
| | |
|---|----|
| 1. Drools 4.0 Release Notes | 1 |
| 1.1. What is new in Drools 4.0 | 1 |
| 1.1.1. Language Expressiveness Enhancements | 1 |
| 1.1.2. Core Engine Enhancements | 1 |
| 1.1.3. IDE Enhancements | 2 |
| 1.1.4. Business Rules Management System - BRMS | 2 |
| 1.1.5. Miscellaneous Enhancements | 2 |
| 1.2. Upgrade tips from Drools 3.0.x to Drools 4.0.x | 3 |
| 1.2.1. API changes | 3 |
| 1.2.2. Rule Language Changes | 4 |
| 1.2.3. Drools Update Tool | 5 |
| 1.2.4. DSL Grammars in Drools 4.0 | 5 |
| 1.2.5. Rule flow Update for 4.0.2 | 6 |
| 2. Installation and Setup (Core and IDE) | 7 |
| 2.1. Installing and using | 7 |
| 2.1.1. Dependencies and jars | 7 |
| 2.1.2. Runtime | 8 |
| 2.1.3. Installing IDE (Rule Workbench) | 8 |
| 2.2. Setup from source | 19 |
| 2.3. Source Checkout | 20 |
| 2.4. Build | 24 |
| 2.4.1. Building the Source | 24 |
| 2.4.2. Building the Manual | 27 |
| 2.5. Eclipse | 31 |
| 2.5.1. Generating Eclipse Projects | 31 |
| 2.5.2. Importing Eclipse Projects | 33 |
| 2.5.3. Exporting the IDE plug-in | 40 |
| 2.5.4. Building the update site | 48 |
| Index | 51 |

Chapter 1. Drools 4.0 Release Notes

1.1. What is new in Drools 4.0

Drools 4.0 is a major update over the previous Drools 3.0.x series. A whole new set of features were developed which special focus on language expressiveness, engine performance and tools availability. The following is a list of the most interesting changes.

1.1.1. Language Expressiveness Enhancements

- New Conditional Elements: from, collect, accumulate and forall
- New Field Constraint operators: not matches, not contains, in, not in, memberOf, not memberOf
- New Implicit Self Reference field: this
- Full support for Conditional Elements nesting, for First Order Logic completeness.
- Support for multi-restrictions and constraint connectives && and ||
- Parser improvements to remove previous language limitations, like character escaping and keyword conflicts
- Support for pluggable dialects and full support for MVEL scripting language
- Complete rewrite of DSL engine, allowing for full l10n
- Fact attributes auto-vivification for return value restrictions and inline-eval constraints
- Support for nested accessors, property navigation and simplified collection, arrays and maps syntax
- Improved support for XML rules

1.1.2. Core Engine Enhancements

- Native support for primitive types, avoiding constant autoboxing
- Support for transparent optional Shadow Facts
- Rete Network performance improvements for complex rules
- Support for Rule-Flows

- Support for Stateful and Stateless working memories (rule engine sessions)
- Support for Asynchronous Working Memory actions
- Rules Engine Agent for hot deployment and BRMS integration
- Dynamic salience for rules conflict resolution
- Support for Parameterized Queries
- Support for halt command
- Support for sequential execution mode
- Support for pluggable global variable resolver

1.1.3. IDE Enhancements

- Support for rule break-points on debugging
- WYSIWYG support for rule-flows
- New guided editor for rules authoring
- Upgrade to support all new engine features

1.1.4. Business Rules Management System - BRMS

- New BRMS tool
- User friendly web interface with nice WEB 2.0 ajax features
- Package configuration
- Rule Authoring easy to edit rules both with guided editor (drop-down menus) and text editor
- Package compilation and deployment
- Easy deployment with Rule Agent
- Easy to organize with categories and search assets
- Versioning enabled, you can easily replace yours assets with previously saved
- JCR compliant rule assets repository

1.1.5. Miscellaneous Enhancements

- Slimmed down dependencies and smaller memory footprint

1.2. Upgrade tips from Drools 3.0.x to Drools 4.0.x

As mentioned before Drools 4.0 is a major update over the previous Drools 3.0.x series. Unfortunately, in order to achieve the goals set for this release, some backward compatibility issues were introduced, as discussed in the mail list and blogs.

This section of the manual is a work in progress and will document a simple how-to on upgrading from Drools 3.0.x to Drools 4.0.x.

1.2.1. API changes

There are a few API changes that are visible to regular users and need to be fixed.

1.2.1.1. Working Memory creation

Drools 3.0.x had only one working memory type that worked like a stateful working memory. Drools 4.0.x introduces separate APIs for Stateful and Stateless working memories that are called now Rule Sessions. In Drools 3.0.x, the code to create a working memory was:

Example 1.1. Drools 3.0.x: Working Memory Creation

```
WorkingMemory wm = rulebase.newWorkingMemory();
```

In Drools 4.0.x it must be changed to:

Example 1.2. Drools 4.0.x: Stateful Rule Session Creation

```
StatefulSession wm = rulebase.newStatefulSession();
```

The StatefulSession object has the same behavior as the Drools 3.0.x WorkingMemory (it even extends the WorkingMemory interface), so there should be no other problems with this fix.

1.2.1.2. Working Memory Actions

Drools 4.0.x now supports pluggable dialects and has built-in support for Java and MVEL scripting language. In order to avoid keyword conflicts, the working memory actions were renamed as showed bellow:

Table 1.1. Working Memory Actions equivalent API methods

| Drools 3.0.x | Drools 4.0.x |
|-------------------------------------|-------------------------------|
| WorkingMemory.assertObject() | WorkingMemory.insert() |
| WorkingMemory.assertLogicalObject() | WorkingMemory.insertLogical() |

| | |
|------------------------------|------------------------|
| WorkingMemory.modifyObject() | WorkingMemory.update() |
|------------------------------|------------------------|

1.2.2. Rule Language Changes

The DRL Rule Language also has some backward incompatible changes as detailed bellow.

1.2.2.1. Working Memory Actions

The Working Memory actions in rule consequences were also changed in a similar way to the change made in the API. The following table summarizes the change:

Table 1.2. Working Memory Actions equivalent DRL commands

| Drools 3.0.x | Drools 4.0.x |
|-----------------|-----------------|
| assert() | insert() |
| assertLogical() | insertLogical() |
| modify() | update() |

1.2.2.2. Primitive support and unboxing

Drools 3.0.x did not had native support for primitive types and consequently, it auto-boxed all primitives in it's respective wrapper classes. That way, any use of a boxed variable binding required a manual unbox.

Drools 4.0.x has full support for primitive types and does not wrap values anymore. So, all previous unwrap method calls must be removed from the DRL.

Example 1.3. Drools 3.0.x manual unwrap

```
rule "Primitive int manual unbox"
when
    $c : Cheese( $price : price )
then
    $c.setPrice( $price.intValue() * 2 )
end
```

The above rule in 4.0.x would be:

Example 1.4. Drools 4.0.x primitive support

```
rule "Primitive support"
when
    $c : Cheese( $price : price )
then
    $c.setPrice( $price * 2 )
end
```


1.2.3. Drools Update Tool

The Drools Update tools is a simple program to help with the upgrade of DRL files from Drools 3.0.x to Drools 4.0.x.

At this point, its main objective is to upgrade the memory action calls from 3.0.x to 4.0.x, but expect it to grow over the next few weeks covering additional scenarios. It is important to note that it does not make a dumb text search and replace in rules file, but it actually parses the rules file and try to make sure it is not doing anything unexpected, and as so, it is a safe tool to use for upgrade large sets of rule files.

The Drools update tool can be found as a maven project in the following source repository <http://anonsvn.labs.jboss.com/labs/jbossrules/trunk/experimental/drools-update/> you just need to check it out, and execute the maven clean install action with the project's pom.xml file. After resolve all the class path dependencies you are able to run the toll with the following command:

```
java -cp $CLASSPATH org.drools.tools.update.UpdateTool -f <filemask> [-d <basedir>] [-s <suffix>]
```

The program parameters are very easy to understand as following.

- -h,--help, Shows a very simple list the usage help
- -d your source base directory
- -f pattern for the files to be updated. The format is the same as used by ANT: * = single file, directory ** = any level of subdirectories EXAMPLE: src/main/resources/**/*.drl = matches all DRL files inside any subdirectory of /src/main/resources
- -s,--suffix the suffix to be added to all updated files

1.2.4. DSL Grammars in Drools 4.0

It is important to note that the DSL template engine was rewritten from scratch to improve flexibility. One of the new features of DSL grammars is the support to Regular Expressions. This way, now you can write your mappings using regexp to have additional flexibility, as explained in the DSL chapter. Although, now you have to escape characters with regexp meaning. Example: if previously you had a matching like:

Example 1.5. Drools 3.0.x mapping

```
[when][]- the {attr} is in [ {values} ]={attr} in ( {values} )
```

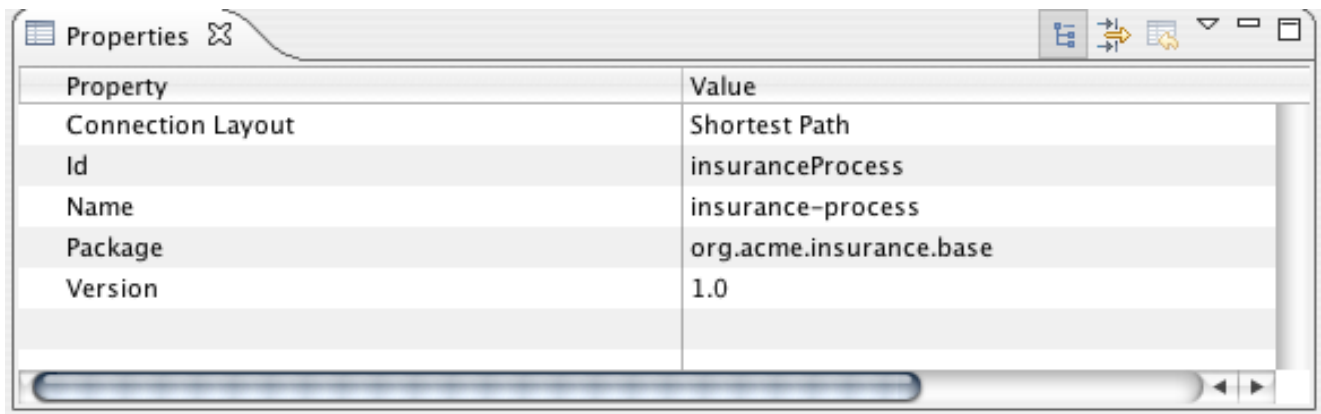
Now, you need to escape '[' and ']' characters, as they have special meaning in regexps. So, the same mapping in Drools 4.0 would be:

Example 1.6. Drools 4.0.x mapping with escaped characters

```
[when][]- the {attr} is in \[ {values} \]={attr} in ( {values} )
```

1.2.5. Rule flow Update for 4.0.2

The Rule flow feature was updated for 4.0.2, and now all your ruleflows must declare a package name.



| Property | Value |
|-------------------|-------------------------|
| Connection Layout | Shortest Path |
| Id | insuranceProcess |
| Name | insurance-process |
| Package | org.acme.insurance.base |
| Version | 1.0 |

Figure 1.1. Rule Flow properties

Chapter 2. Installation and Setup

(Core and IDE)

2.1. Installing and using

Drools provides an Eclipse-based IDE (which is optional), but at its core only Java 1.4 (J2SE) is required.

A simple way to get started is to download and install the Eclipse plug-in - this will also require the Eclipse GEF framework to be installed (see below, if you don't have it installed already). This will provide you with all the dependencies you need to get going: you can simply create a new rule project and everything will be done for you. Refer to the chapter on the Rule Workbench and IDE for detailed instructions on this. Installing the Eclipse plug-in is generally as simple as unzipping a file into your Eclipse plug-in directory.

Use of the Eclipse plug-in is not required. Rule files are just textual input (or spreadsheets as the case may be) and the IDE (also known as the Rule Workbench) is just a convenience. People have integrated the rule engine in many ways, there is no "one size fits all".

Alternatively, you can download the binary distribution, and include the relevant jars in your projects classpath.

2.1.1. Dependencies and jars

Drools is broken down into a few modules, some are required during rule development/compiling, and some are required at runtime. In many cases, people will simply want to include all the dependencies at runtime, and this is fine. It allows you to have the most flexibility. However, some may prefer to have their "runtime" stripped down to the bare minimum, as they will be deploying rules in binary form - this is also possible. The core runtime engine can be quite compact, and only require a few 100 kilobytes across 2 jar files.

The following is a description of the important libraries that make up JBoss Rules

- drools-core.jar - this is the core engine, runtime component. Contains both the RETE engine and the LEAPS engine. This is the only runtime dependency if you are pre-compiling rules (and deploying via Package or RuleBase objects).
- drools-compiler.jar - this contains the compiler/builder components to take rule source, and build executable rule bases. This is often a runtime dependency of your application, but it need not be if you are pre-compiling your rules. This depends on drools-core
- drools-jsr94.jar - this is the JSR-94 compliant implementation, this is essentially a layer over the drools-compiler component. Note that due to the nature of the JSR-94 specification, not all

features are easily exposed via this interface. In some cases, it will be easier to go direct to the Drools API, but in some environments the JSR-94 is mandated.

- `drools-decisiontables.jar` - this is the decision tables 'compiler' component, which uses the `drools-compiler` component. This supports both excel and CSV input formats.

There are quite a few other dependencies which the above components require, most of which are for the `drools-compiler`, `drools-jsr94` or `drools-decisiontables` module. Some of these (such as the XML libraries) may not be required if you run in a Java 1.5 environment. Some key ones to note are "JCI" - which is the apache Java Compiler Interface utility which provides runtime compiling capability, "POI" which provides the spreadsheet parsing ability, and "antlr" which provides the parsing for the rule language itself.

NOTE: if you are using Drools in J2EE or servlet containers and you come across classpath issues with "JDT", then you can switch to the janino compiler. Set the system property "drools.compiler": For example: `-Ddrools.compiler=JANINO`.

For up to date info on dependencies in a release, consult the `README_DEPENDENCIES.txt` file, which can be found in the lib directory of the download bundle, or in the root of the project directory.

2.1.2. Runtime

The "runtime" requirements mentioned here are if you are deploying rules as their binary form (either as Package objects, or RuleBase objects etc). This is an optional feature that allows you to keep your runtime very light. You may use `drools-compiler` to produce rule packages "out of process", and then deploy them to a runtime system. This runtime system only requires `drools-core.jar` for execution. This is an optional deployment pattern, and many people do not need to "trim" their application this much, but it is an ideal option for certain environments.

2.1.3. Installing IDE (Rule Workbench)

The rule workbench (for Eclipse) requires that you have Eclipse 3.2 or greater, as well as Eclipse GEF 3.2 or greater. You can install it either by downloading the plug-in or, or using the update site.

Another option is to use the JBoss IDE, which comes with all the plug-in requirements pre packaged, as well as a choice of other tools separate to rules. You can choose just to install rules from the "bundle" that JBoss IDE ships with.

2.1.3.1. Installing GEF (a required dependency)

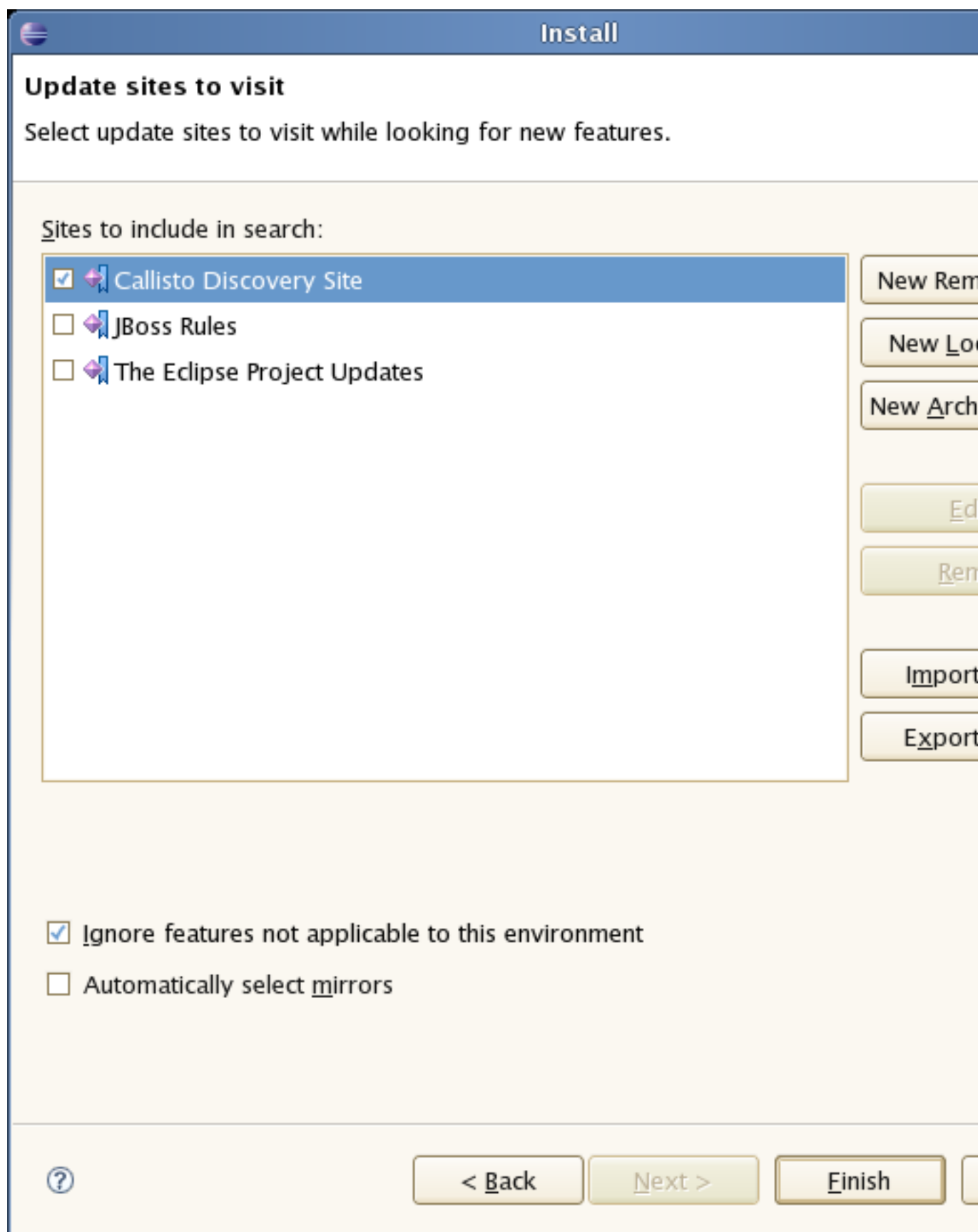
GEF is the Eclipse Graphical Editing Framework, which is used for graph viewing components in the plug-in.

If you don't have GEF installed, you can install it using the built in update mechanism (or downloading GEF from the Eclipse.org website not recommended). JBoss IDE has GEF already, as do many other "distributions" of Eclipse, so this step may be redundant for some people.

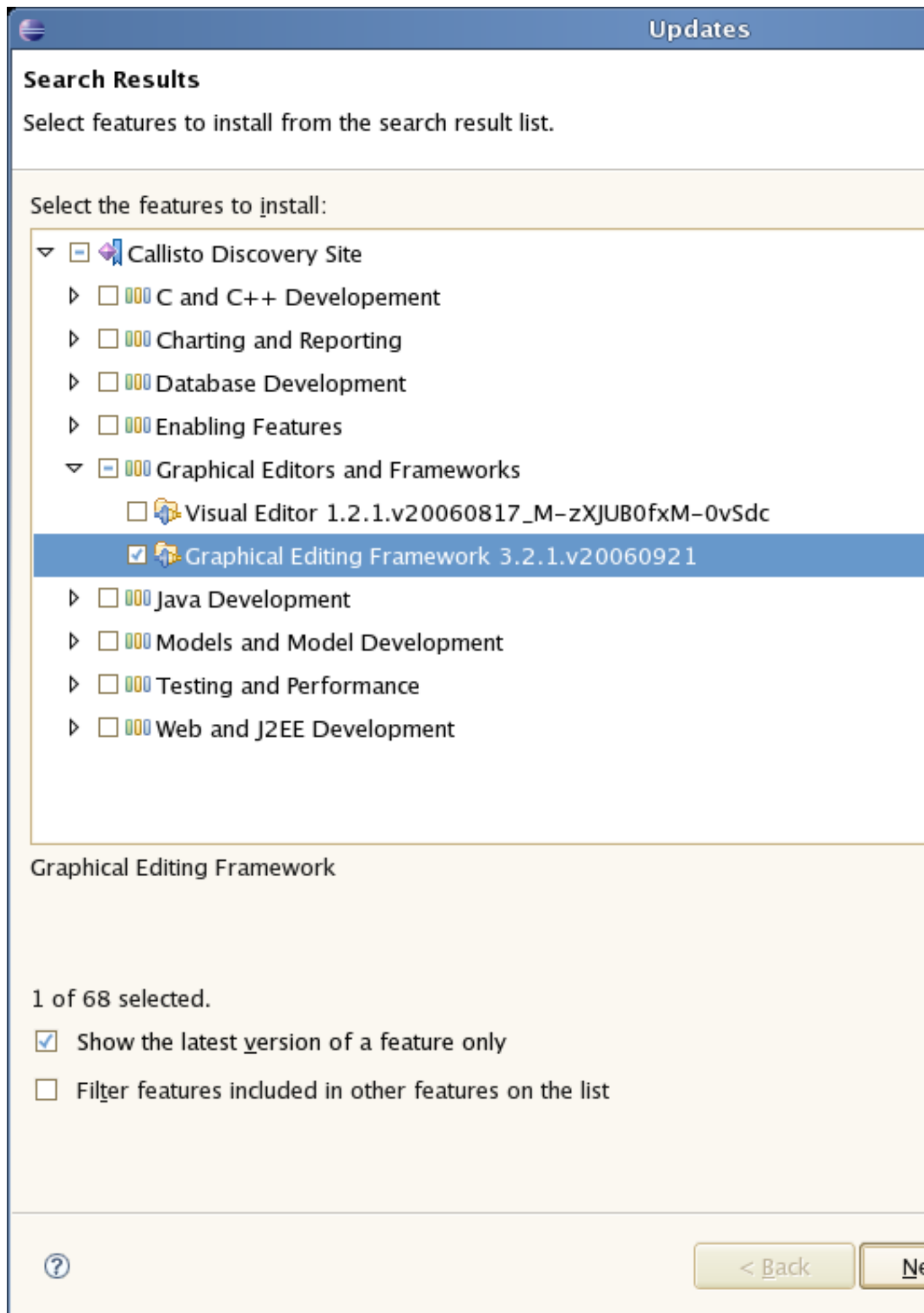
First you open the Help->Software updates->Find and install from the help menu. Then you choose the Calisto update site:

If you aren't using Calisto you can use the following update site to download GEF

```
http://europa-mirror1.Eclipse.org/tools/gef/update-site/releases/
```



Next you choose the GEF plug-in:



Press next, and agree to install the plug-in (an Eclipse restart may be required). Once this is completed, then you can continue on installing the rules plug-in.

2.1.3.2. Installing from zip file

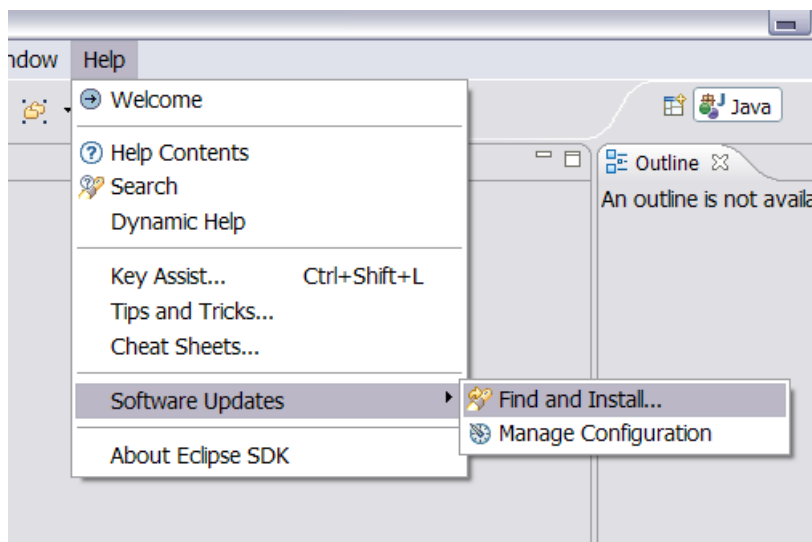
To install from the zip file, download and unzip the file. Inside the zip you will see a plug-in directory, and the plug-in jar itself. You place the plug-in jar into your Eclipse applications plug-in directory, and restart Eclipse.

2.1.3.3. Installing from the update site

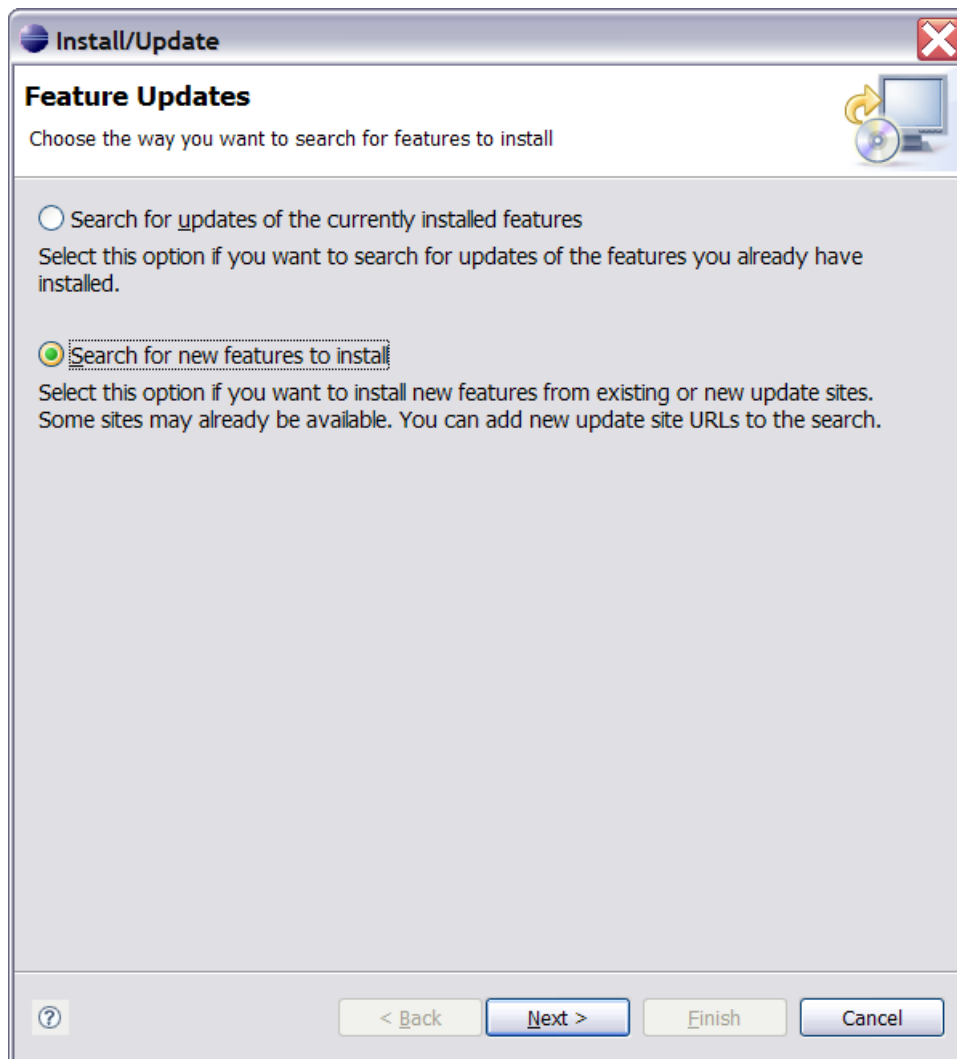
Using the update site is a handy way to install the plug-in, and keep it up to date (the Eclipse platform will check for updates as needed). It gives you a good chance of staying up to date with improvements, fixes etc.

Some firewalls may cause trouble with using update sites in Eclipse, if you have issues, then install it manually from the plug-in. Also, if you have previously installed the plug-in manually, you will need to manually remove it from your plug-in directory.

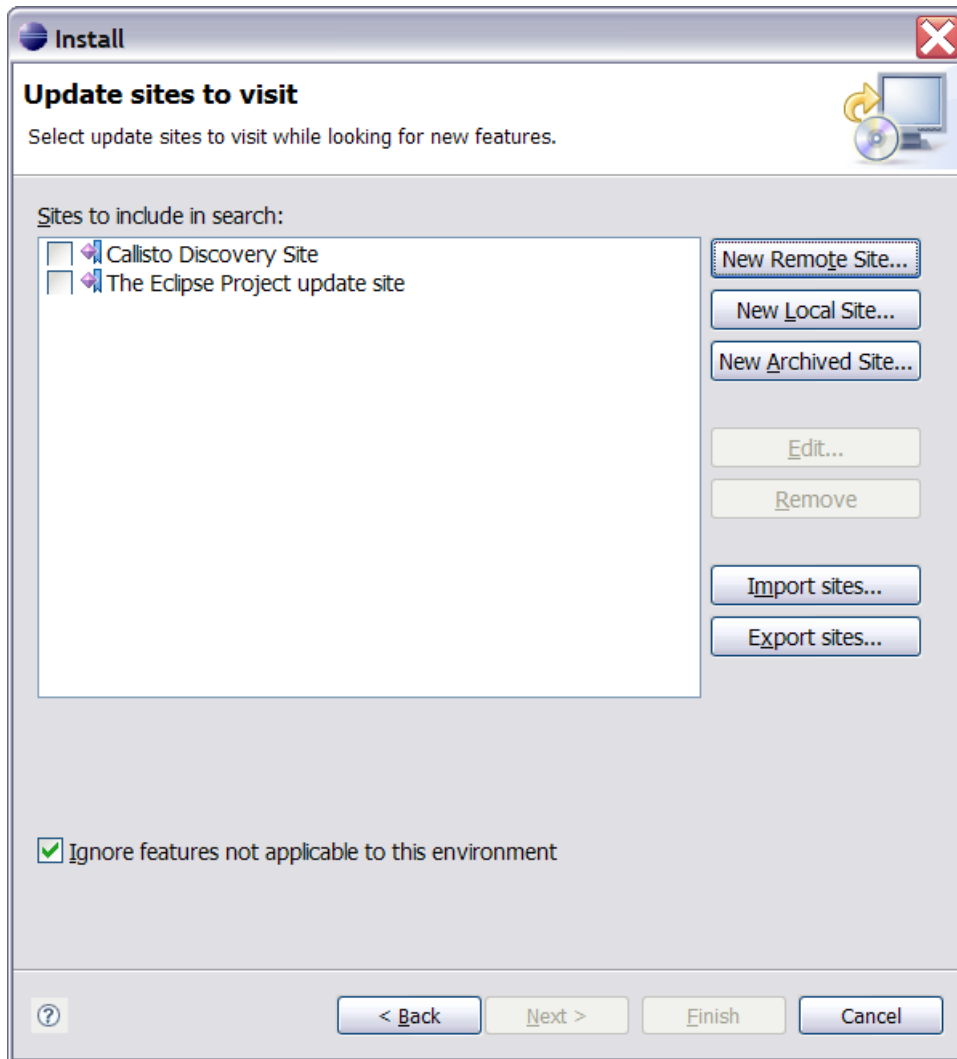
Step 1. Use the Eclipse help menu to find the feature installer.



Step 2: Choose the option for installing a new feature (obviously in future, if you want to check for updates, you use the other option !).

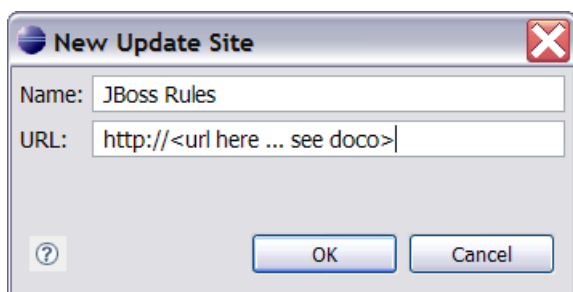


Step 3: This screen will show what update sites are already configured for your Eclipse instance.

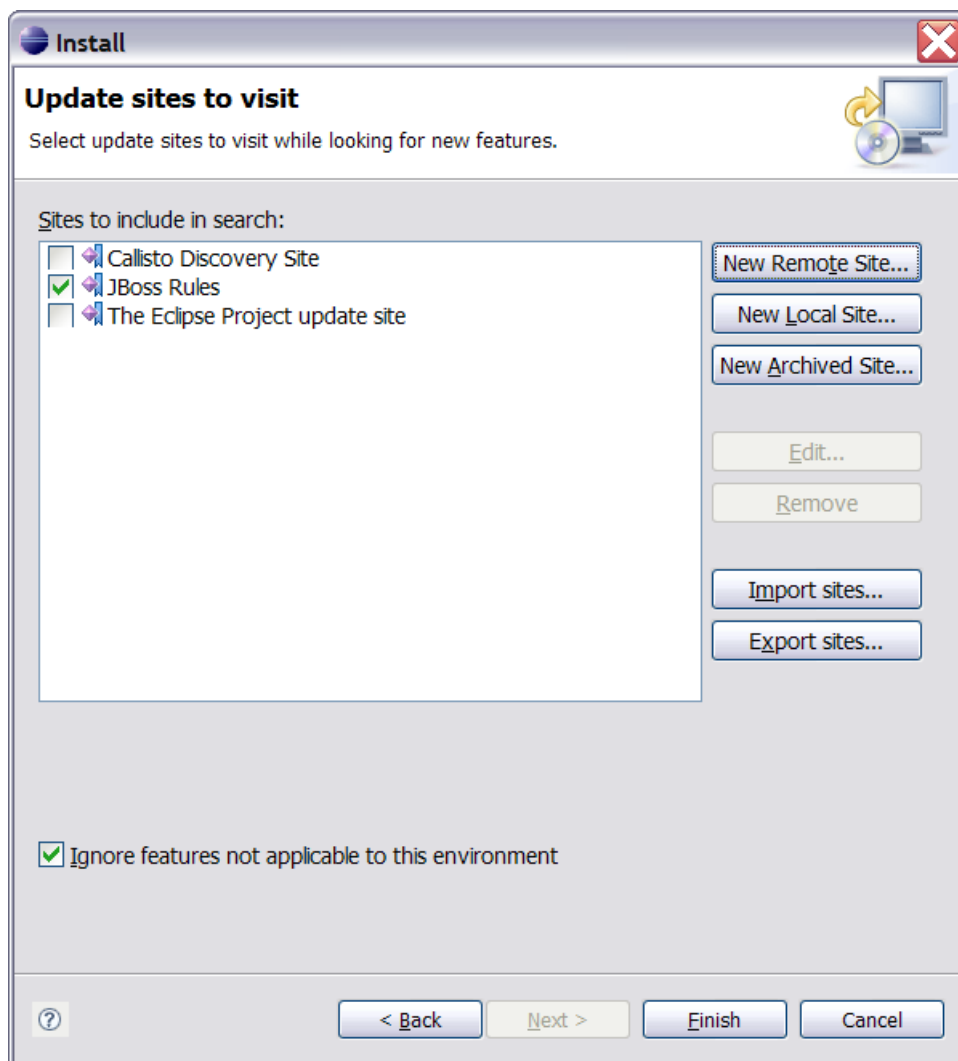


Step 4: This screen is where you enter in the remote site details. You give it a name eg "JBoss Drools" and the url.

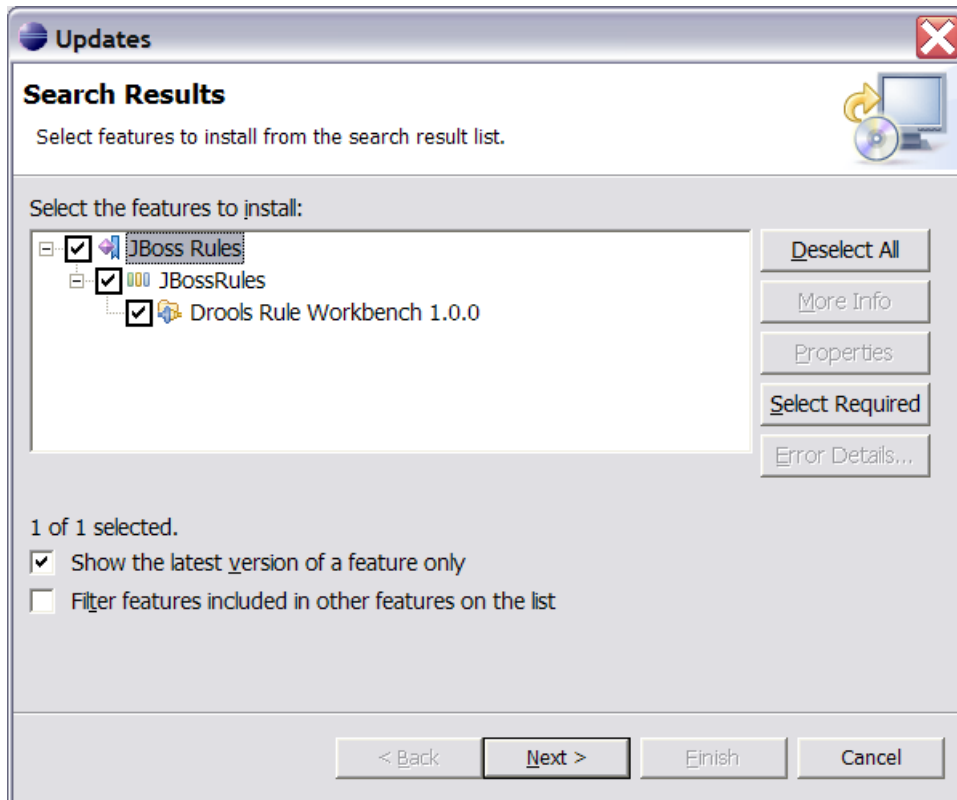
```
Check the Drools IDE Update Site section of the Drools Downloads webpage:  
http://labs.jboss.com/drools/downloads.html
```



Step 5: Select the new update site you just added. Eclipse will remember this for when it checks for updates automatically in the future.



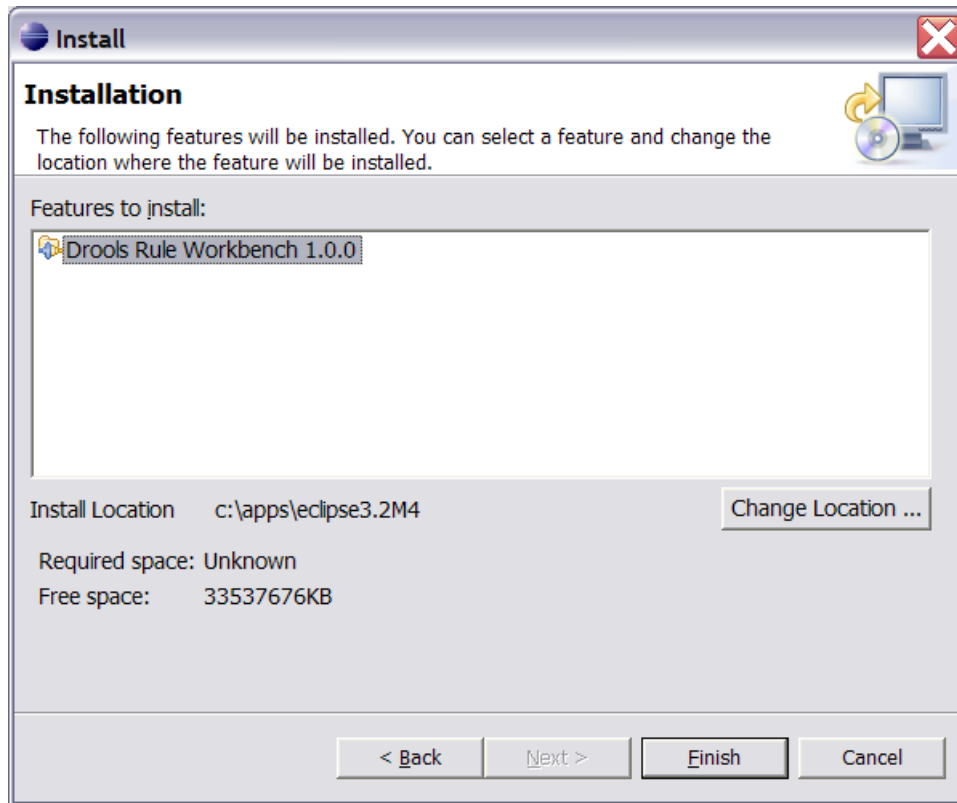
Step 6: You should see the available features (Drools IDE) retrieved from the update site.



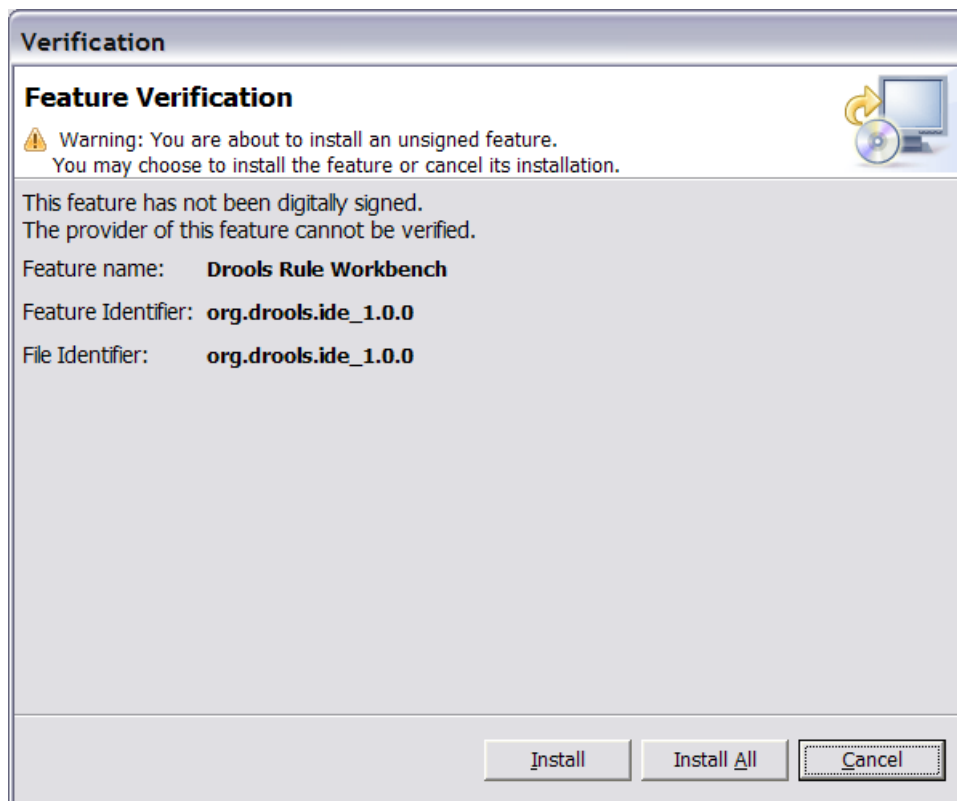
Step 7: The license agreement. Choose the option to accept the license agreement. Once this happens, the workbench will start downloading. Might be an opportune time to go have a coffee.



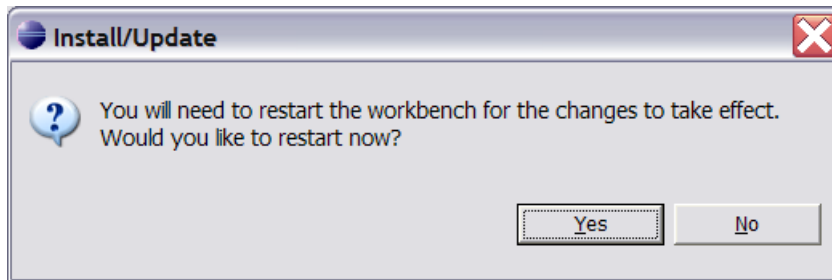
Step 8: Confirm that this is the feature you want.



Step 9: Press Accept to accept the fact that the feature is not digitally signed. No one signs their features, its a silly default screen in Eclipse.



Step 10: The workbench will need to restart now for the feature to take effect.



Now go have another coffee, and then take a look at the chapter on the Rule Workbench for what you can do with it.

2.2. Setup from source

As Drools is an open source project, instructions for building from source are part of the manual ! Building from source means you can stay on top with the latest features. Whilst aspects of Drools are quite complicated, many users have found ways to become contributors.

Drools works with JDK1.5 and above. you will need also need to have the following tools installed. Minimum requirement version numbers provided.

- Eclipse 3.2

<http://www.eclipse.org/>

- Subversion Client 1.3

<http://subversion.tigris.org>

<http://tortoisesvn.tigris.org> - recommended win32 client

- Maven 2.0.7

<http://maven.apache.org/>

- Ant 1.7.0

<http://ant.apache.org>

Ensure the executables for ant, maven and java are in your path. The examples given illustrative and are for a win32 system:

```
Path=D:\java\jdk1.5.0_8\bin;D:\java\apache-ant-1.7\bin;D:\java\maven-2.0.7\bin
```

Following environment variables will also need to be set. The examples given illustrative and are for a win32 system::

```
JAVA_HOME=D:\java\jdk1.5.0_8
```

```
ANT_HOME=D:\java\apache-ant-1.6.5
MAVEN_HOME=D:\java\maven-2.0.7
```

Past releases used to have an ant based build mechanism, but now maven is mandatory, although Ant is used internally in maven for document building proposes

2.3. Source Checkout

Drools is available from two Subversion repositories.

- Anonymous SVN

<http://anonsvn.labs.jboss.com/labs/jbossrules/trunk/>

- Developers secured SVN

<https://svn.labs.jboss.com/labs/jbossrules/trunk/>

To checkout Drools source code just execute the following command.

```
fmeyer:~/jboss $ svn checkout
http://anonsvn.labs.jboss.com/labs/jbossrules/trunk/ jbossrules
```

And wait to complete the files download.

```
A    jbossrules/drools-repository
A    jbossrules/drools-repository/.classpath
A    jbossrules/drools-repository/.project
A    jbossrules/drools-repository/doc
A    jbossrules/drools-repository/doc/repository_layout.jpeg
A    jbossrules/drools-repository/doc/high_level_design.jpeg
A    jbossrules/drools-repository/doc/javadoc
A    jbossrules/drools-repository/doc/javadoc/serialized-form.html
A    jbossrules/drools-repository/doc/javadoc/index-all.html
A    jbossrules/drools-repository/doc/javadoc/stylesheet.css
A    jbossrules/drools-repository/doc/javadoc/allclasses-frame.html
A    jbossrules/drools-repository/doc/javadoc/package-list
A    jbossrules/drools-repository/doc/javadoc/overview-tree.html
A    jbossrules/drools-repository/doc/javadoc/org
A    jbossrules/drools-repository/doc/javadoc/org/drools
A    jbossrules/drools-repository/doc/javadoc/org/drools/repository
A
    jbossrules/drools-repository/doc/javadoc/org/drools/repository/class-use
A
    jbossrules/drools-repository/doc/javadoc/org/drools/repository/class-use/
RuleSet.html
A
    jbossrules/drools-repository/doc/javadoc/org/drools/repository/class-use/
RulesRepositoryException.html
```



```
A
  jbossrules/drools-repository/doc/javadoc/org/drools/repository/class-use/
RulesRepository.html
A
  jbossrules/drools-repository/doc/javadoc/org/drools/repository/RuleSet.html

....

snip

....

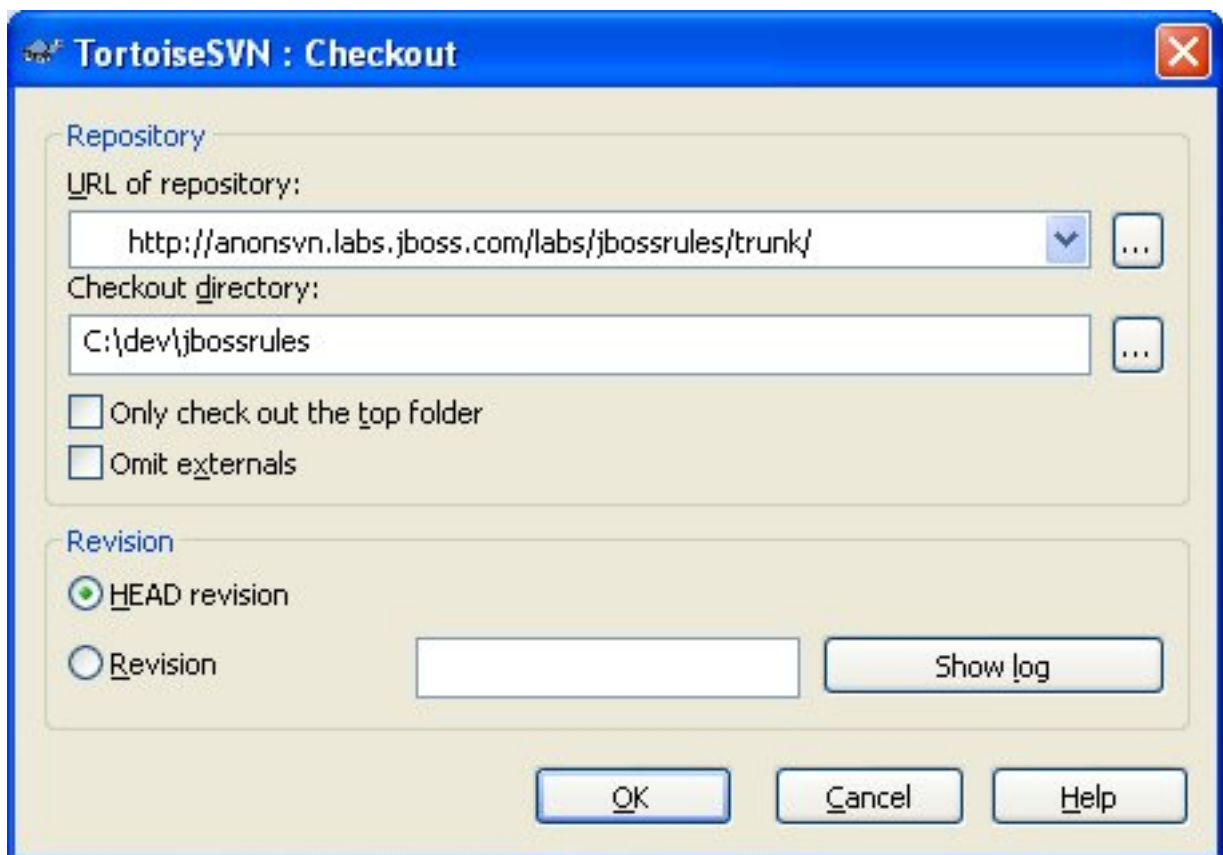
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/waltz
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/waltz/waltz.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/manners
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/manners/manners.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/waltzdb
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
benchmark/waltzdb/waltzdb.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/TroubleTicketWithDSL.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/TroubleTicket.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/calculate.rfm
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/generation.rf
```

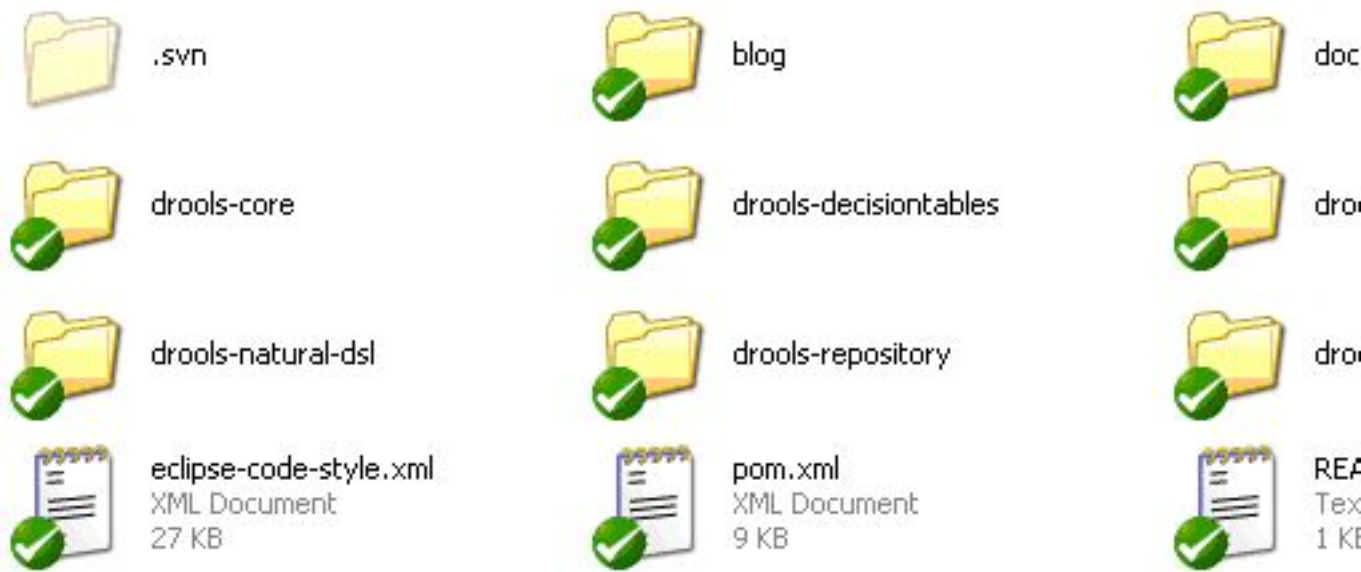
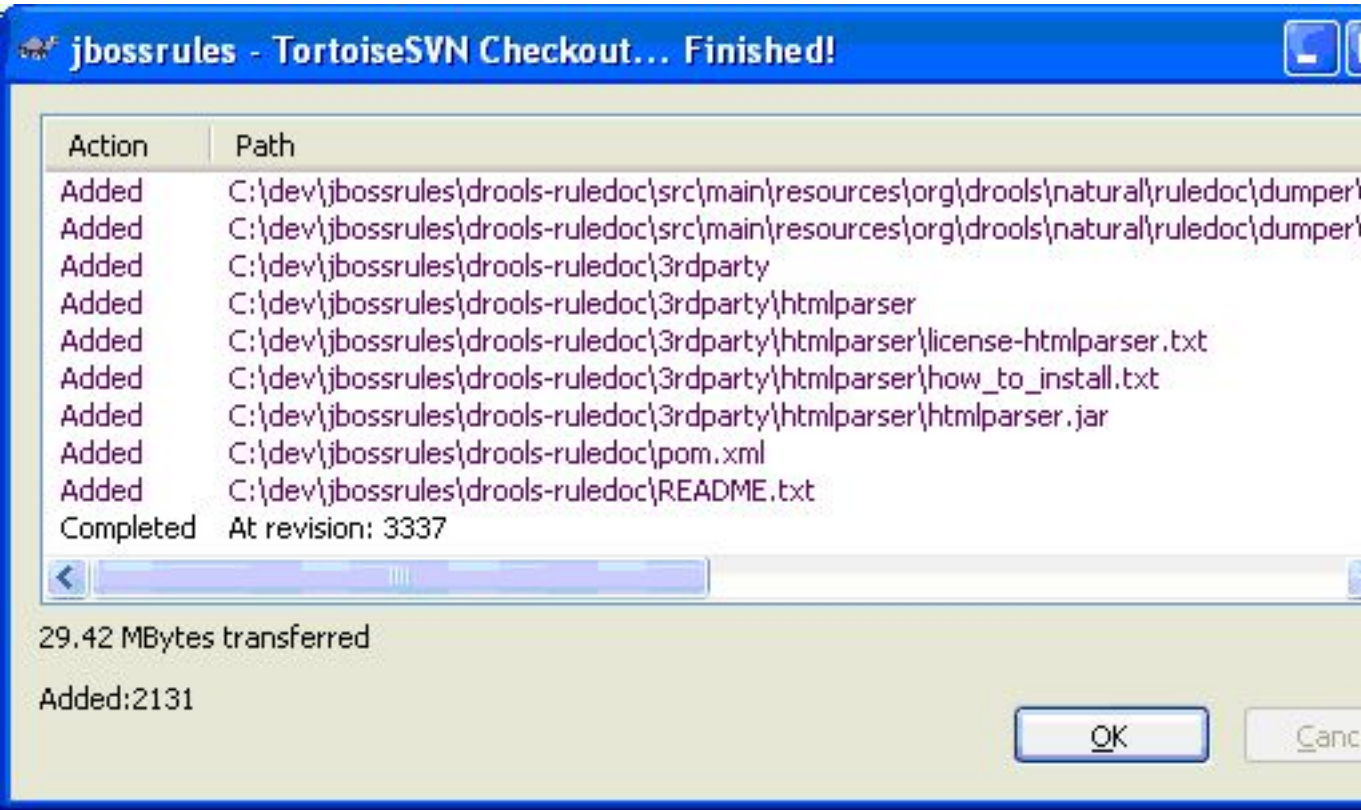
```
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/calculate.rf
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/registerNeighbor.rfm
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/killAll.rfm
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/registerNeighbor.rf
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/conway-agendagroup.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/killAll.rf
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/conway-ruleflow.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/conway/generation.rfm
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/ticketing.dsl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/StateExampleUsingSalience.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/golf.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/LogicalAssertionsNotPingPong.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/StateExampleDynamicRule.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/sudoku
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/sudoku/sudoku.drl
A
  jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/HelloWorld.drl
```

```
A      jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/ExamplePolicyPricing.xls
A      jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/HonestPolitician.drl
A      jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/Fibonacci.drl
A      jbossrules/drools-examples/drools-examples-drl/src/main/rules/org/drools/
examples/StateExampleUsingAgendGroup.drl
A      jbossrules/drools-examples/drools-examples-drl/pom.xml
A      jbossrules/drools-examples/drools-examples-drl/build.xml
U      jbossrules
Checked out revision 13656.
```

Although, we highly recommend command line tools to work with repository you can also use both Eclipse's integrated SVN client or TortoiseSVN

Setup TortoiseSVN to checkout from the subversion repository and click 'OK' Once the checkout has finished you should see the folders as shown below.





2.4. Build

2.4.1. Building the Source

Now that we have the source the next step is to build and install the source. Since version 3.1 Drools uses `mvn` to build the system. There are two profiles available which enable the associated modules "documentation" and "Eclipse"; this enables quicker building of the core modules for developers. The Eclipse profile will download Eclipse into the drools-Eclipse folder, which is over

100MB download (It depends on your operating system), however this only needs to be done once; if you wish you can move that Eclipse download into another location and specify it with `-DlocalEclipseDrop=/folder/jboss-rules/local-Eclipse-drop-mirror`. The following builds all the jars, the documentation and the Eclipse zip with a local folder specified to avoid downloading Eclipse:

```
mvn -Declipse -Ddocumentation clean install
-DlocalEclipseDrop=/folder/jboss-rules/local-Eclipse-drop-mirror
```

You can produce distribution builds, which puts everything into zips, as follows:

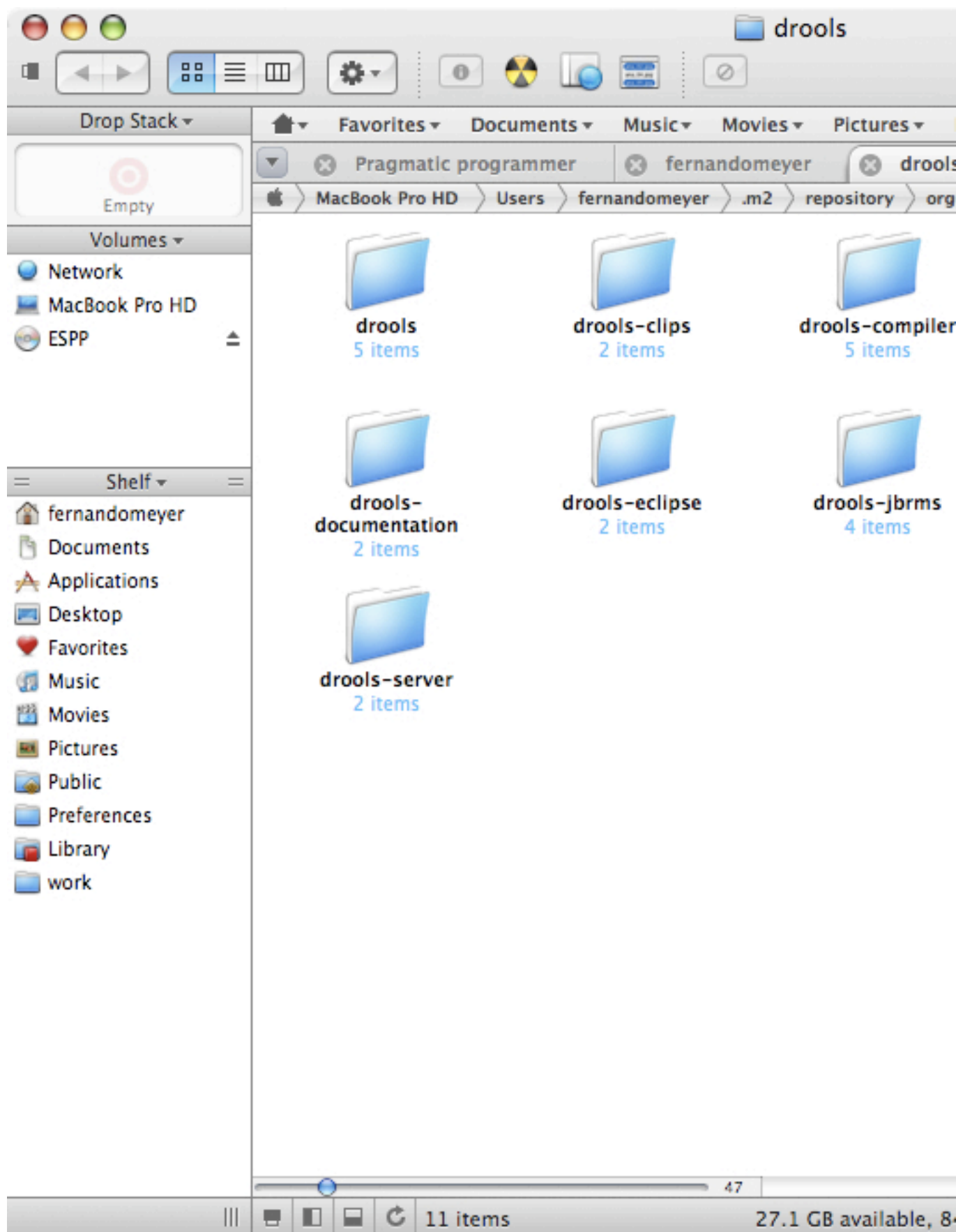
```
mvn -Declipse -Ddocumentation clean install
-DlocalEclipseDrop=/folder/jboss-rules/local-Eclipse-drop-mirror
mvn -Ddocumentation -Declipse -Dmaven.test.skip
package javadoc:javadoc assembly:assembly
-DlocalEclipseDrop=/folder/jboss-rules/local-Eclipse-drop-mirror
```

Note that install must be done first as javadoc:javadoc won't work unless the jars are in the local maven repo, but the tests can be skipped on the second run. `assembly:assembly` fails unless you increase the available memory to Maven, on windows the following command worked well: set `MAVEN_OPTS=-Xmx512m`

Type `mvn clean` to clear old artifacts, and then test and built the source, and report on any errors.

The resulting jars are put in the `/target` directory from the top level of the project.

As maven builds each module it will install the resulting jars in the local Maven 2 repository automatically. Where it can be easily used from other project `pom.xml` or copied else where.



2.4.2. Building the Manual

The building of the manual is now integrated into the maven build process, and is built by either using the profile (-Ddocumentation) switch or cding into the main directory. The manual can still be built from ant command line too by cding into the documentation/manual itself.

Drools uses Docbook for this manual. Ant is used internally in maven to build documents and this build produces three different formats, all sharing the same images directory.

- html_single

The entire manual in a single html document

- html

The manual is split into multiple documents and placed in a frameset. The left frame provides navigation

- Eclipse

Documentation suitable for including in an Eclipse plug-in

The manual can be generated from the project pom.xml by calling 'mvn package' in the documentation directory or adding the -Ddocumentation switch when you build the sources, with the generated documentation being copied to 'target/'. What actually happens is that maven call a separate ant build.xml for the manual, located at documentation/manual; the documentation is generated into documentation/manual/build before being copied to 'target/'.

```
fmeyer:~/projects/jbosrules/documentation $ mvn clean package
[INFO] Scanning for projects...
[INFO]
-----
-
[INFO] Building Drools :: Documentation
[INFO]    task-segment: [install]
[INFO]
-----
-
[INFO] [antrun:run {execution: manual}]
[INFO] Executing tasks
[delete] Deleting directory
/Users/fernandomeyer/projects/jbosrules/documentation/manual/build

clean:

all.doc:

lang.all:

lang.misc:
```

```
[copy] Copying 188 files to
/Users/fernandomeyer/projects/jbossrules/documentation/manual/build/en/
shared/images
[copy] Copying 1 file to
/Users/fernandomeyer/projects/jbossrules/documentation/manual/build/en/
shared/css

lang.dochtml:
[mkdir] Created dir:
/Users/fernandomeyer/projects/jbossrules/documentation/manual/build/en/html
[copy] Copying 1 file to
/Users/fernandomeyer/projects/jbossrules/documentation/manual/build/en/html
[java] Writing bk01-toc.html for book
[java] Writing pr01.html for preface(preface)
[java] Writing ch01s02.html for section
[java] Writing ch01s03.html for section
[java] Writing ch01s04.html for section
[java] Writing ch01s05.html for section
[java] Writing ch01s06.html for section
[java] Writing ch01.html for chapter
[java] Writing ch02s02.html for section
[java] Writing ch02s03.html for section
[java] Writing ch02s04.html for section
[java] Writing ch02s05.html for section
[java] Writing ch02.html for chapter
[java] Writing ch03s02.html for section
[java] Writing ch03s03.html for section
[java] Writing ch03s04.html for section
[java] Writing ch03s05.html for section
[java] Writing ch03s06.html for section
[java] Writing ch03s07.html for section
[java] Writing ch03s08.html for section
[java] Writing ch03s09.html for section
[java] Writing ch03.html for chapter
[java] Writing ch04.html for chapter
[java] Writing ch05.html for chapter
[java] Writing ch06s02.html for section
[java] Writing ch06s03.html for section
[java] Writing ch06s04.html for section
[java] Writing ch06s05.html for section
[java] Writing ch06.html for chapter
[java] Writing ch07s02.html for section
[java] Writing ch07s03.html for section
[java] Writing ch07.html for chapter
[java] Writing ch08.html for chapter
[java] Writing ch09.html for chapter
[java] Writing ch10s02.html for section
[java] Writing ch10.html for chapter
[java] Writing ch11.html for chapter
```

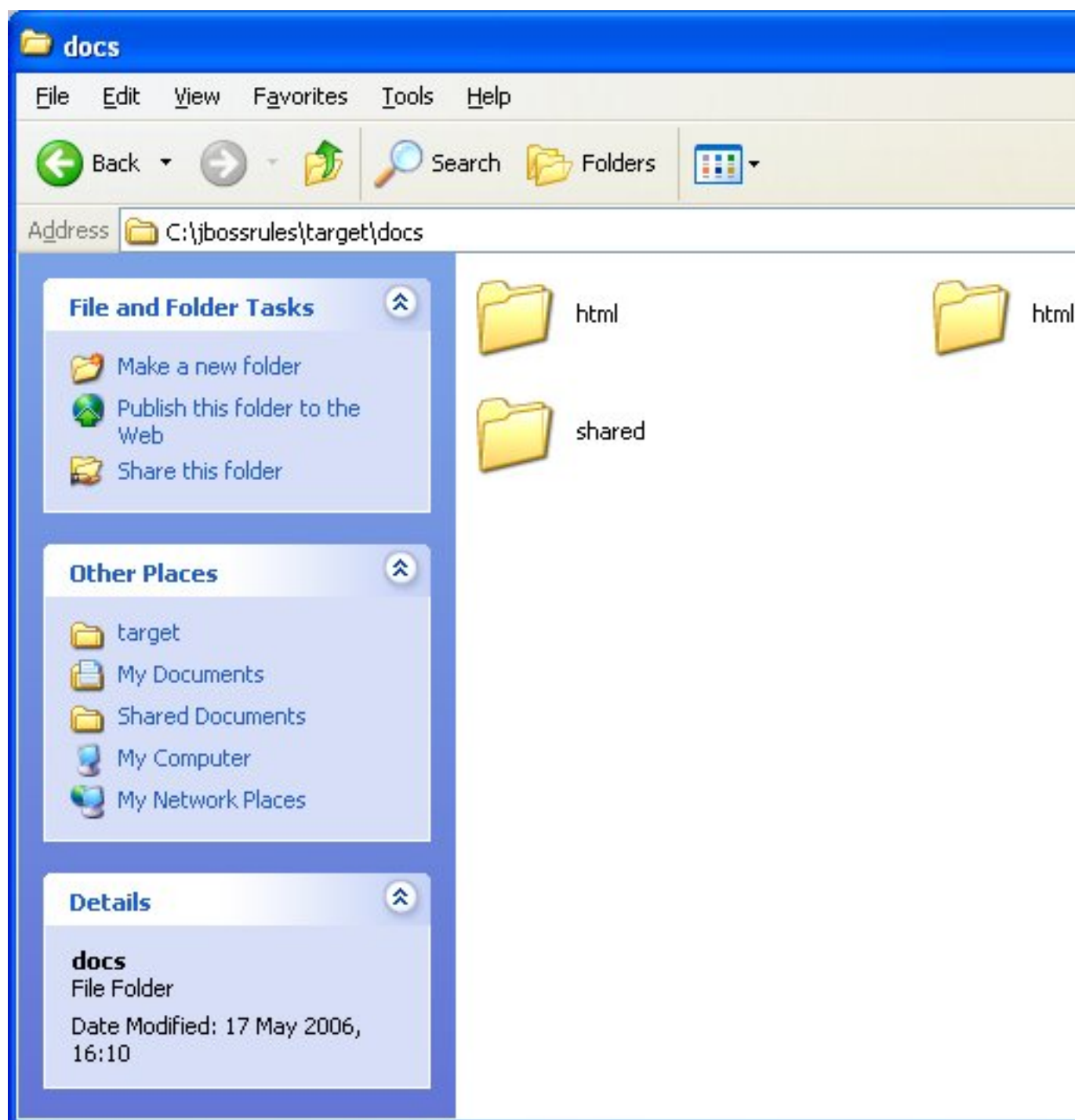


```
[java] Writing pt01.html for part
[java] Writing ix01.html for index
[java] Writing title.html for book

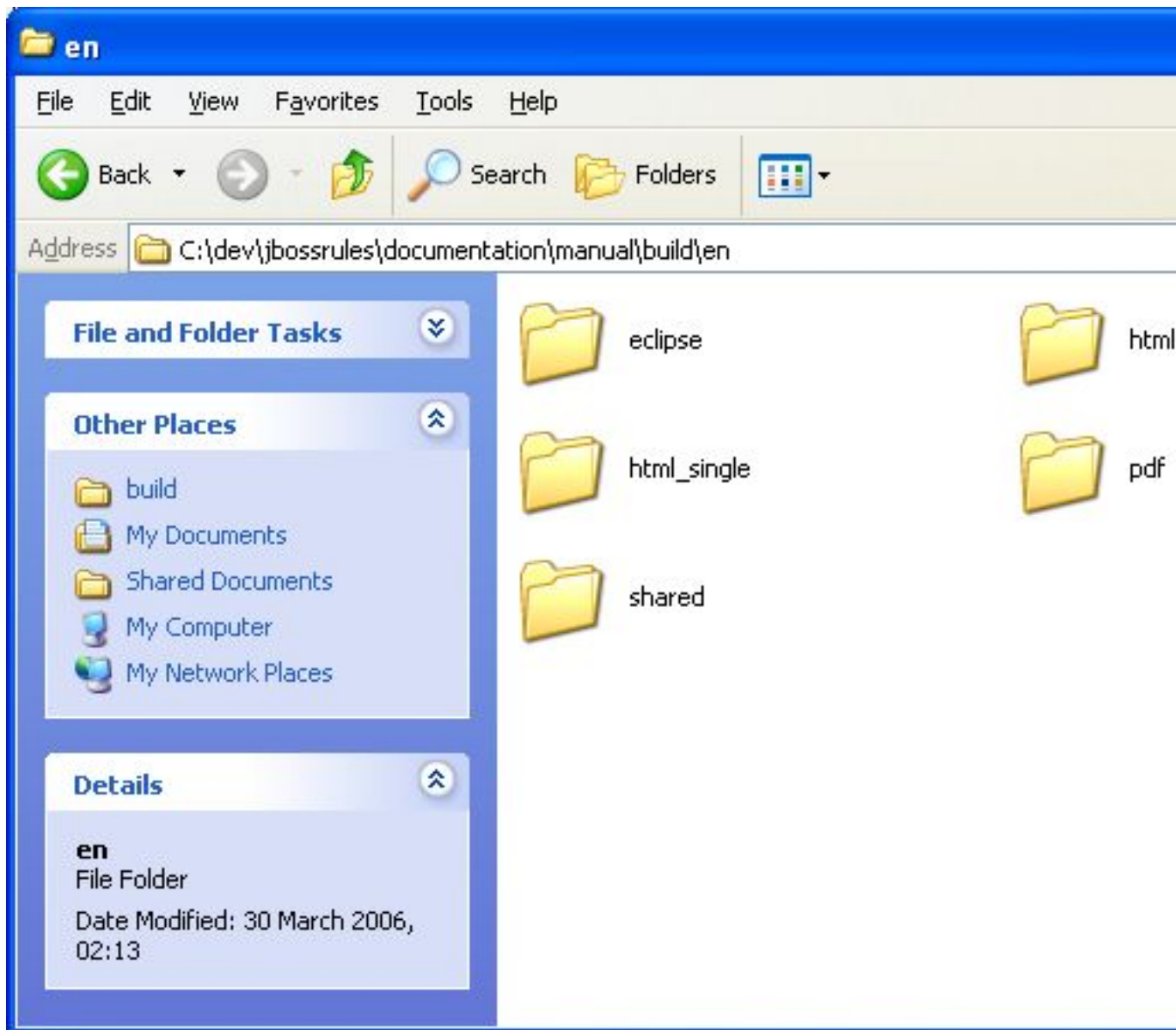
...snip ...

[INFO]
-----
[INFO] BUILD SUCCESSFUL
[INFO]
-----
[INFO] Total time: 51 seconds
[INFO] Finished at: Mon Jul 21 12:03:38 BRT 2007
[INFO] Final Memory: 5M/10M
[INFO]
----->
```

The generated manual can be found in the `target\drools-documentation$VERSION.jar` file, a compressed archive with all formats.



The manual was first generated into the manual's `build` directory, as shown below, before being copied across.



2.5. Eclipse

2.5.1. Generating Eclipse Projects

The Drools project has Eclipse projects checked in for convenience. However, these are originally generated by maven 2. If you have maven 2 installed, you can also regenerate the Eclipse projects automatically, or even generate it for IntelliJ etc, see the instructions below for this (most people can ignore this section)

Maven is able to generate standard Eclipse projects, but it is not able to generate Eclipse plug-in projects. To generate the Eclipse projects for drools-core, drools-compiler and drools-jsr94 type `'mvn Eclipse:Eclipse'`.

```
C:\dev\jbossrules>mvn eclipse:eclipse
[INFO] Scanning for projects...
[INFO] Reactor build order:
[INFO]   Drools
[INFO]   Drools :: Rete-00 Core
[INFO]   Drools :: Compiler
[INFO]   Drools :: JSR-94 API Module
[INFO] Searching repository for plugin with prefix: 'eclipse'.
[INFO] org.apache.maven.plugins: checking for updates from central
[INFO] org.codehaus.mojo: checking for updates from central
[INFO] artifact org.apache.maven.plugins:maven-eclipse-plugin:
tes from central
Downloading: http://repo1.maven.org/maven2/org/apache/maven/plu
e-plugin/2.1/maven-eclipse-plugin-2.1.pom
2K downloaded
Downloading: http://repo1.maven.org/maven2/org/apache/maven/plu
e-plugin/2.1/maven-eclipse-plugin-2.1.jar
40K downloaded
[INFO] -----
[INFO] Building Drools
[INFO]   task-segment: [eclipse:eclipse]
[INFO] -----
[INFO] Preparing eclipse:eclipse
[INFO] No goals needed for project - skipping
[INFO] [eclipse:eclipse]
[INFO] Not running eclipse plugin goal for pom project
[INFO] -----
[INFO] Building Drools :: Rete-00 Core
[INFO]   task-segment: [eclipse:eclipse]
[INFO] -----
[INFO] Preparing eclipse:eclipse
[INFO] [resources:resources]
[INFO] Using default encoding to copy filtered resources.
```

```

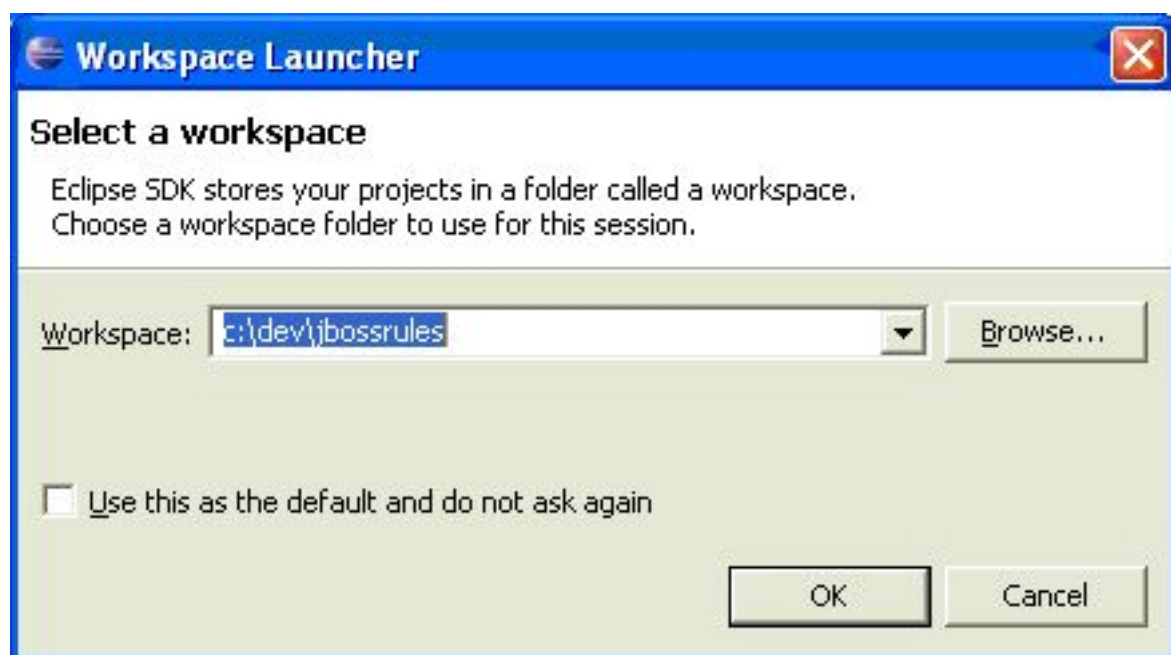
o antlr:stringtemplate:jar:2.3b6
o colt:colt:jar:1.2.0
o commons-lang:commons-lang:jar:2.1
o antlr:antlr:jar:2.7.6
o jsr94:jsr94:jar:1.1
o commons-collections:commons-collections:jar:3.1
o drools-asm:drools-asm:jar:2.2.1
o antlr:antlr3:jar:3.0ea8
o jsr94:jsr94-tck:jar:1.0.2
o xerces:xercesImpl:jar:2.6.2
o xml-apis:xml-apis:jar:1.0.b2
o commons-logging:commons-logging-api:jar:1.0.4
o eclipse:jdtcore:jar:3.2.0.v_642
o jci:jci:jar:SNAPSHOT-378493+patch
o jsr94:jsr94-sigtest:jar:1.1

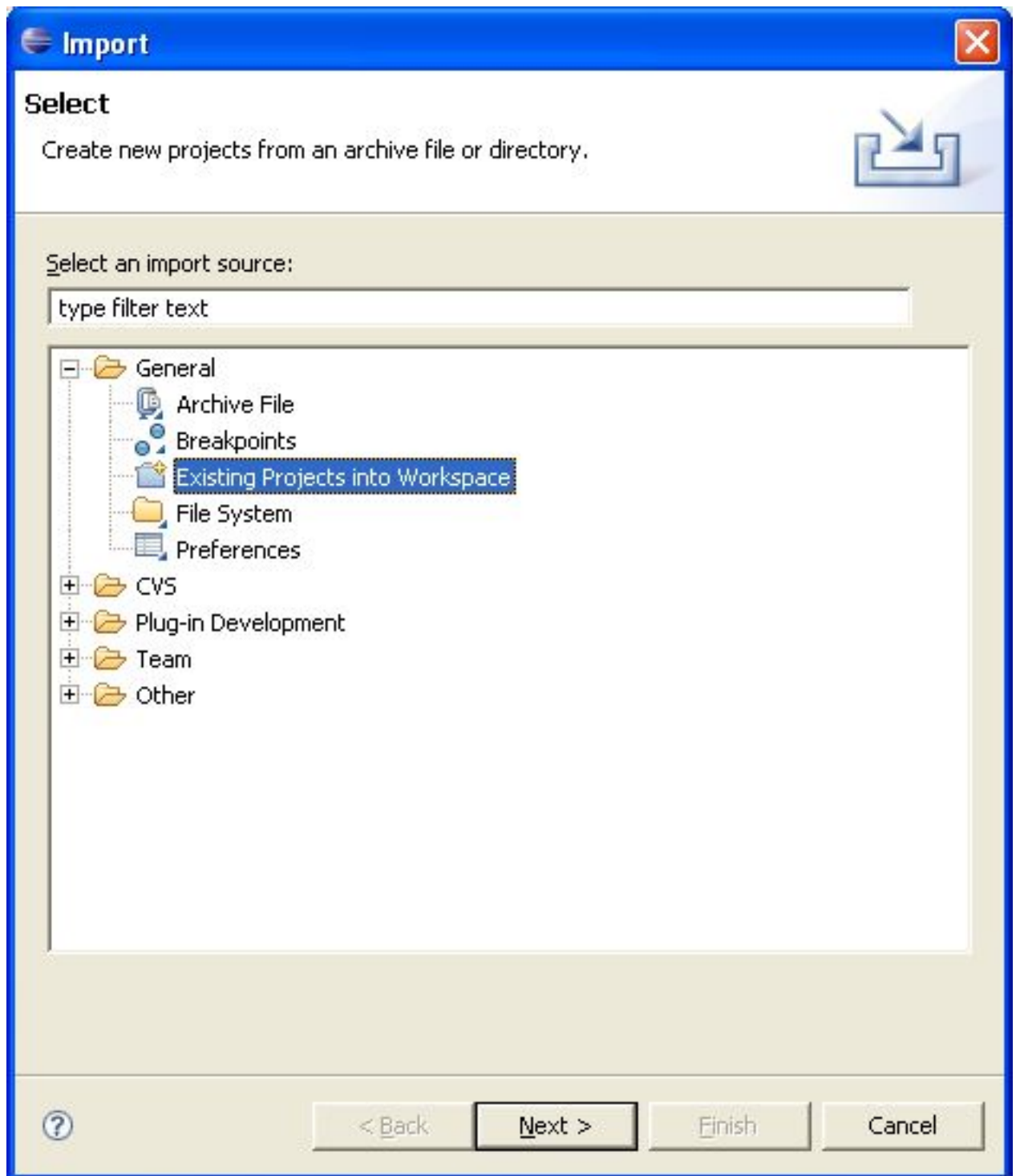
[INFO] Wrote Eclipse project for "drools-jsr94" to C:\dev\jboss
4.
[INFO]
[INFO]
[INFO] -----
[INFO] Reactor Summary:
[INFO] -----
[INFO] Drools .....
[INFO] Drools :: Rete-00 Core .....
[INFO] Drools :: Compiler .....
[INFO] Drools :: JSR-94 API Module .....
[INFO] -----
[INFO] BUILD SUCCESSFUL
[INFO] -----
[INFO] Total time: 7 seconds
[INFO] Finished at: Wed Mar 29 23:27:40 BST 2006
[INFO] Final Memory: 4M/8M
[INFO] -----
C:\dev\jbssrules>

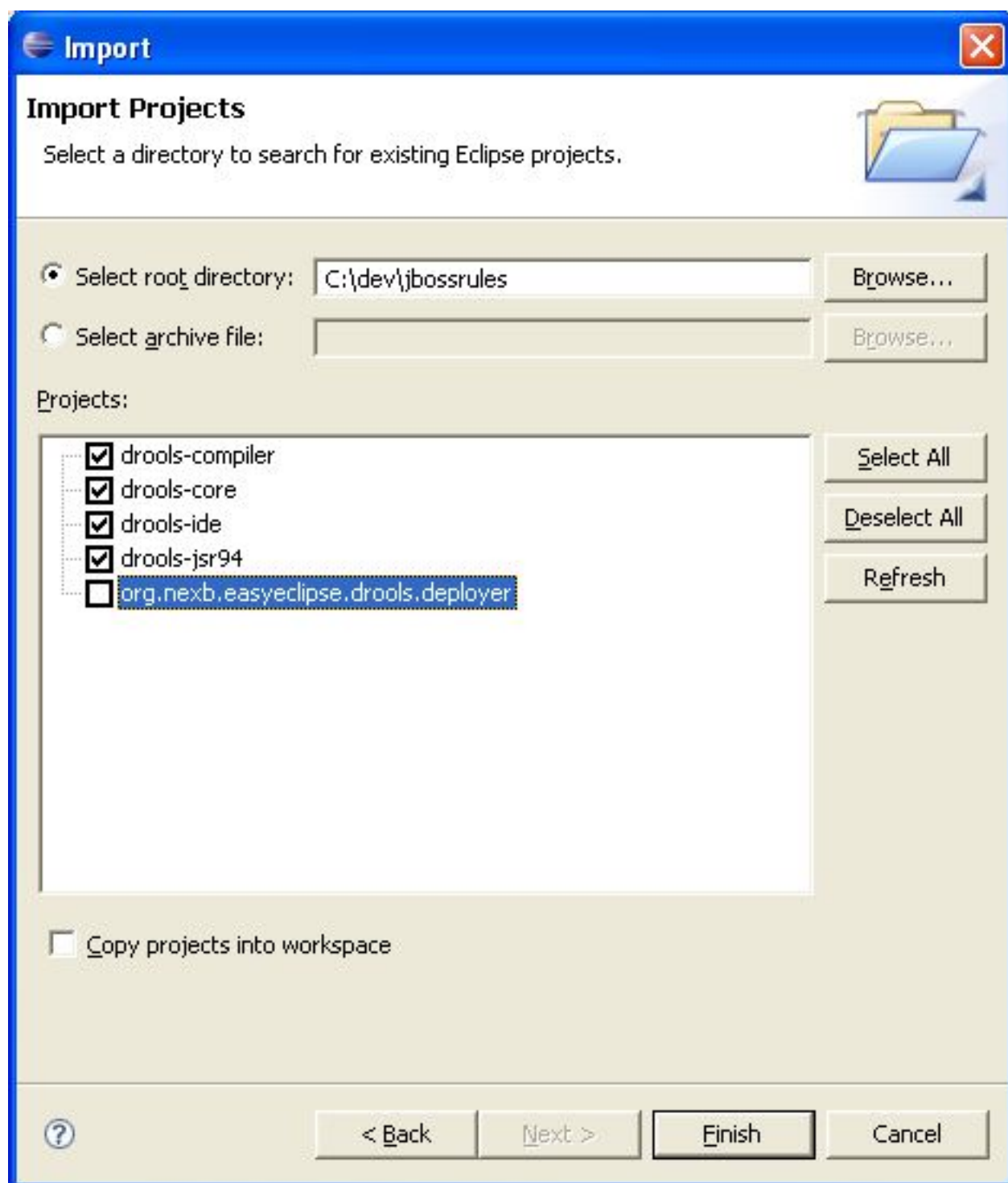
```

2.5.2. Importing Eclipse Projects

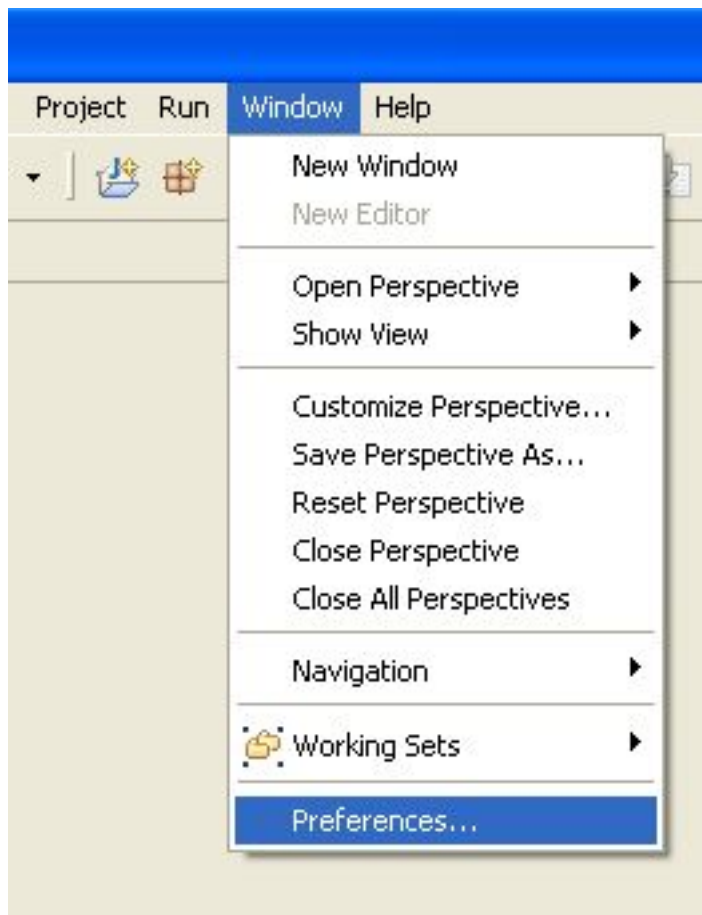
With the Eclipse project files generated they can now be imported into Eclipse. When starting Eclipse open the workspace in the root of your subversion checkout.

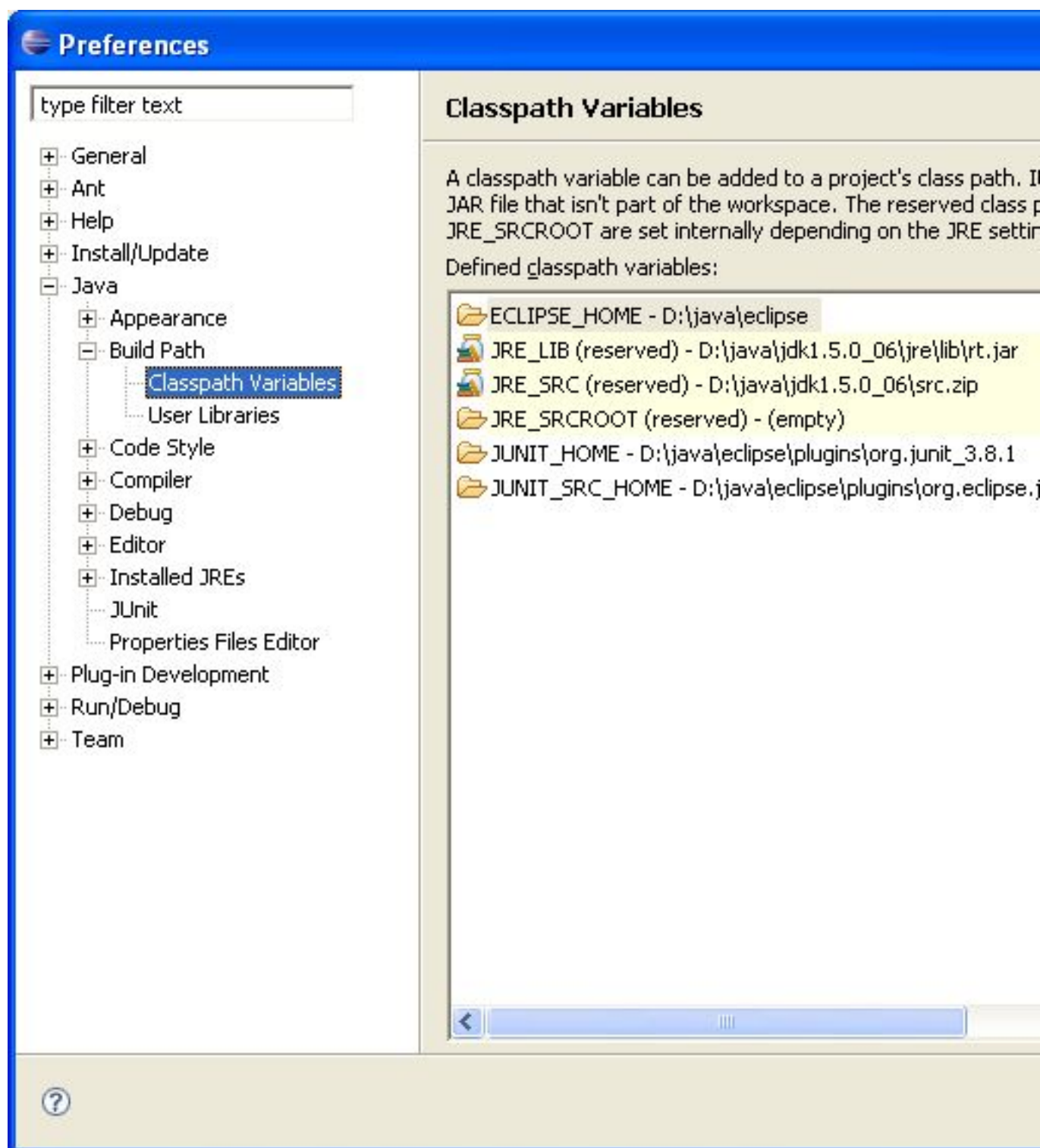


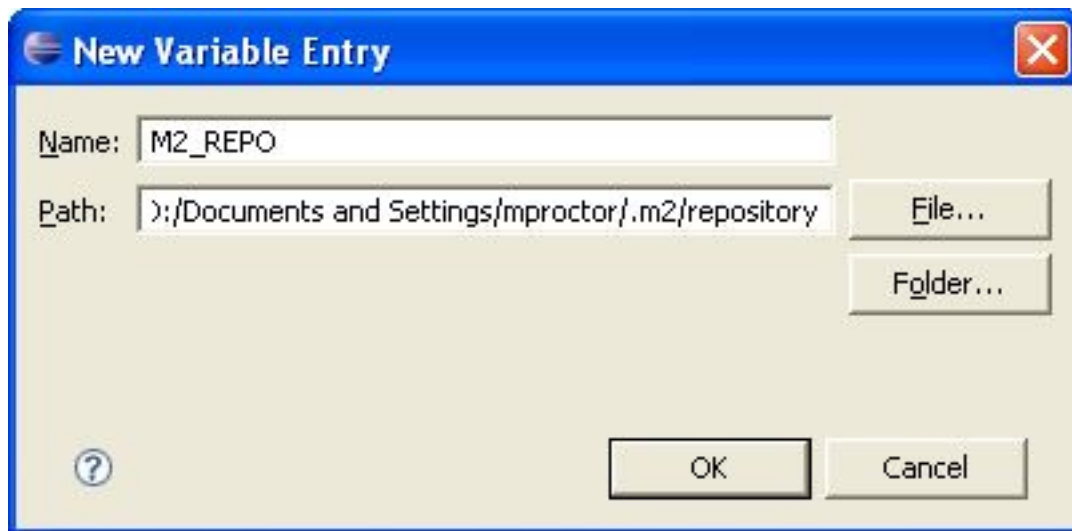


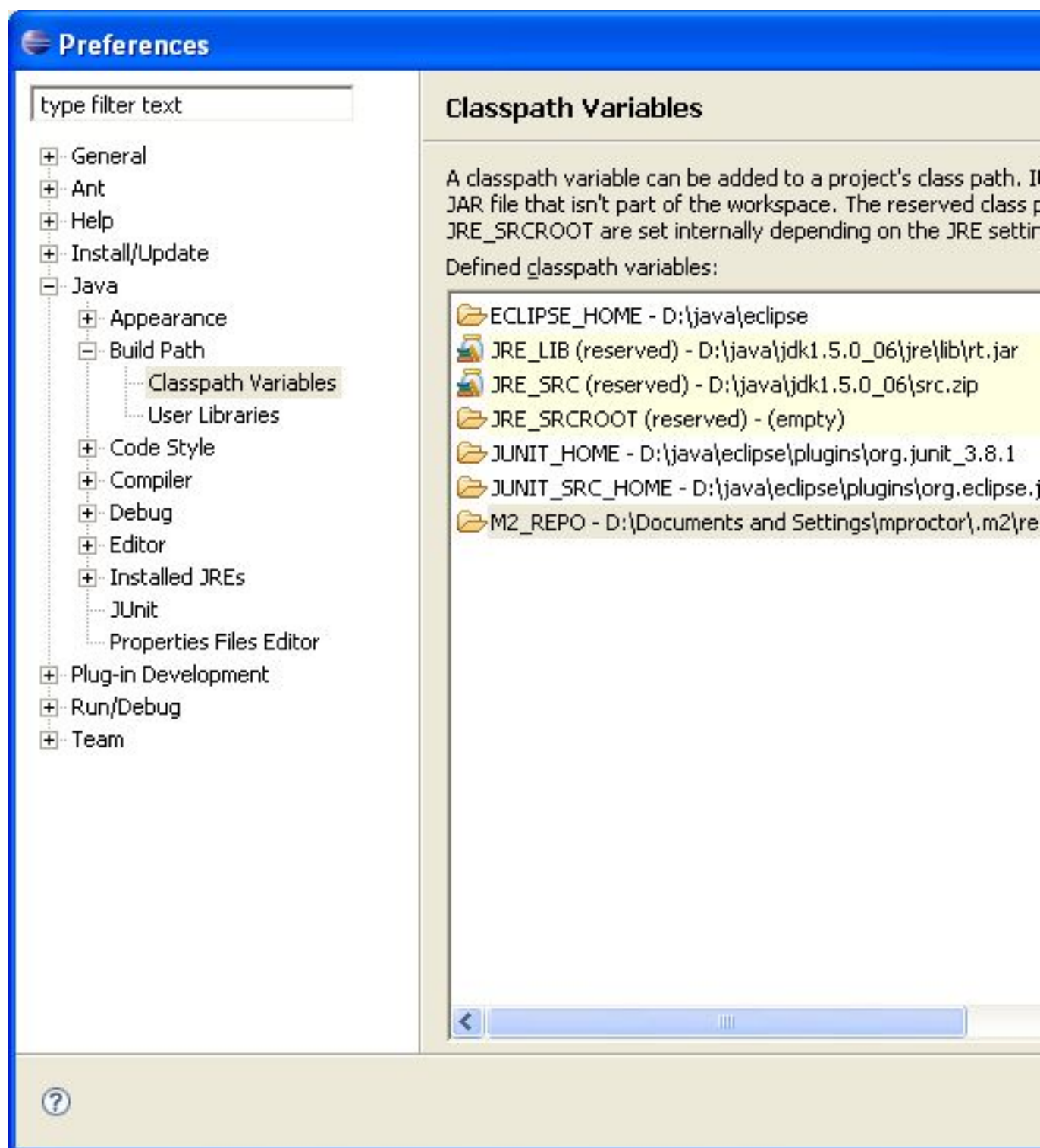


When calling 'mvn install' all the project dependencies were downloaded and added to the local Maven repository. Eclipse cannot find those dependencies unless you tell it where that repository is. To do this setup an M2_REPO classpath variable.



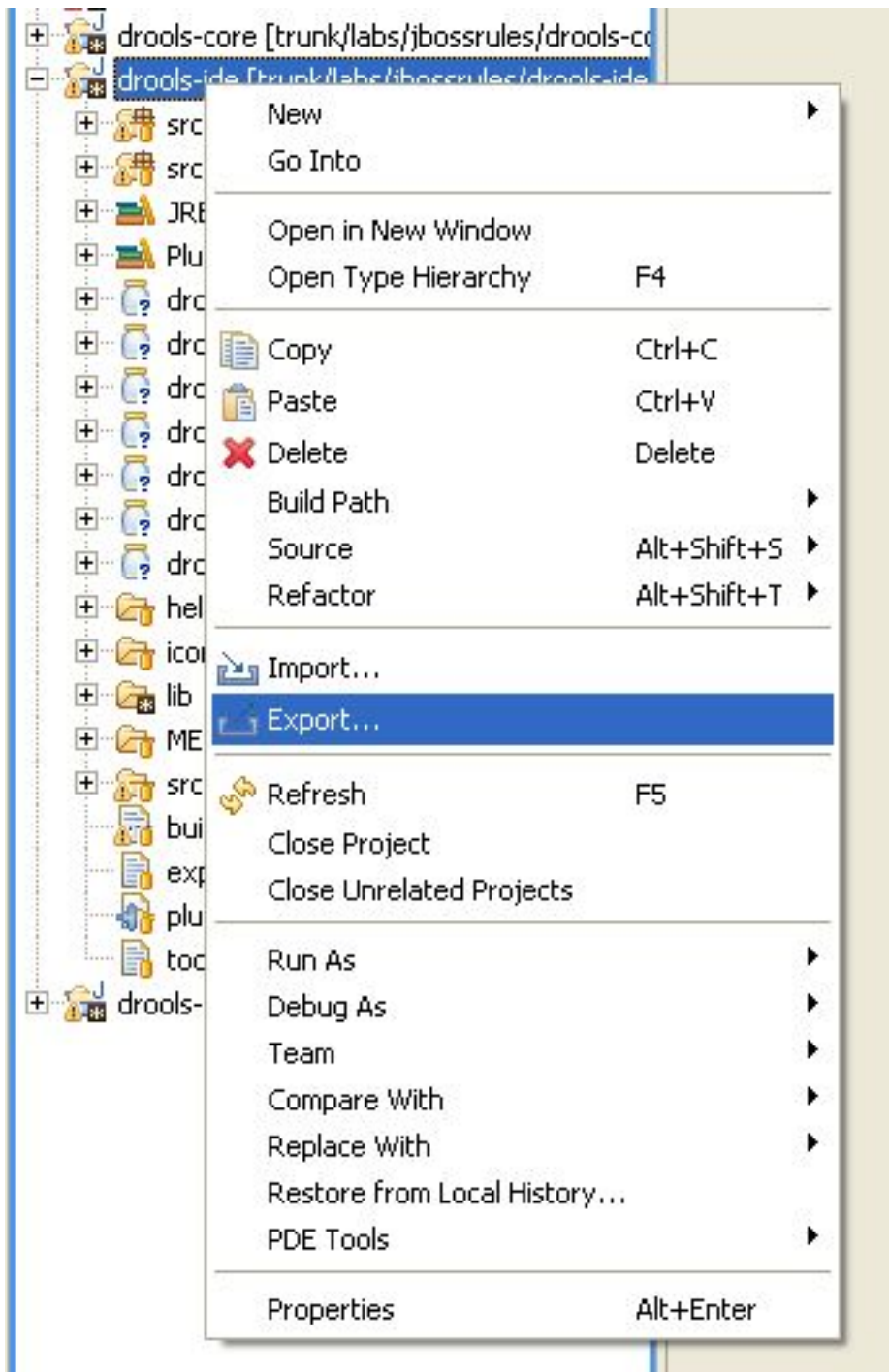


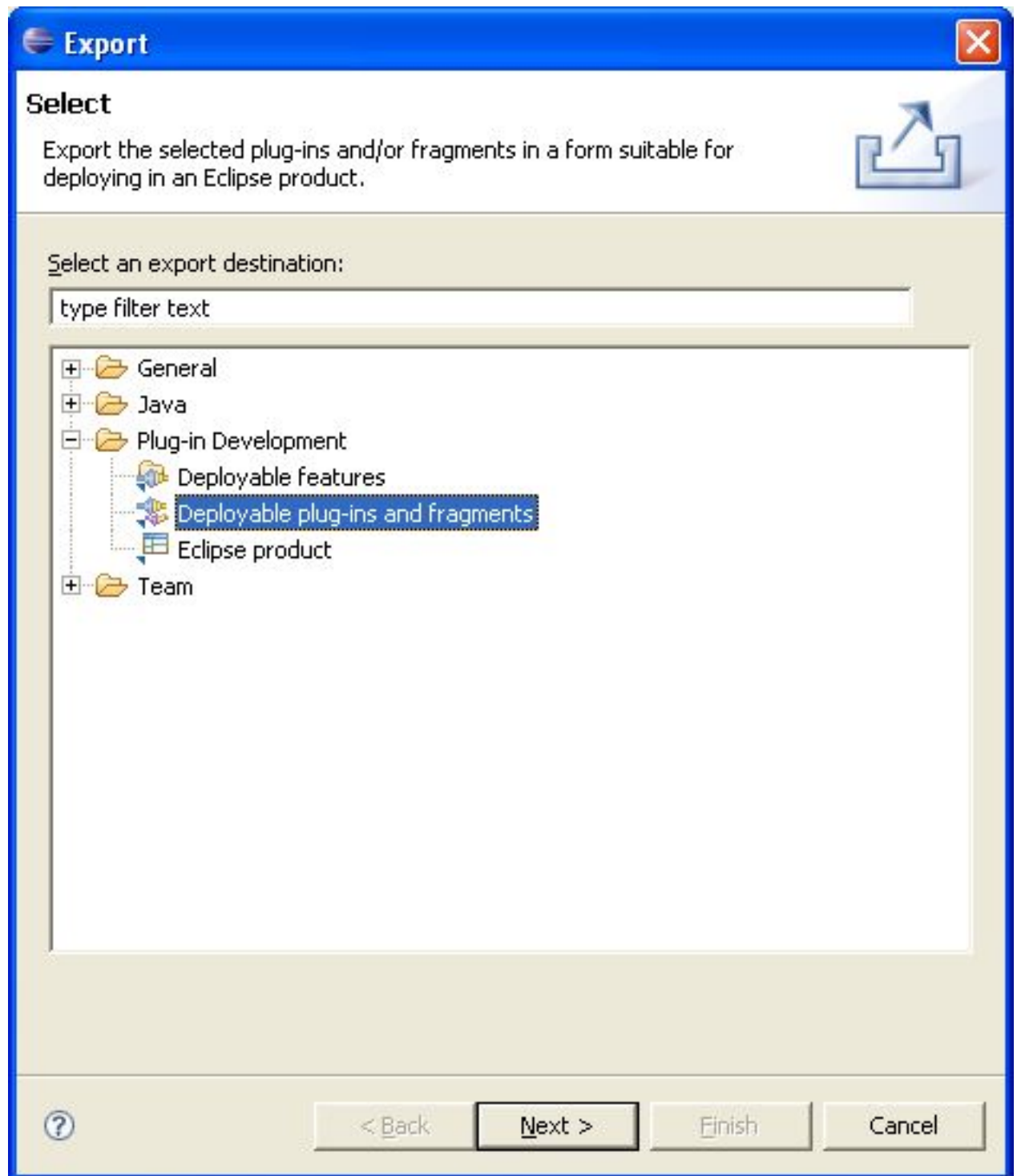


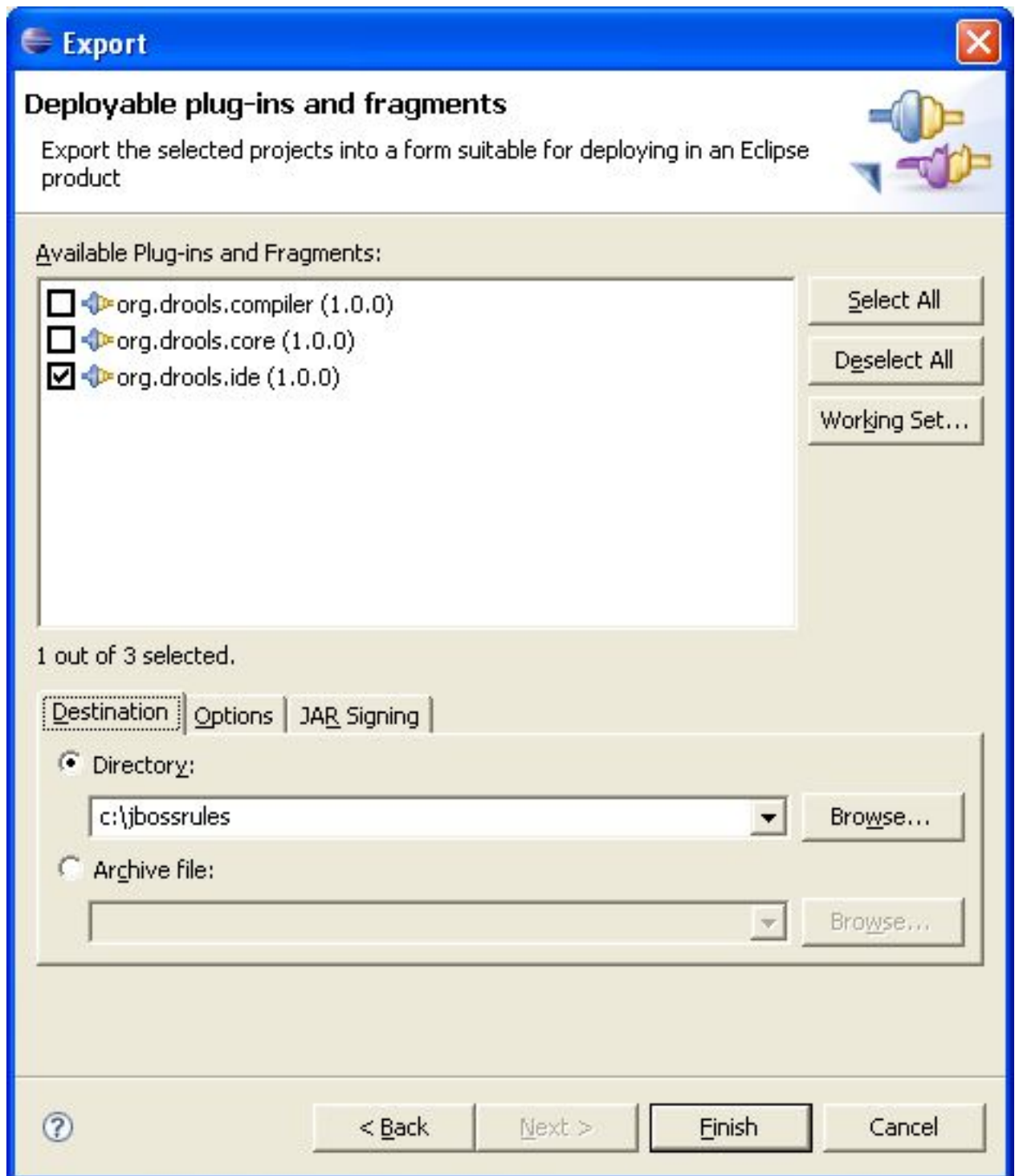


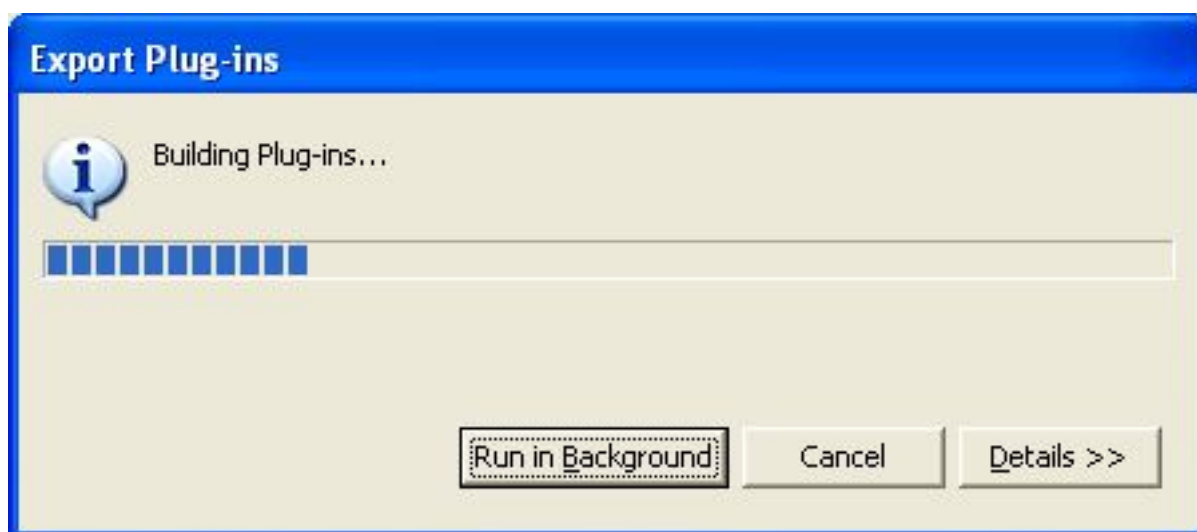
2.5.3. Exporting the IDE plug-in

The drools-ide project was checked out using subversion and is ready for exporting.

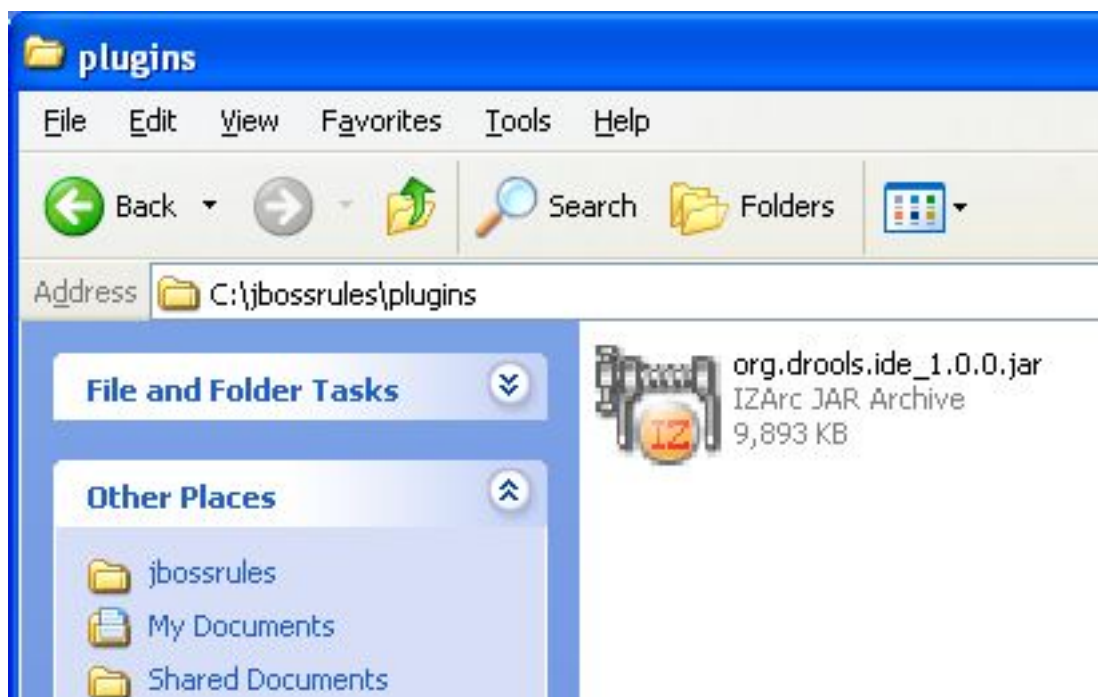


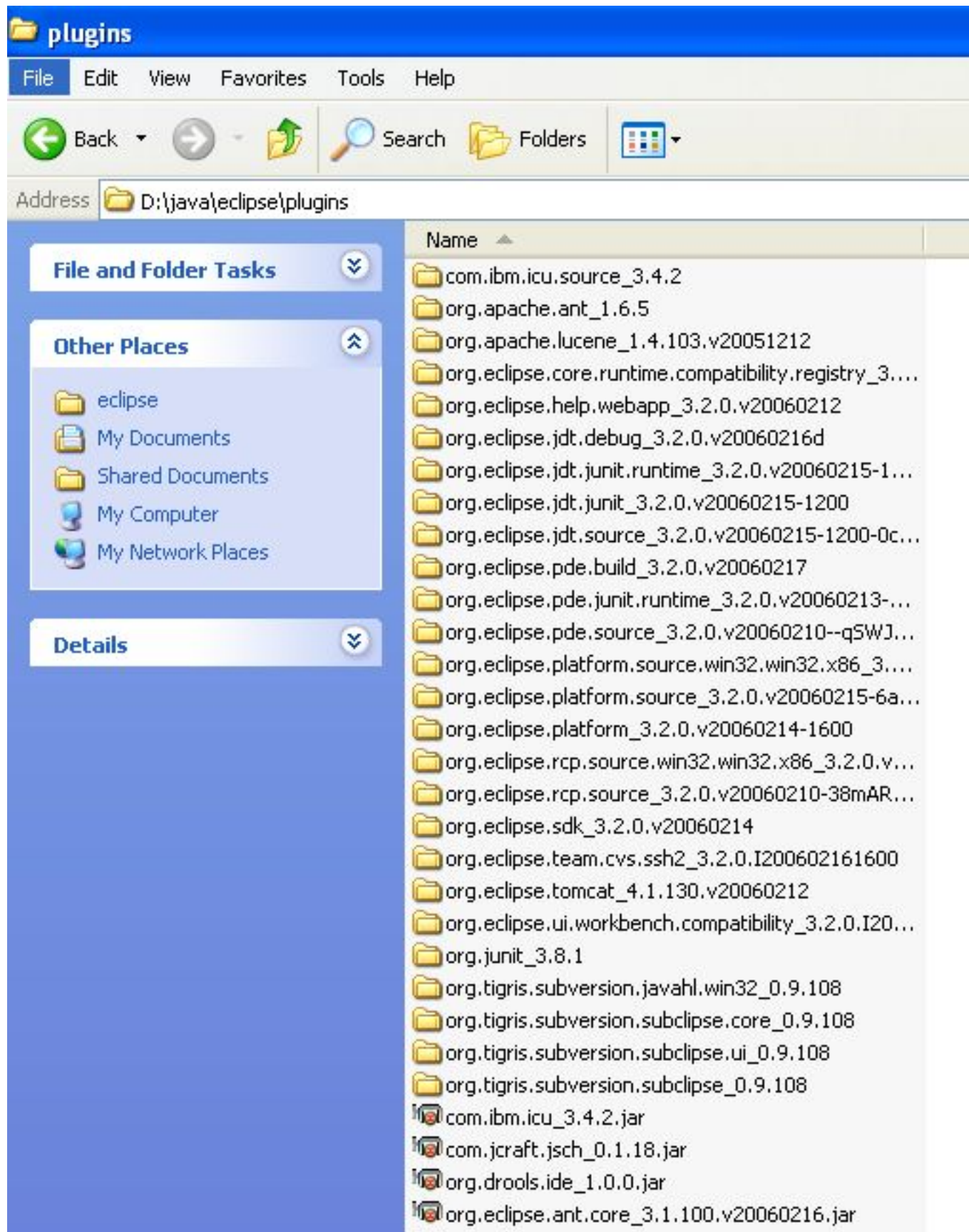




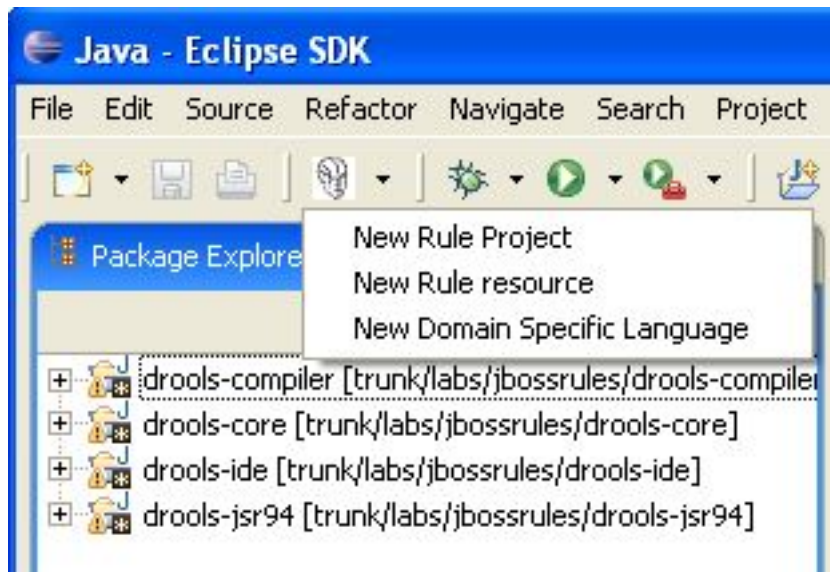


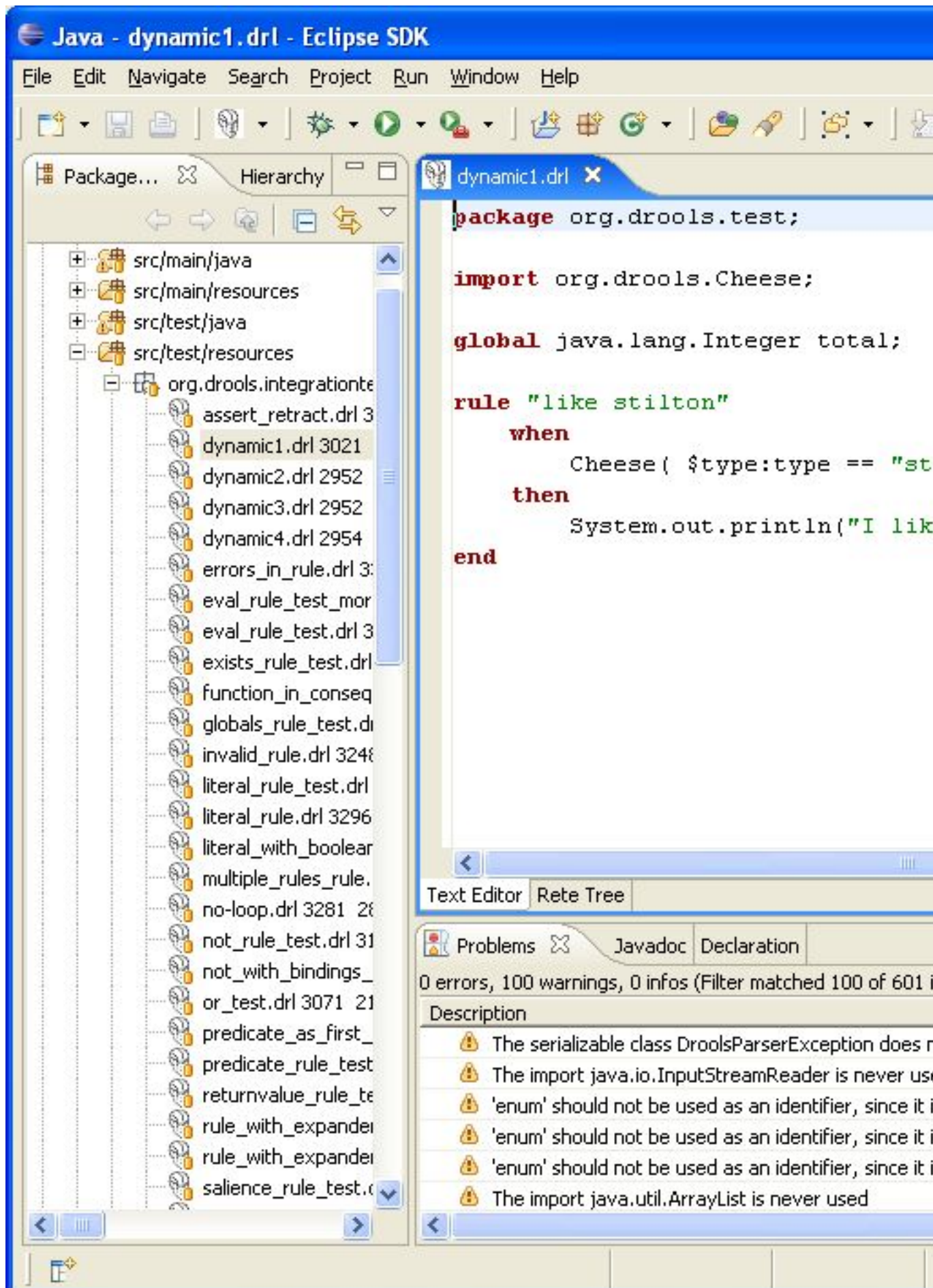
Once the plug-in has been built open the output directory and copy the jar to the Eclipse plug-in directory.





At this point if Eclipse is already open it will need to be restarted. At which point you show now see the new Drools menu icon and drl's should have icons and be provided with syntax highlighting and intellisense.





2.5.4. Building the update site

There is also an update site for the plug-in. For developers who want to update the update site (ha) you will need to get to the update site project (or create a new one). They are kept in SVN, but in /jbossrules/update instead of /trunk. They are plain vanilla Eclipse feature and site projects.

PLEASE REMEMBER that the plug-in in the downloads directory, as a zip, should also be updated at the same time as the update site (as they are alternative ways of getting the same plug-in).

Eclipse refreshing plug-ins in features and sites seems to not work, so what is best is to manually edit the site.xml project and the feature.xml. To do this, open the site.xml file in the drools-ide-update project, it should look something like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<site>
  <!-- change both the jar and the version number, make sure the new
  features jar is named
        the same as what you put in -->
  <feature url="features/org.drools.ide_1.0.2.jar" id="org.drools.ide"
  version="1.0.2">
    <category name="JBossRules"/>
  </feature>
  <category-def name="JBossRules" label="JBoss Rules"/>
</site>
```

Change the version attribute to be something new, and also the name of the feature jar to have a new version number at the end.

Go into the /feature directory, and unzip the feature jar to get to the feature.xml. (the feature jar really just contains the feature.xml). Open the feature.xml, and it should look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<feature
  id="org.drools.ide"
  label="Drools Rule Workbench"
  version="1.0.2"> <!-- UPDATE THIS !! -->

  <description>
    JBoss Rules (Drools) Workbench for developers.
  </description>

  <copyright>
    Copyright 2005 JBoss Inc
  </copyright>

  <license>
    Licensed under the Apache License, Version 2.0(the "License");
    you may not use this file except in compliance with the License.
    You may obtain a copy of the License at
```

```
http://www.apache.org/licenses/LICENSE-2.0
```

```
Unless required by applicable law or agreed to in writing, software  
distributed under the License is distributed on an &quot;AS IS&quot; BASIS,  
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.  
See the License for the specific language governing permissions and  
limitations under the License.
```

```
</license>
```

```
<plug-in  
id="org.drools.ide"  
download-size="0"  
install-size="0"  
version="1.0.0"/> <!-- THIS JUST HAS TO BE CONSISTENT WITH THE plug-in -->
```

```
</feature>
```

Change the version number in the FEATURE tag to be the same as what you referred to in the site.xml. If you changed the version number of the main plug-in, you will need to put the version number in the plug in tag (which refers to org.drools.ide plug-in). Then zip up the feature.xml into a jar with the same name as you referred to in the site.xml.

Finally, drop the plug-in jar into the /plugins jar directory of the update site (get the actual plug-in from the exported plug-in in the previous step). Now you can upload the site as is, and it will show up as a new version for Eclipse clients.

Index

A

ant, 19

D

docbook, 27

E

eclipse, 19

Eclipse, 31, 33

H

html, 27

M

maven, 19, 31

Maven 2, 24

P

path, 19

S

subversion, 19, 20

T

TortoiseSVN, 19

