Figure 2.8. Opening the New Rule Resource Wizard**

On the wizard page first select `/rules` as a top level directory to store your rules and type the rule name. Next it's mandatory to specify the rule package name. It defines a namespace that groups rules together.



**Figure 2.9. New Rule Resource Wizard**

As a result the wizard generates a rule skeleton to get you started.

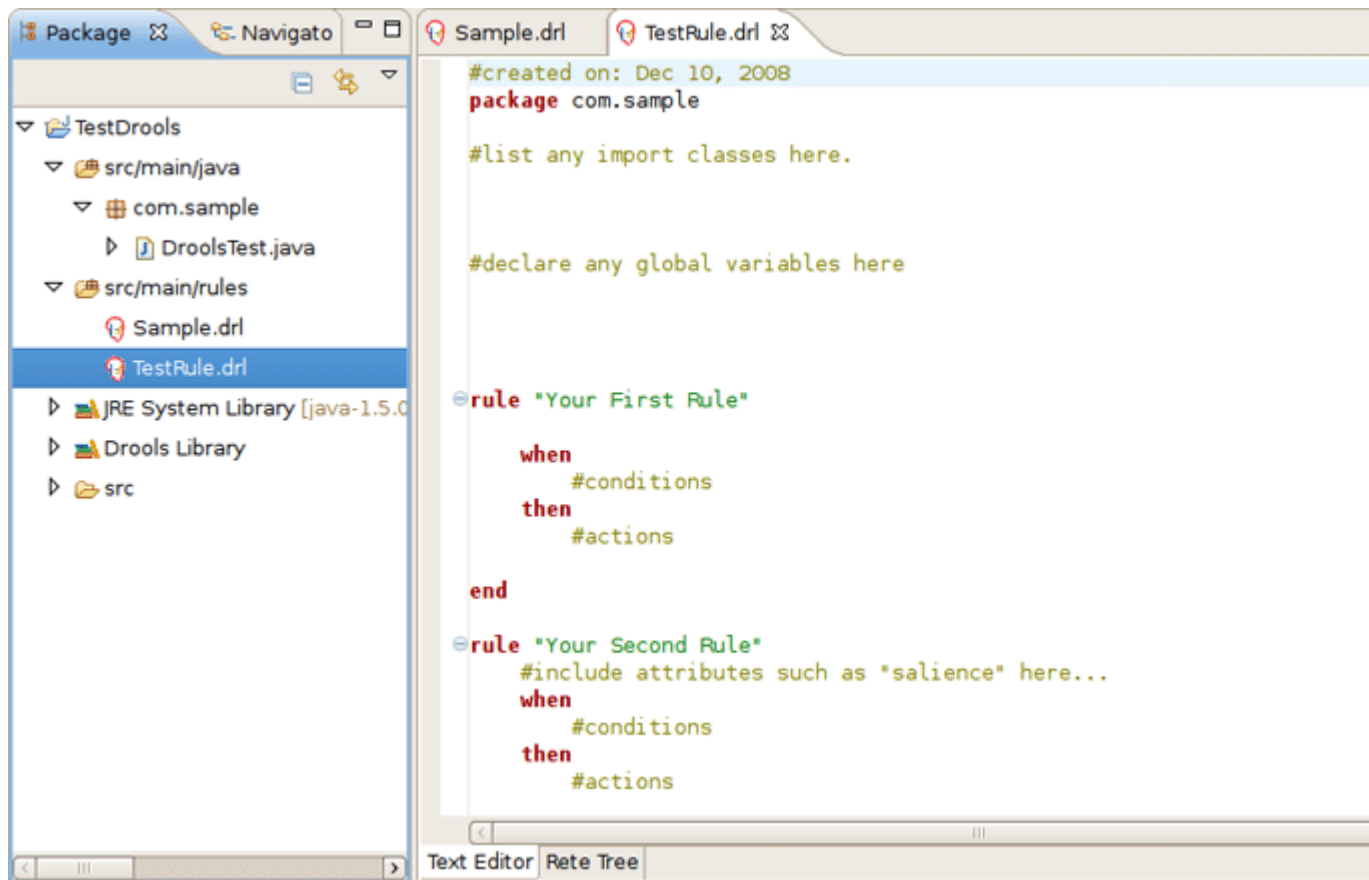


Figure 2.10. New Rule

# Debugging rules

This chapter describes how to debug rules during the execution of your Drools application.

## 3.1. Creating Breakpoints

At first, we'll focus on how to add breakpoints in the consequences of your rules.

Whenever such a breakpoint is uncanceled during the execution of the rules, the execution is halted. It's possible then inspect the variables known at that point and use any of the default debugging actions to decide what should happen next (step over, continue, etc). To inspect the content of the working memory and agenda the Debug views can be used.

You can add/remove rule breakpoints in `.drl` files in two ways, similar to adding breakpoints to Java files:

- Double-click the ruler in the [Rule editor](#) at the line where you want to add a breakpoint.

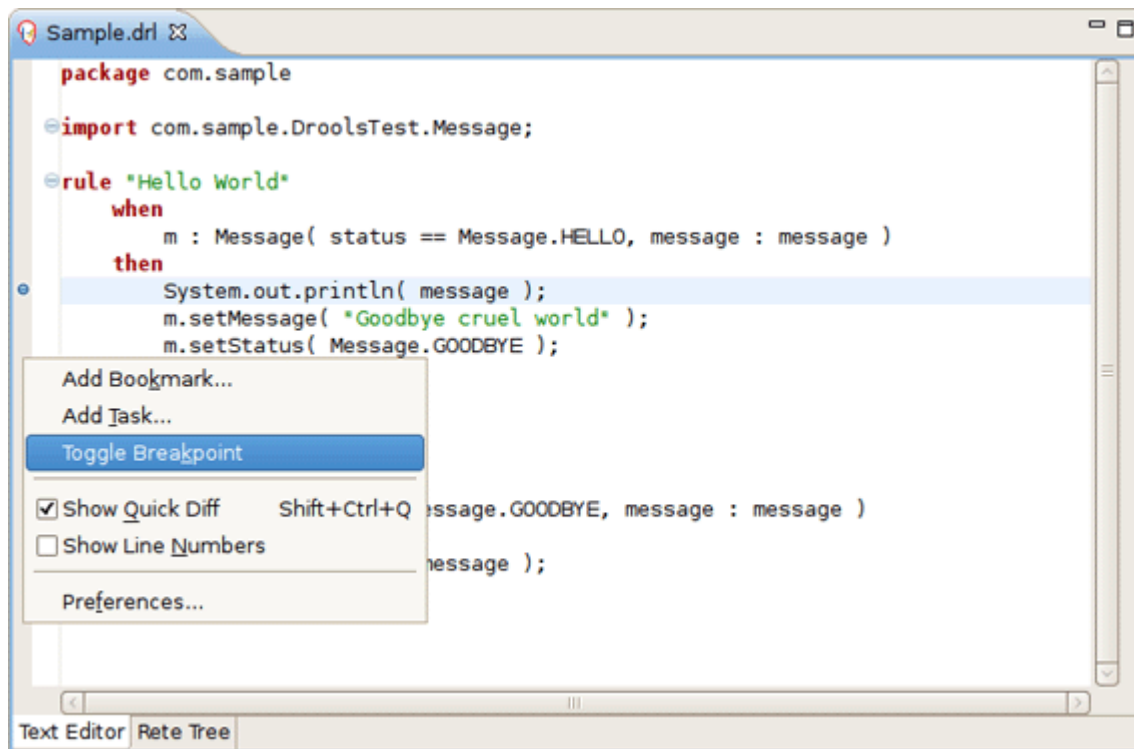


### Tip:

Note that rule breakpoints can only be created in the consequence of a rule. Double-clicking on a line where no breakpoint is allowed will do nothing.

A breakpoint can be removed by double-clicking the ruler once more.

- Right-click the ruler. Select [Toggle Breakpoint](#) action in the appeared popup menu. Clicking the action will add a breakpoint at the selected line or remove it if there is one already.



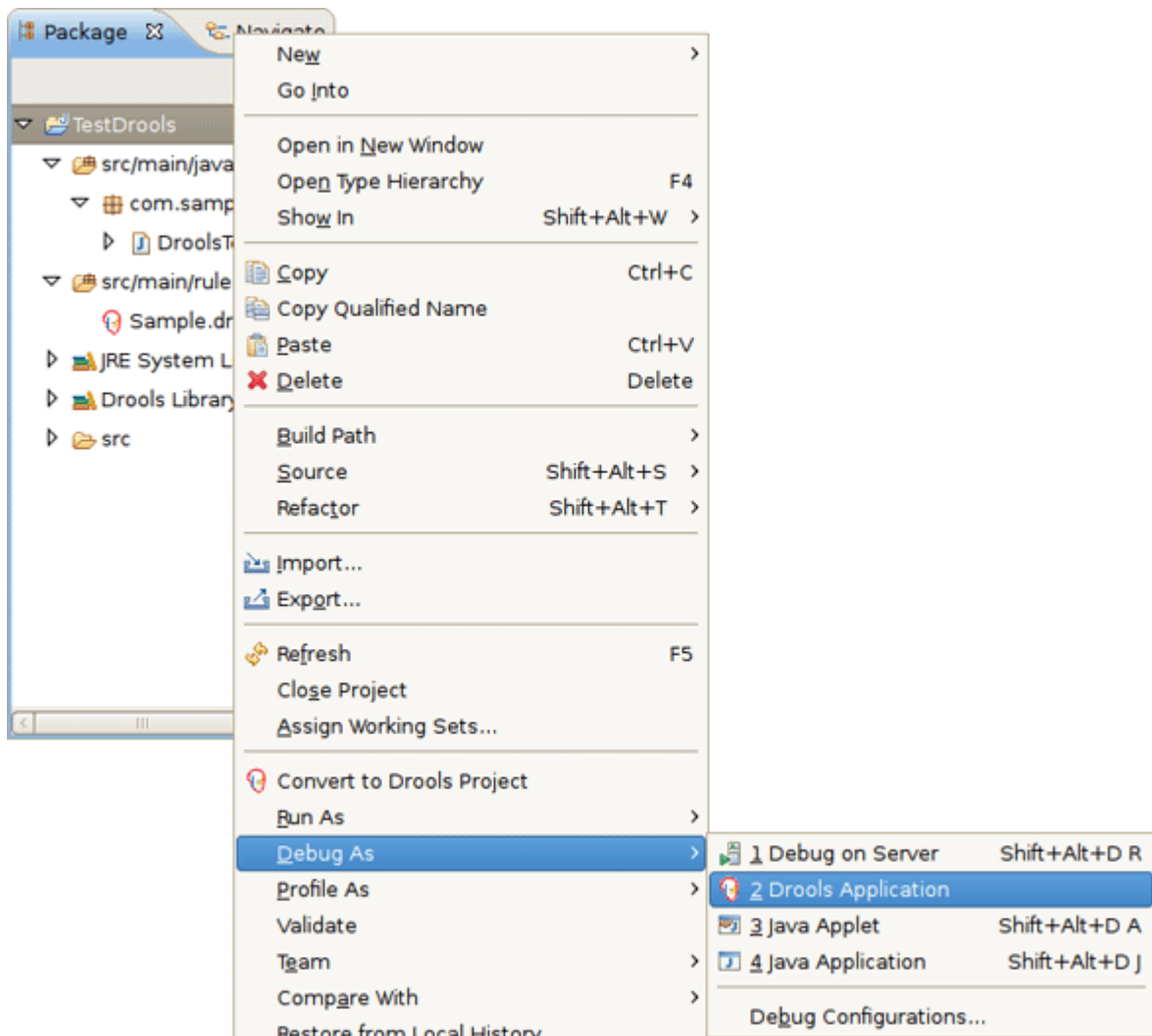
**Figure 3.1. Adding Breakpoints**

The [Debug perspective](#) contains a [Breakpoints view](#) which can be used to see all defined breakpoints, get their properties, enable/disable or remove them, etc. You can switch to it by navigating to [Window > Perspective > Others > Debug](#).

## 3.2. Debugging

Drools breakpoints are only enabled if you debug your application as a Drools Application. To do this you should perform one of the actions:

- Select the main class of your application. Right click it and select [Debug As > Drools Application](#).

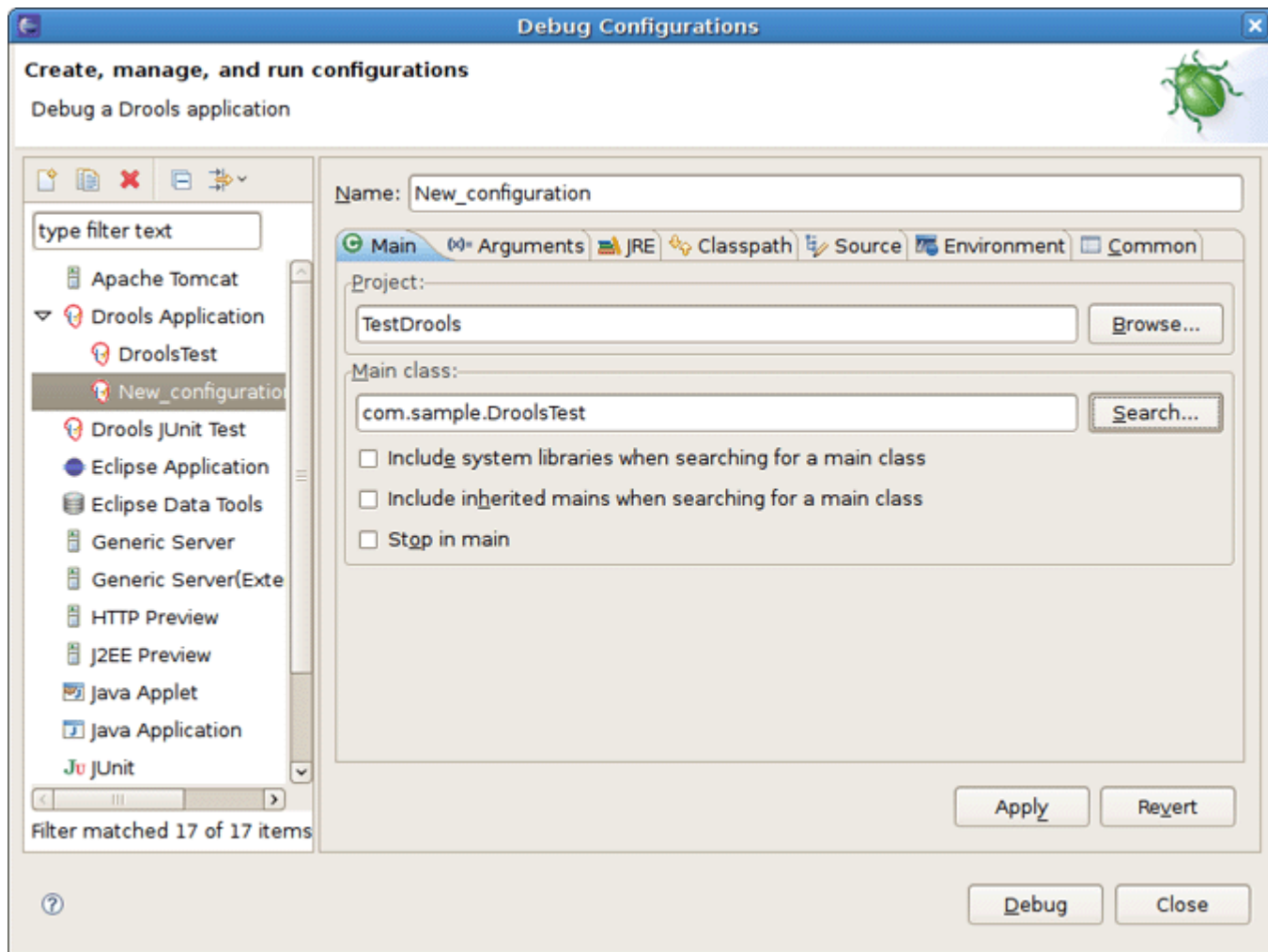


**Figure 3.2. Debugging Drools Application**

- Alternatively, you can also go to [Debug As > Debug Configuration](#) to open a new dialog for creating, managing and running debug configurations.

Select the [Drools Application](#) item in the left tree and click the [New launch configuration](#) button (leftmost icon in the toolbar above the tree). This will create a new configuration and already fill in some of the properties (like the Project and Main class) based on main class you selected in the beginning. All properties shown here are the same as any standard Java program.





**Figure 3.3. New Debug Configuration**



### Tip:

Remember to change the name of your debug configuration to something meaningful.

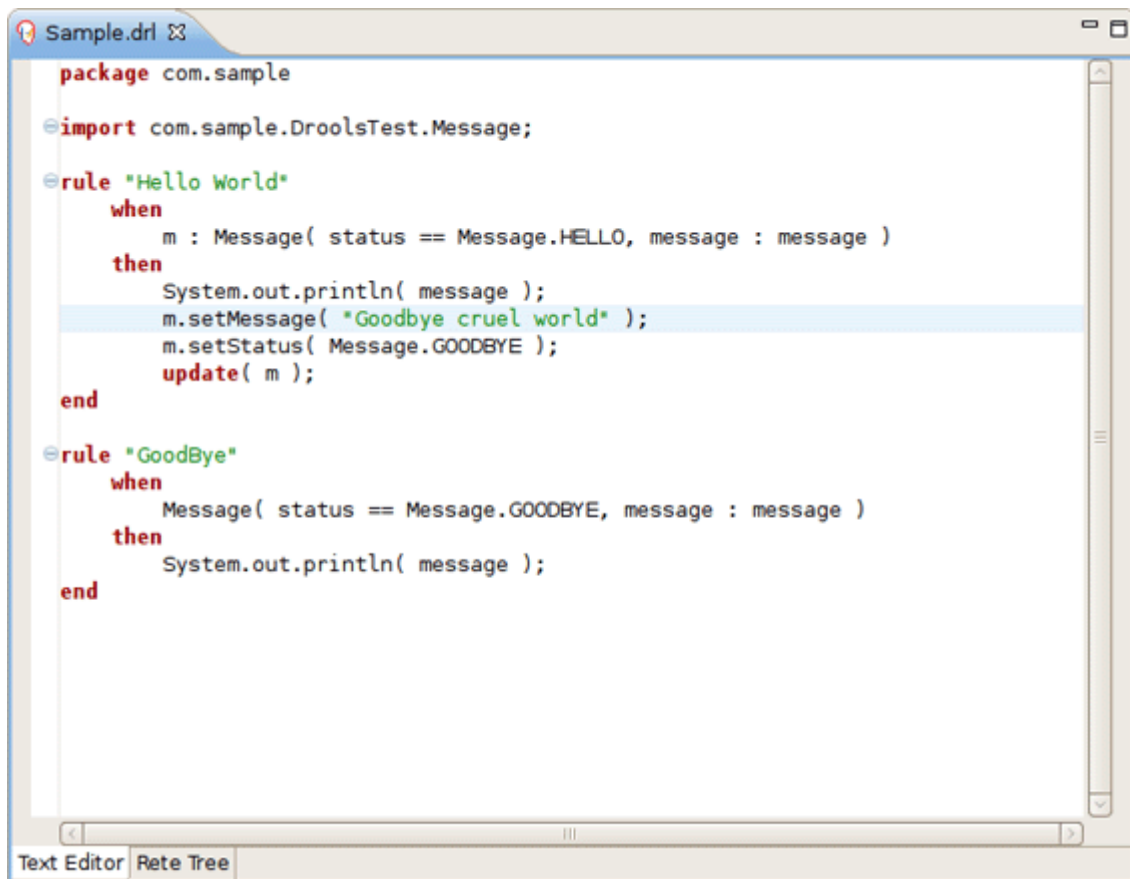
Next click the [Debug](#) button on the bottom to start debugging your application.

After enabling the debugging, the application starts executing and will halt if any breakpoint is encountered. This can be a Drools rule breakpoint, or any other standard Java breakpoint. Whenever a Drools rule breakpoint is encountered, the corresponding `.drl` file is opened and the active line is highlighted. The Variables view also contains all rule parameters and their value. You can then use the default Java debug actions to decide what to do next (resume, terminate, step over, etc.). The debug views can also be used to determine the contents of the working memory and agenda at that time as well (you don't have to select a working memory now, the current executing working memory is automatically shown).

# Editors

## 4.1. The Rule Editor

The [Rule editor](#) works on files that have a [.drl](#) (or [.rule](#) in the case of spreading rules across multiple rule files) extension.



**Figure 4.1. New Rule**

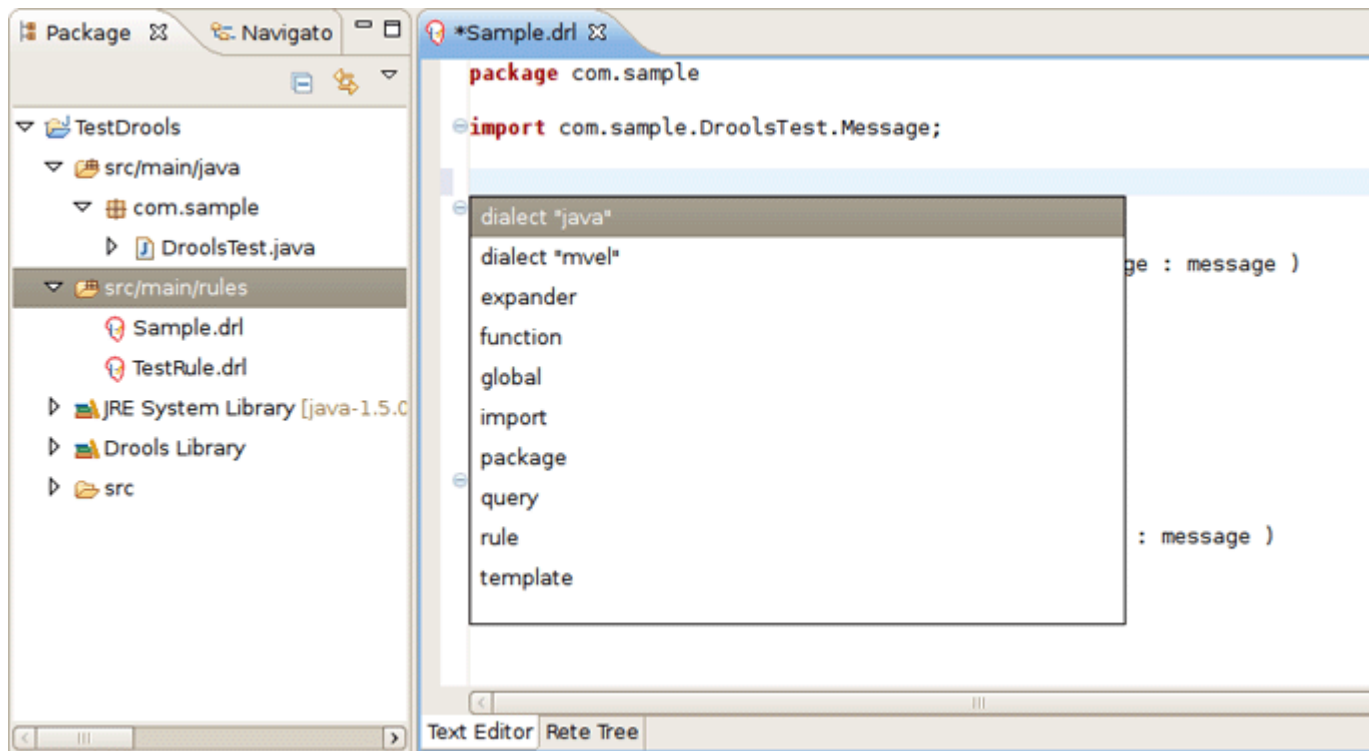
The editor follows the pattern of a normal text editor in eclipse, with all the normal features of a text editor:

- [Content Assist](#)
- [Code Folding](#)
- [Synchronization with Outline View](#)

### 4.1.1. Content Assist

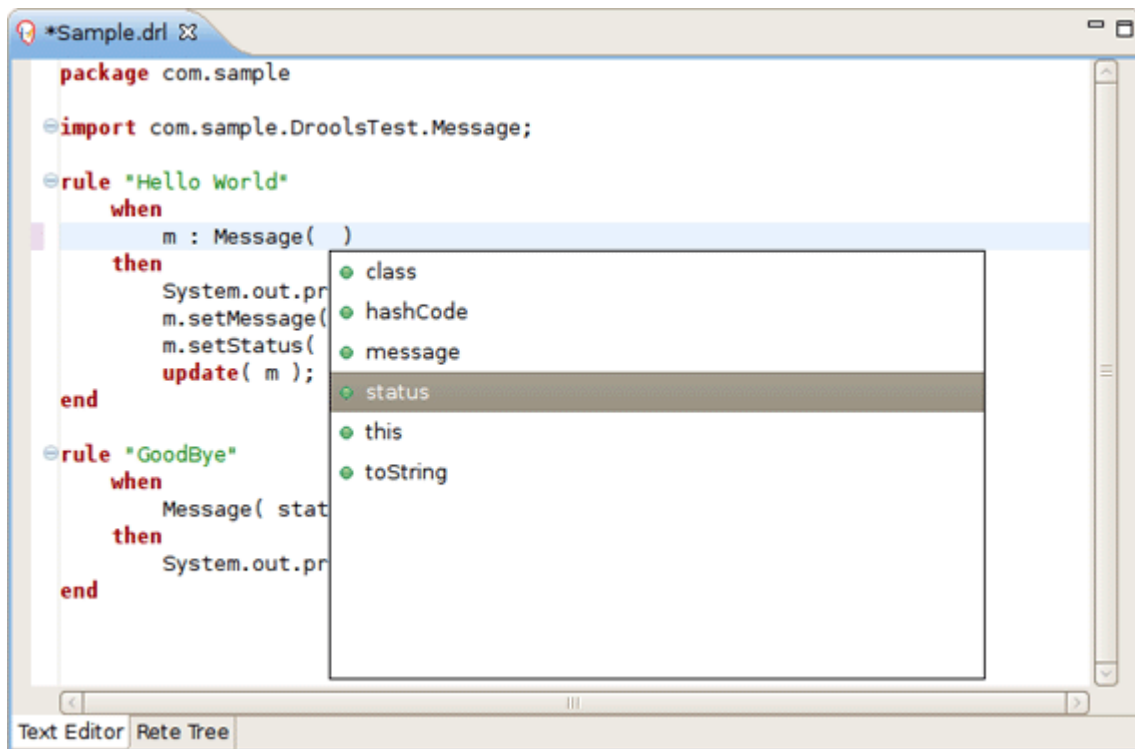
While working in the [Rule editor](#) you can get a content assistance the usual way by pressing [Ctrl + Space](#).

Content Assist shows all possible keywords for the current cursor position.



**Figure 4.2. Content Assist Demonstration**

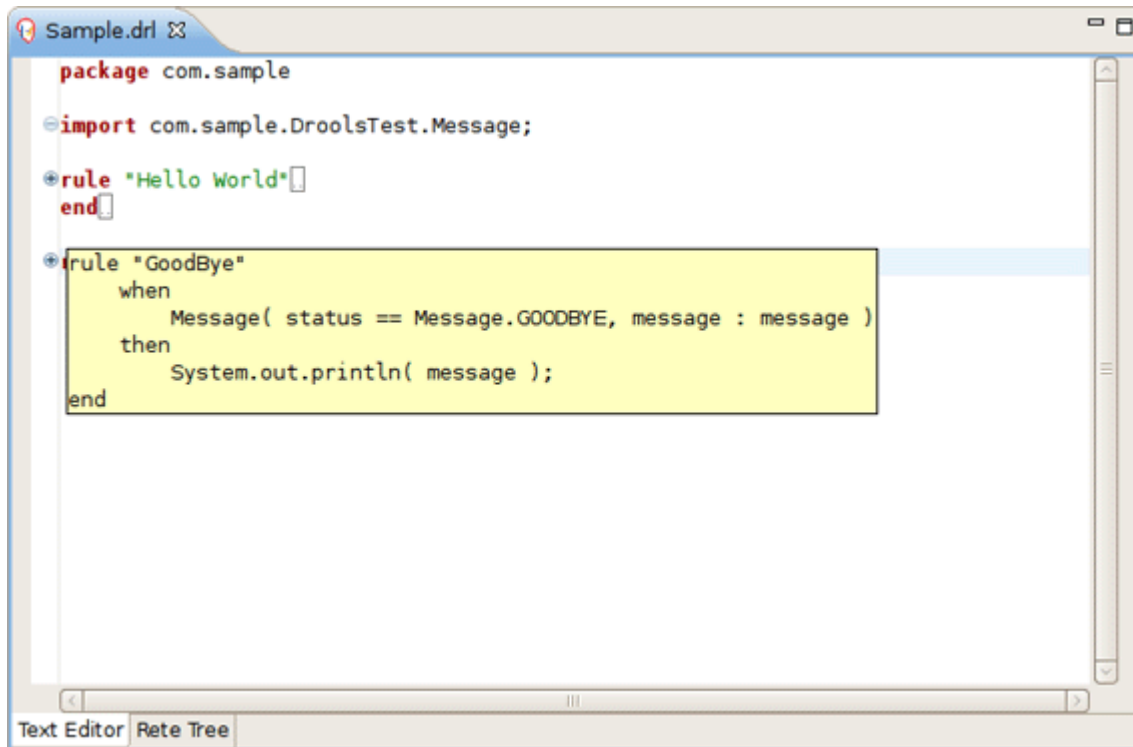
Content Assist inside of the `Message` suggests all available fields.



**Figure 4.3. Content Assist Demonstration**

### 4.1.2. Code Folding

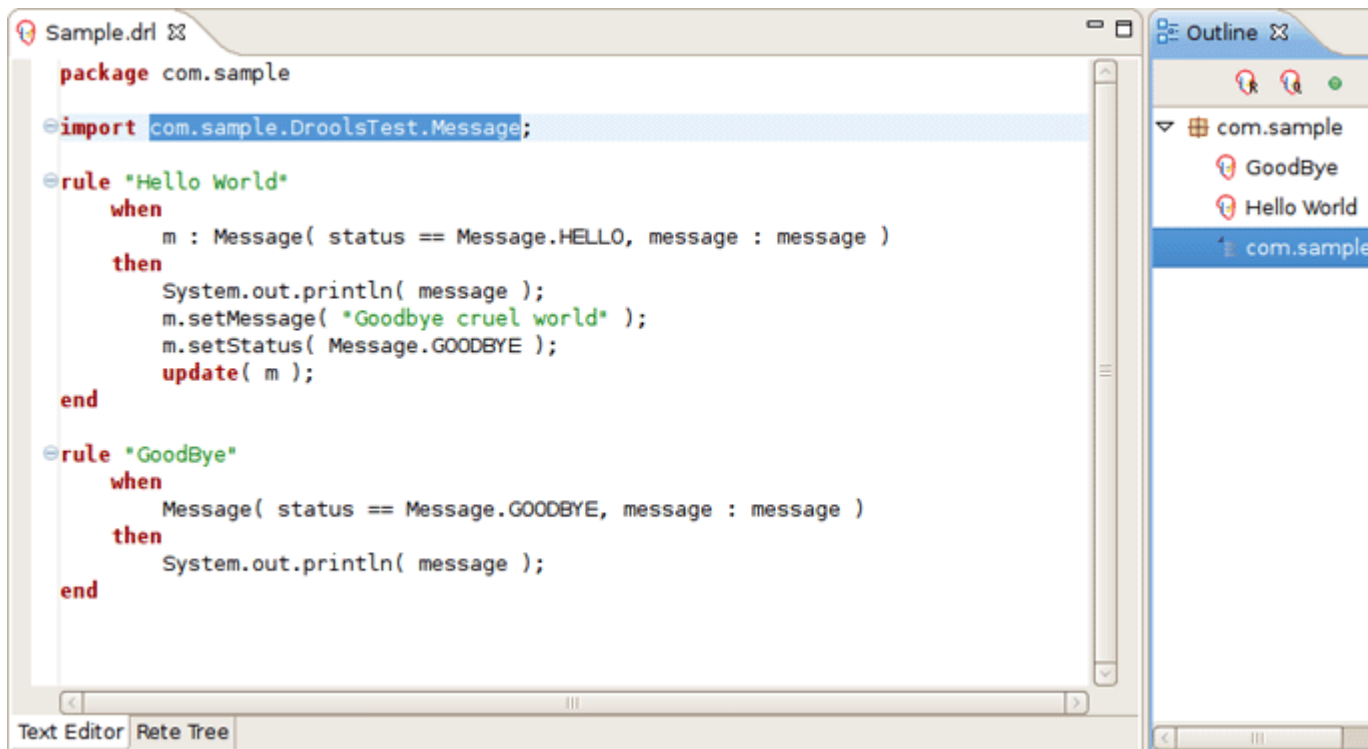
Code folding is also available in the [Rule editor](#). To hide/show sections of the file use the icons with minus/plus on the left vertical line of the editor.



**Figure 4.4. Code Folding**

### 4.1.3. Synchronization with Outline View

The [Rule editor](#) works in synchronization with the [Outline view](#) which shows the structure of the rules, imports in the file and also globals and functions if the file has them.

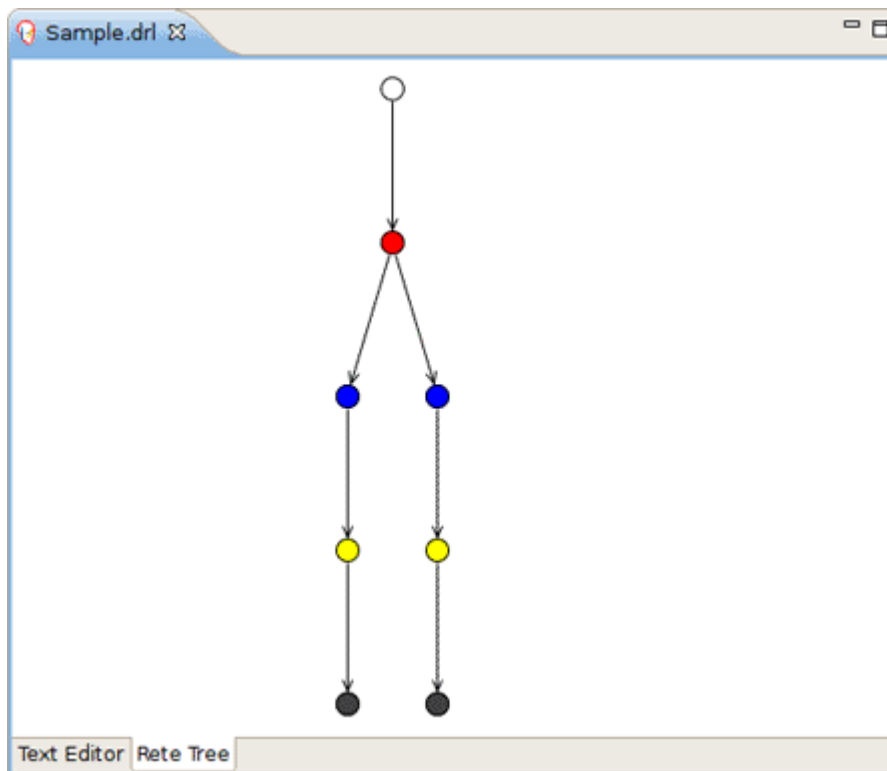


**Figure 4.5. Synchronization with Outline View**

The view is updated on save. It provides a quick way of navigating around rules by names in a file which may have hundreds of rules. The items are sorted alphabetically by default.

#### 4.1.4. The Rete Tree View

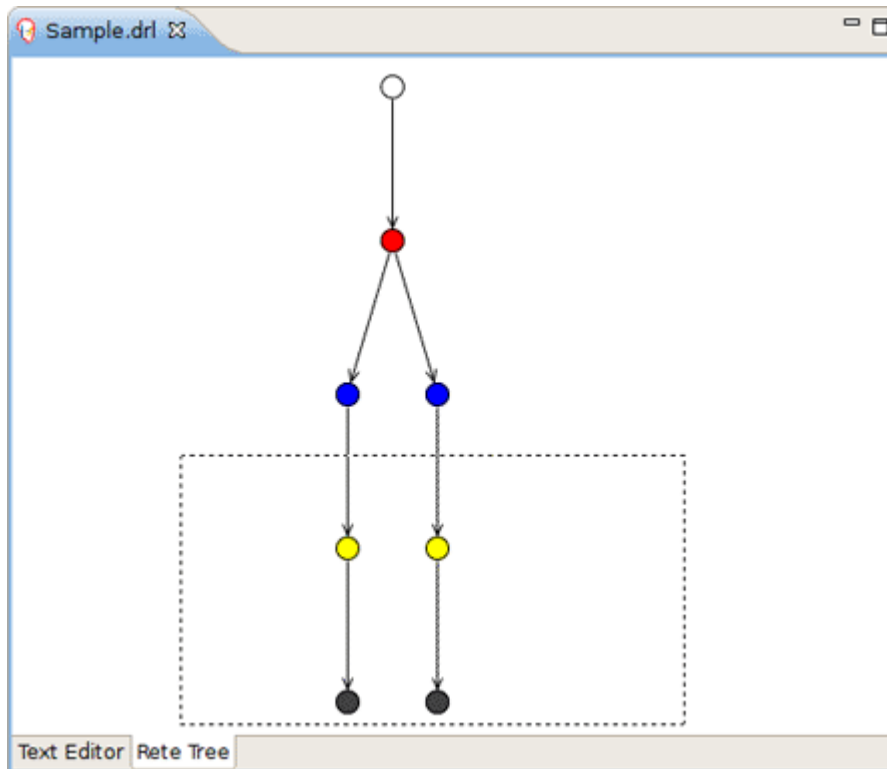
The [Rete Tree view](#) shows you the current Rete Network for your [.drl](#) file. Just click on the [Rete Tree tab](#) at the bottom of the [Rule editor](#).



**Figure 4.6. Rete Tree**

Afterwards you can generate the current Rete Network visualization. You can push and pull the nodes to arrange your optimal network overview.

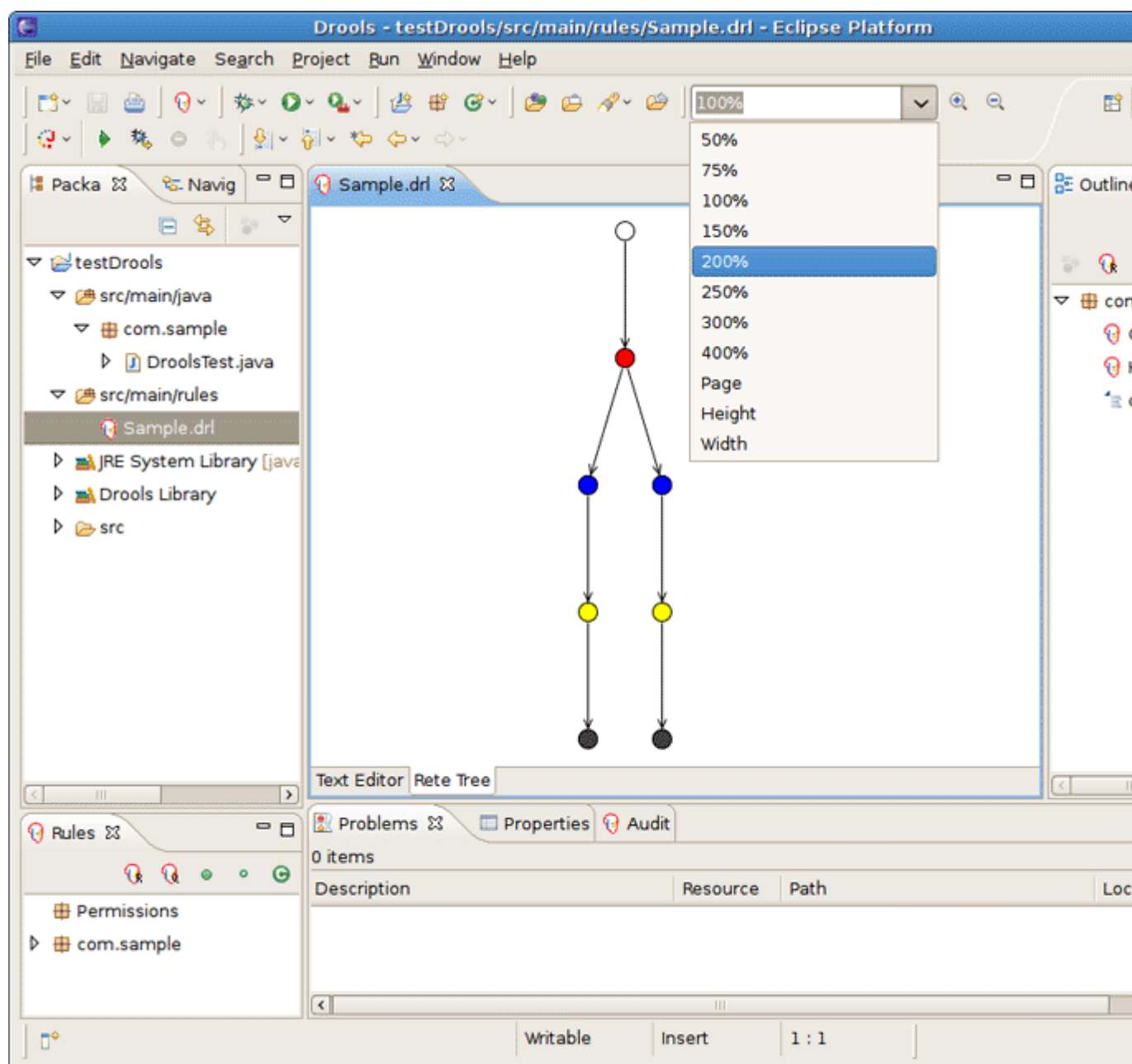
If you got hundreds of nodes, select some of them with a frame. Then you can pull groups of them.



**Figure 4.7. Selecting the nodes in the Rete Tree with Frame**

You can zoom in and out the Rete tree in case not all nodes are shown in the current view. For this use the combo box or "+" and "-" icons on the toolbar.



**Figure 4.8. Rete Tree Zooming****Note:**

The [Rete Tree view](#) works only in Drools Rule Projects, where the Drools Builder is set in the project properties.

We hope, this guide helped you to get started with the JBoss BPMN Convert module. Besides, for additional information you are welcome on [JBoss forum](http://www.jboss.com/index.html?module=bb&op=viewforum&f=201) [http://www.jboss.com/index.html?module=bb&op=viewforum&f=201].