Eclipse Java IDE - Tutorial

Lars Vogel

Version 1.6

30.12.2009

Eclipse Java IDE

This article describes the usage of Eclipse as a Java IDE. It describes the installation of Eclipse, the creation of Java programs, the usage of external jars, quick fix and content assist and the usage of the Eclipse update manager.

This article is based on Eclipse 3.5 (Eclipse Galileo).

Table of Contents

1. Eclipse Overview

Eclipse an open source community whose projects building tools and frameworks for creating general purpose application.

The most popular usage of Eclipse is as a Java development environment which will be described in this article.

2. Getting started

2.1. Installation

Download "Eclipse IDE for Java Developers" from the website Eclipse Downloads and unpack it to a directory. This is sufficient for Eclipse to be used; no additional installation procedure is required.

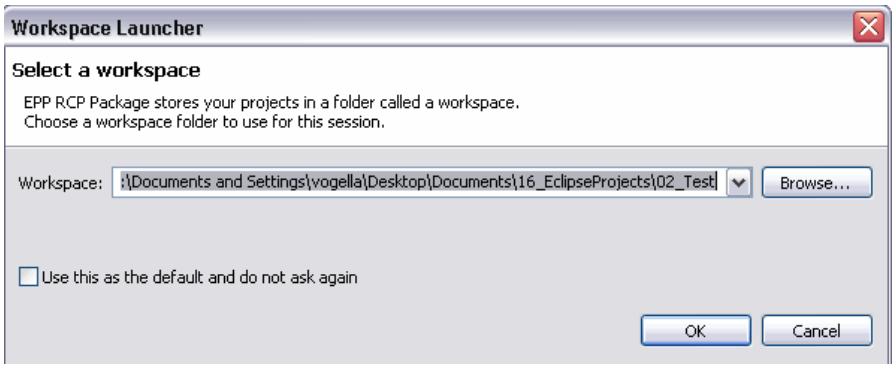

Use a directory path which does not contain spaces in its name.


Eclipse requires an installed Java Runtime. I recommended to use Java 6 (also known as Java 1.6).

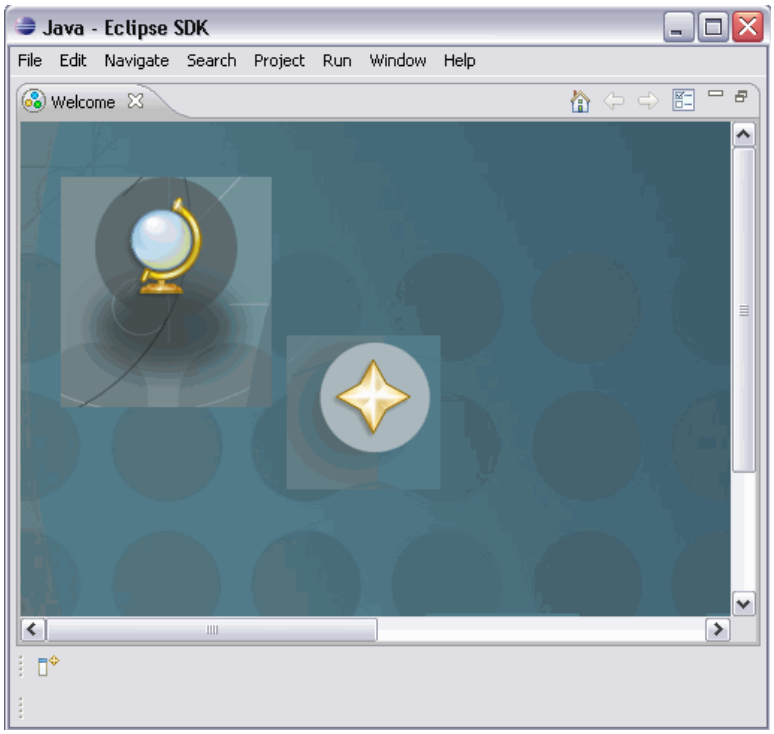
2.2. Start Eclipse

To start Eclipse double-click on the file eclipse.exe in your installation directory.

The system will prompt you for a workspace. The workspace is the place there you store your Java projects (more on workspaces later). Select a suitable (empty) directory and press Ok.



Eclipse will start and show the Welcome page.



Close the welcome page by press in little x besides the Welcome.

3. Eclipse UI Overview

Eclipse provides perspectives, views and editors. Views and editors are grouped into perspectives. All projects are located in a workspace.

3.1. Workspace

The workspace is the physical location (file path) you are working in. You can choose the workspace during startup of eclipse or via the menu (File-> Switch Workspace-> Others).

All your projects, sources files, images and other artifacts will be stored and saved in your workspace.

To predefine the workspace you can use the startup parameter -data path_to_workspace, e.g. c:\eclipse.exe -data "c:\temp" Please note that you have to put the path name into brackets.

To see the current workspace directory in the title of Eclipse use -showLocation as additional parameter.

3.2. Perspective

A perspective is a visual container for a set of views and editors.

You can change the layout within a perspective (close / open views, editors, change the size, change the position, etc.)

A common problem is that you closed a view and don't know how to re-open this view. You can reset a perpective it to it original state via the menu "Window" -> "Reset Perspective".

Eclipse allow you to switch to another perspective via the menu Window->Open Perspective -> Other.

For Java development you usually use the "Java Perspective".

3.3. Views and Editors

A view is typically used to navigate a hierarchy of information or to open an editor. Changes in a view are directly applied.

Editors are used to modify elements. Editors can have code completion, undo / redo, etc. To apply the changes in an editor to the underlying resources, e.g. Java source file, you usually have to save.

4. Create your first Java program

The following will describe how to create a minimal Java program using Eclipse. It will be the classical "Hello World" program. Our program will write "Hello Eclipse!" to the console.

4.1. Create project

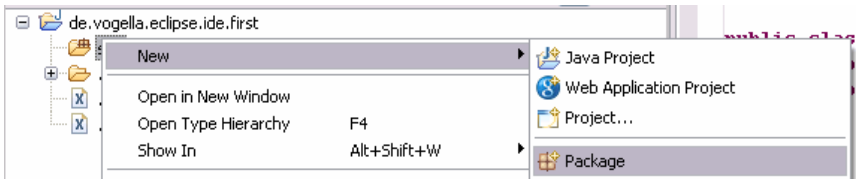
Select from the menu File -> New-> Java project. Maintain "de.vogella.eclipse.ide.first" as the project name. Select "Create separate source and output folders".

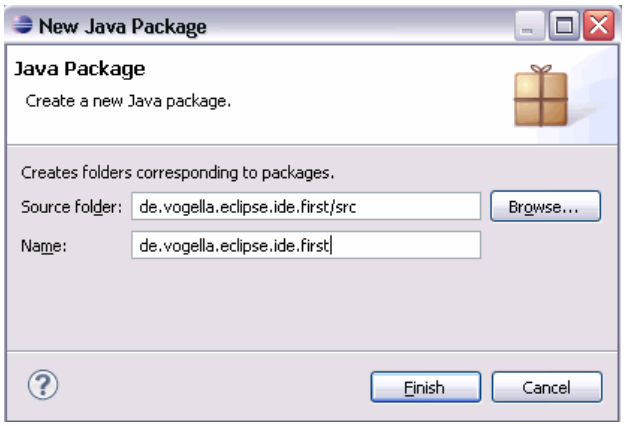
Press finish to create the project. A new project is created and displayed as a folder. Open the folder "de.vogella.eclipse.ide.first"

4.2. Create package

Create now a package. A good convention is to use the same name for the top package as the project. Create therefore the package "de.vogella.eclipse.ide.first".

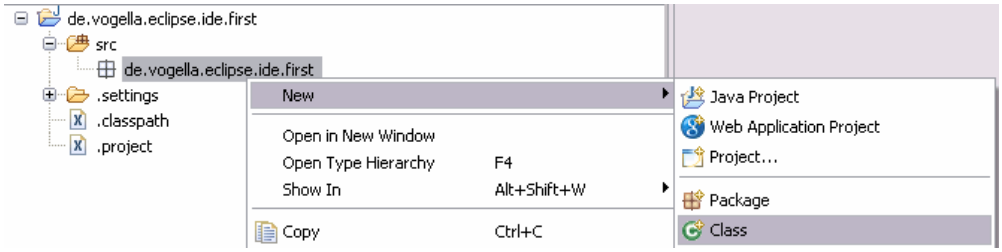
Select the folder src, right mouse click on it and select New -> Package.



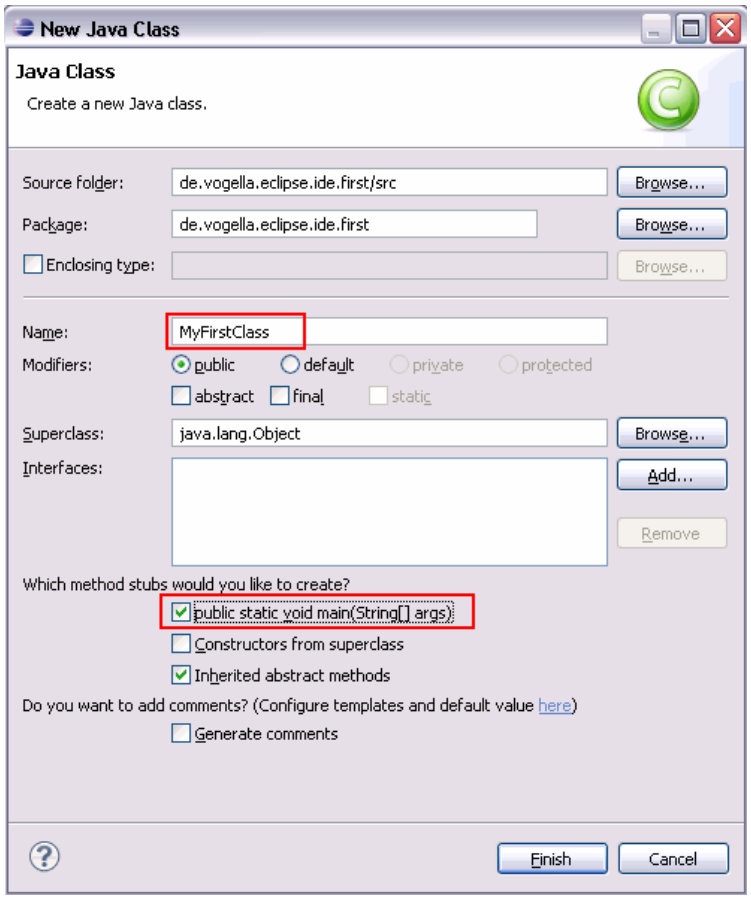


4.3. Create Java class

Right click on your package and select New -> Class



Create MyFirstClass, select the flag "public static void main (String[] args)"



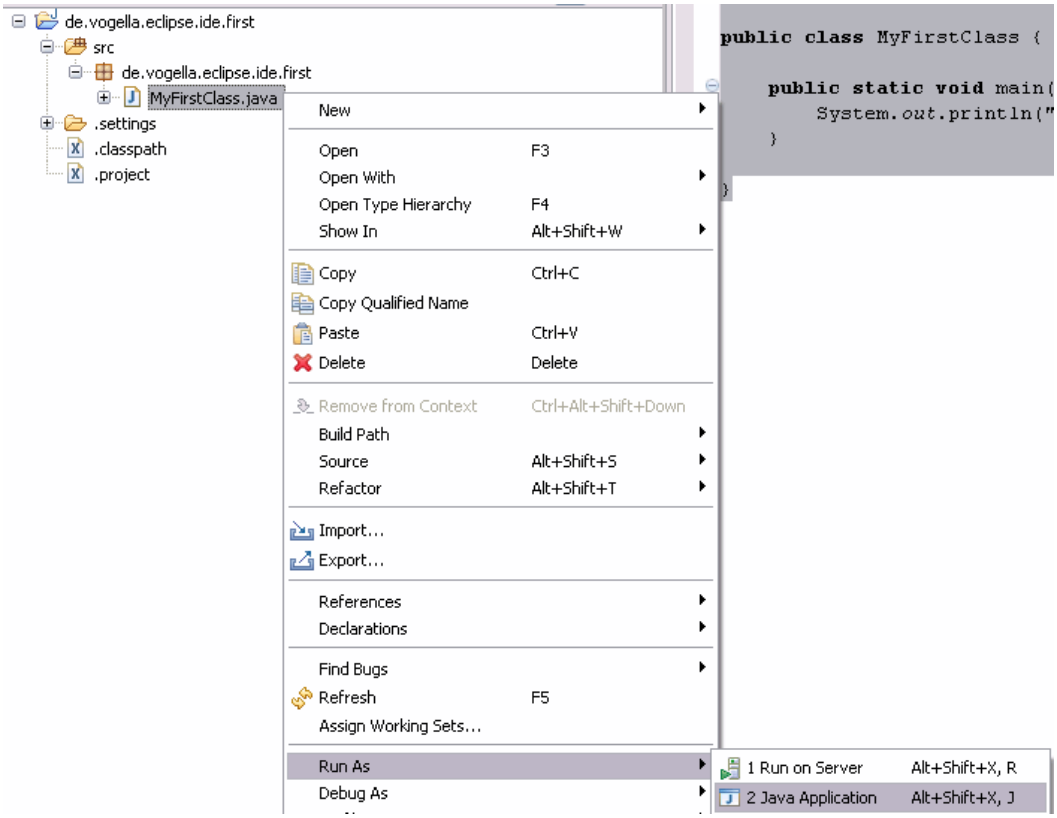
Maintain the following code.

```
package de.vogella.eclipse.ide.first;
public class MyFirstClass {
       public static void main(String[] args) {
```

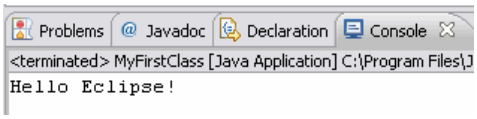
```
        System.out.println("Hello Eclipse!");
    }
}
```

4.4. Run your project in Eclipse

Now run your code. Right click on your Java class and select Run-as-> Java application

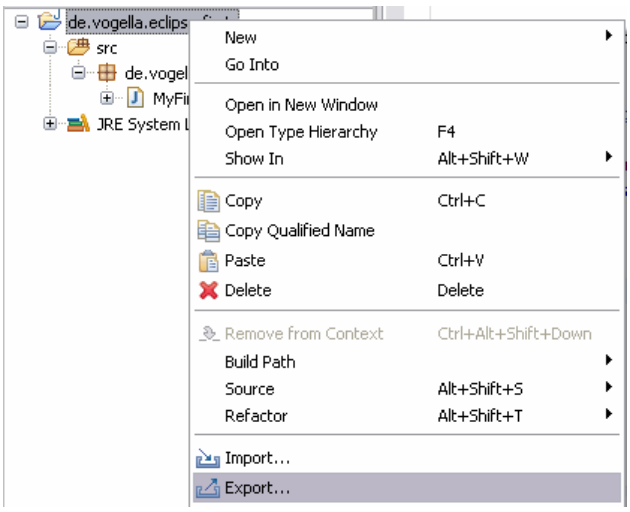


Finished! You should see the output in the console.

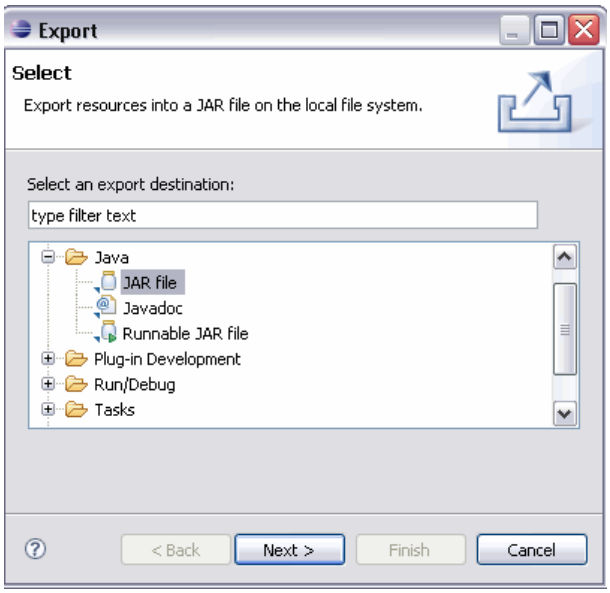


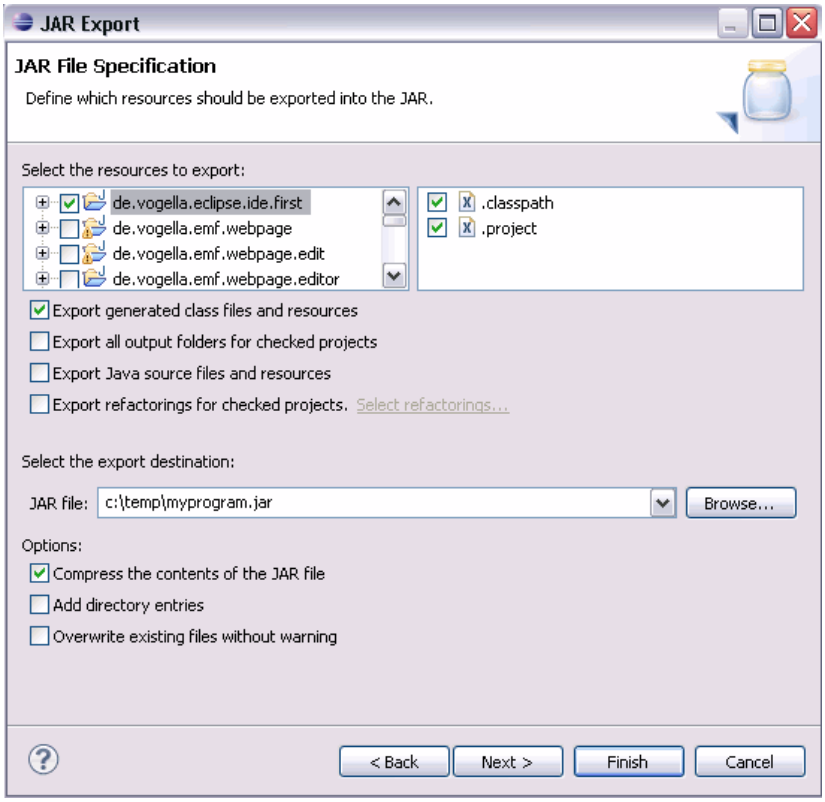
4.5. Run your Java program outside Eclipse (create jar file)

To run your Java program outside of Eclipse you need to export it as a jar file. Select your project, right click on it and select "Export".



Select JAR file, select next. Select your project and maintain the export destination and a name for the jar file. I named it "myprogram.jar".





Press finish. This will create a jar file in your select output directory.

4.6. Run your program outside Eclipse

Open a command shell, e.g. under Microsoft Windows select Start -> Run and type in cmd. This should open a consle.

Switch to your output directory, e.g. by typing cd path, e.g. if you jar is located in "c:\temp" type "cd c:\temp".

To run this program you need to include the jar file into your classpath. See Classpath and Java JAR Files for details.

```
java -classpath myprogram.jar de.vogella.eclipse.ide.first.MyFirstClass
```

```
C:\temp>java -classpath myprogram.jar de.vogella.eclipse.ide.first.MyFirstClass
Hello Eclipse!
```

Congratulations! You created your first Java project, a package a tiny Java program and you ran this program inside Eclipse and outside

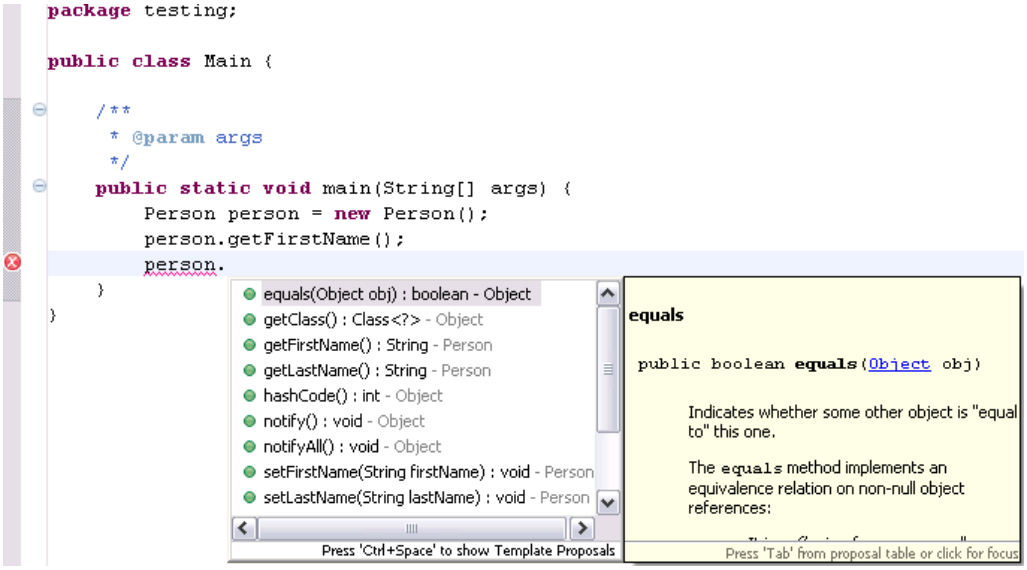
5. Content Assists and Quick Fix

For a list of the most important Eclipse shortcuts please see Eclipse Shortcuts

5.1. Content assist

The content assistant allows you to get input help in an editor. It can be invoked by CTRL + Space.

For example type syso and then press [Ctrl + Space] and it will be replaced by System.out.println(""). Or if you have an object, e.g. Person P and need to see the methods of this object you can type p. (or press CTRL + Space) which activates also the content assist.

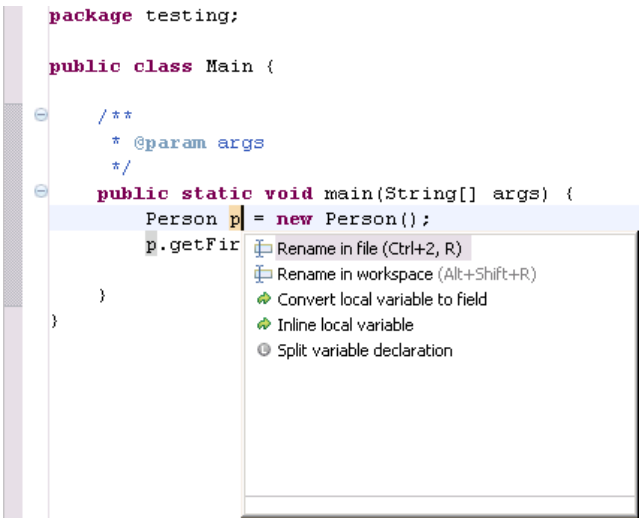


5.2. Quick Fix

Whenever there is a problem Eclipse will underline the problematic place in the coding. Select this and press (Ctrl+1)

For example type "myBoolean = true;" If myBoolean is not yet defined, Eclipse will highlight it as an error. Select the variable and press "Ctrn+1", then Eclipse will suggest to create a field or local variable.

Quick Fix is extremely powerful, it allows you to create new local / field variables, new methods, classes, put try and catch around your exceptions, assign a statement to a variable etc.



6. Using jars (libraries)

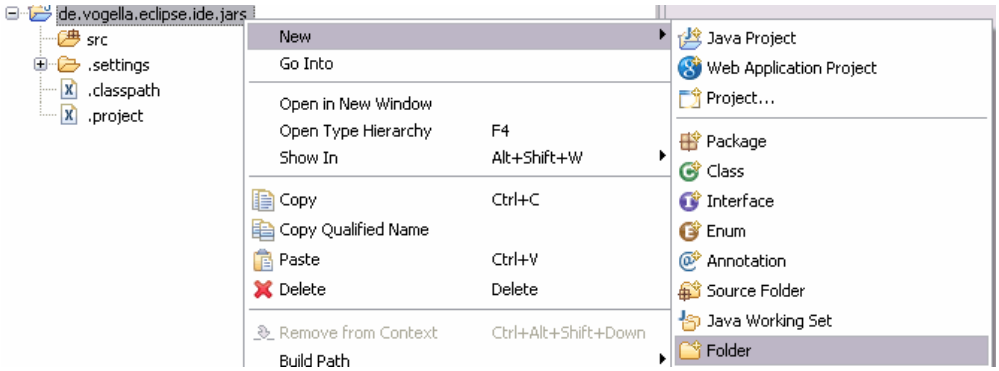
6.1. Adding external library (.jar) to the Java classpath

The following describes how to add external jars to your project.

The following assumes you have a jar available.

If you need an example for working with jars you can use JFreeChart Tutorial

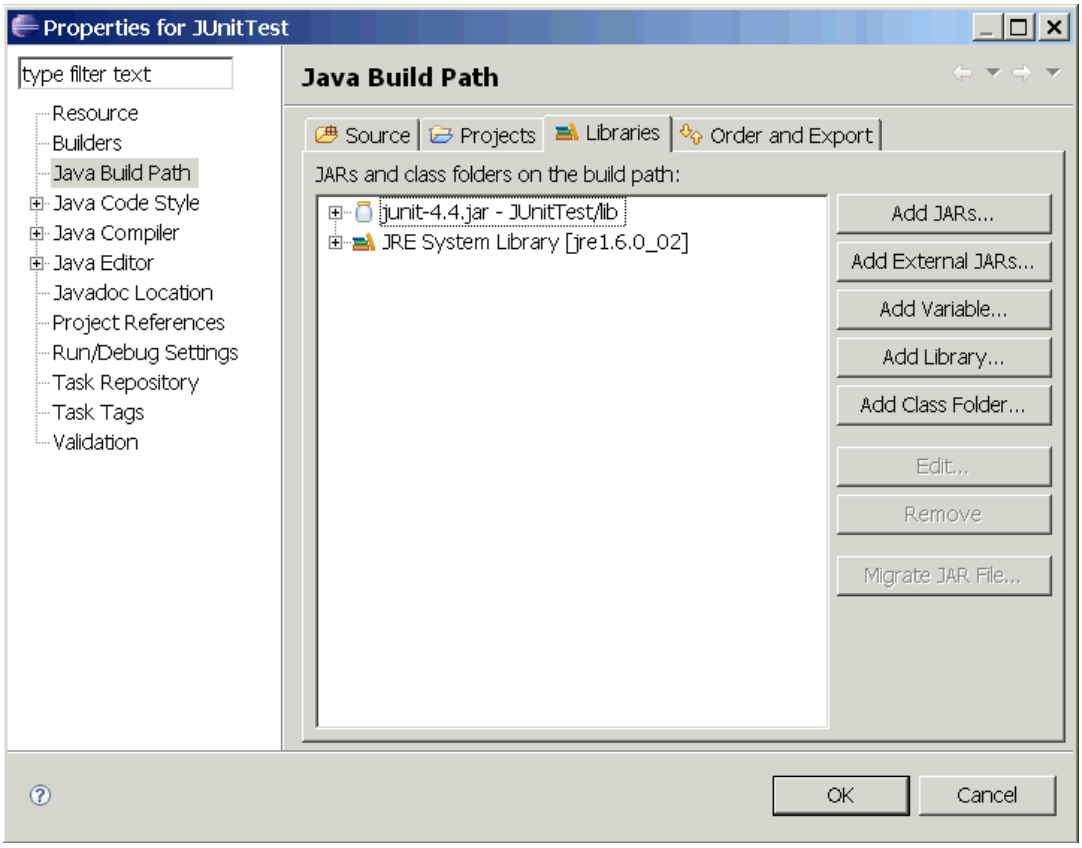
Create a new Java project "de.vogella.eclipse.ide.jars". Create a new folder called "lib" (or use your existing folder) by right click on your project and selecting New -> Folder



From the menu select File -> Import -> File system. Select your jar and select the folder lib as target.

Select your project, right mouse click and select properties. Under libraries select "Add JARs".

The following example shows how the result would look like if junit-4.4.jar would be added to a project.



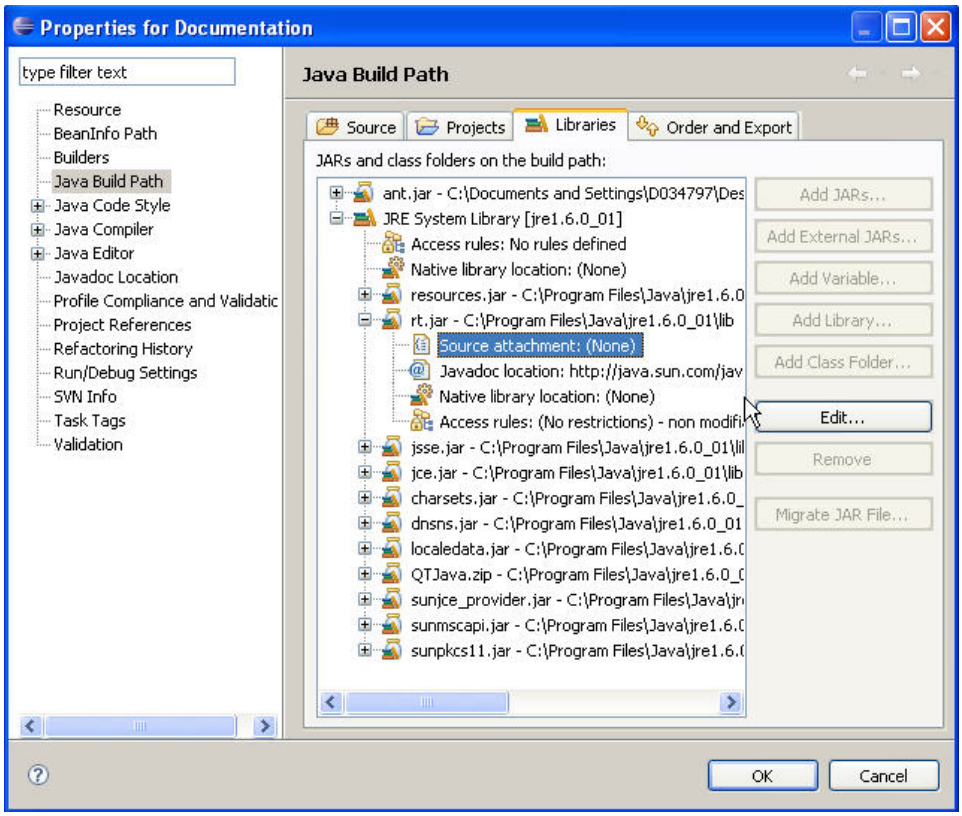
6.2. Show source code for jar

To browse the source of a type contained in library you can attach a source archive or source folder to this library. The editor will then show the source instead of a the decompiled code. Setting the source attachment also allows source level stepping with the debugger.

The Source Attachment dialog can be reached via:

Open the Java Build Path page of a project (Projects > Properties > Java Build Path). On the Libraries page expand the library's node and select the Source attachment attribute and press Edit

Maintain the location to the source attachement.



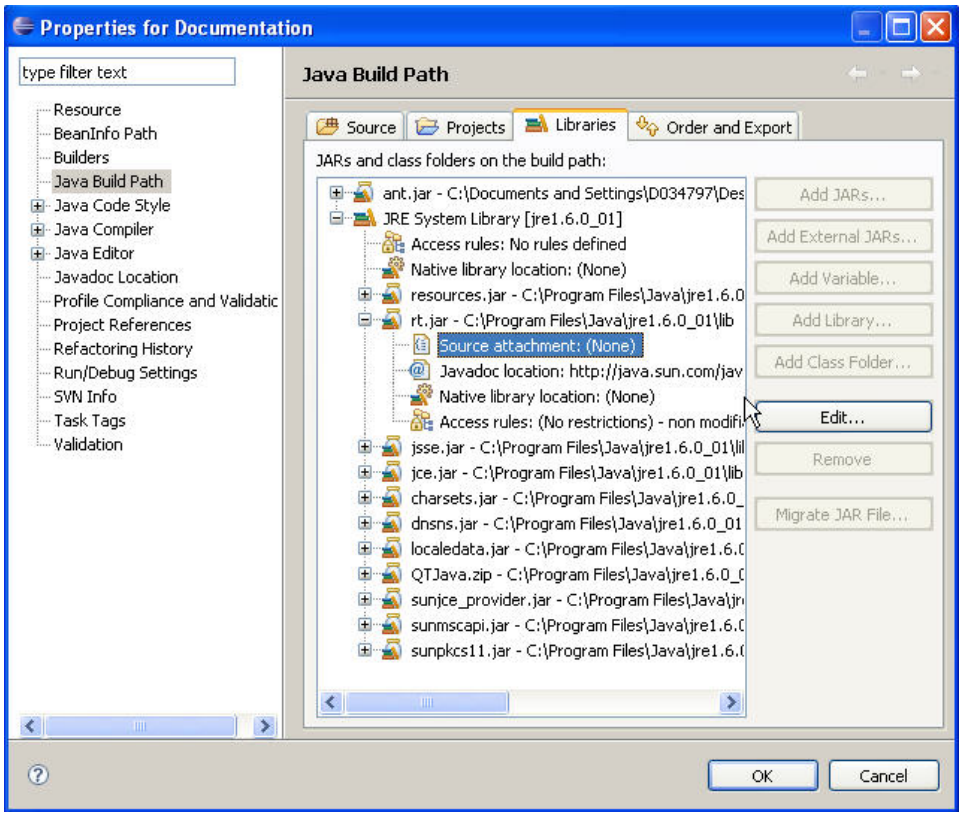
In the Location path field, enter the path of an archive or a folder containing the source.

6.3. Add the Javadoc for a jar

Download the javadoc of the jar and put it somewhere in your filesystem.

Open the Java Build Path page of a project (Projects > Properties > Java Build Path). On the Libraries page expand the library's node and select the Javadoc location attribute and press Edit

Maintain the location to the api.



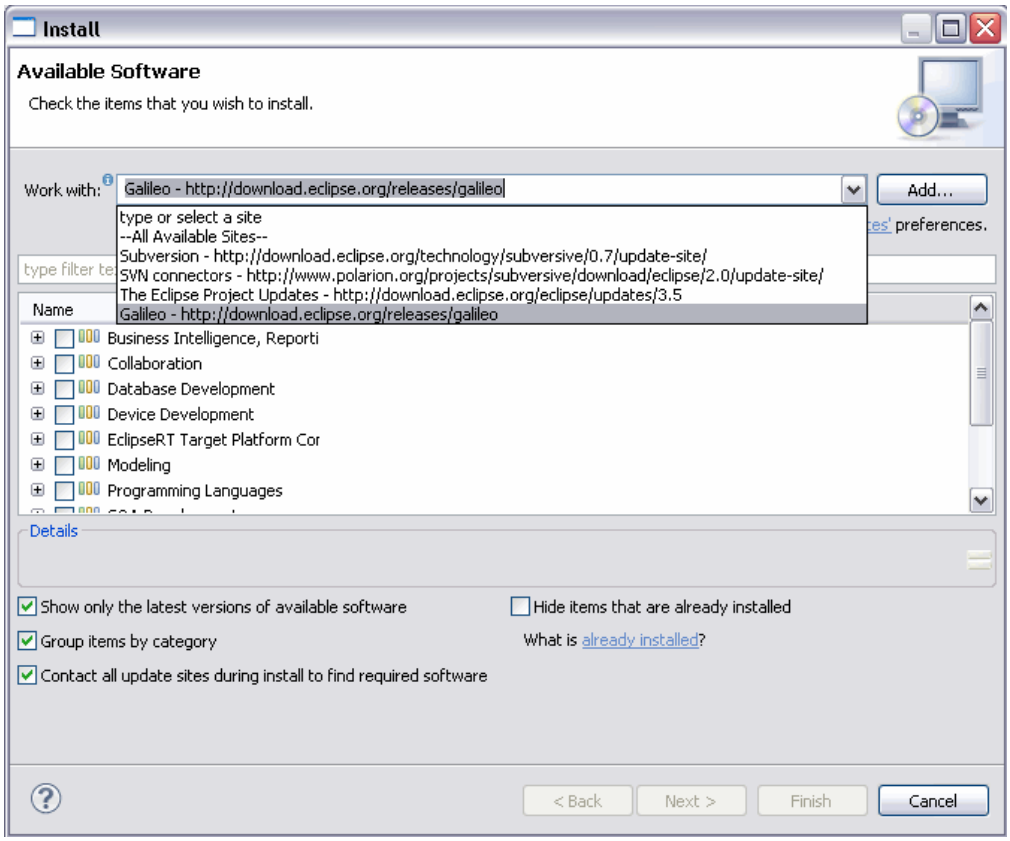
7. Updates and Installation of Plugins

7.1. Eclipse Update Manager

Eclipse provides functionality via so-called features (which contain plugins). Eclipse 3.5 contains a Software Update Manager which allows you to update existing plugins and to install new plugins.

To update your existing installation select the menu Help -> Check for Updates. The system will verify if for the installed plugins updates are available or not.

To install new functionality, select Help-> Install New Software.



Select from the list a update site from which you would like to install new software. For example if you want to install new plugins from Galileo select the Galileo Update Site.

Sometimes you have to uncheck "Group items by category" – not all available Plugins are categorized. If they are not categorized they will not be displayed. See Eclipse bug .

To add a new update site select, press the button "Add" and input the URL. This will then make this update site available and will allow you to install software from this site.

7.2. Manual installation of plugins (dropins folder)

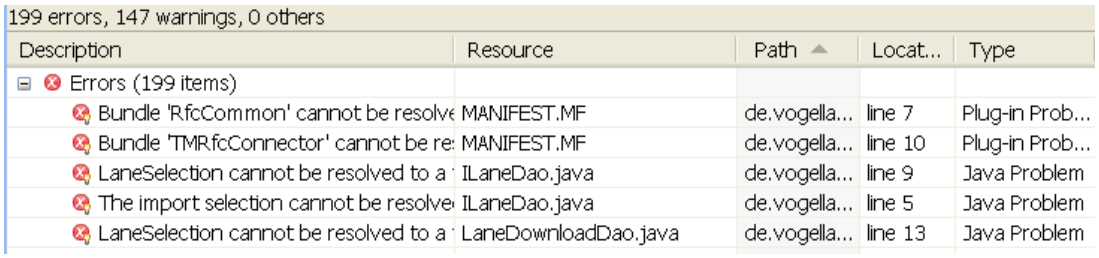
If you're using Plugins where no Software Site is available, then you can use the Dropins folder in your Eclipse installation directory.

To do this put the plugin into Eclipse "dropins" folder and restart Eclipse. Eclipse should detect the new plugin and install it for you.

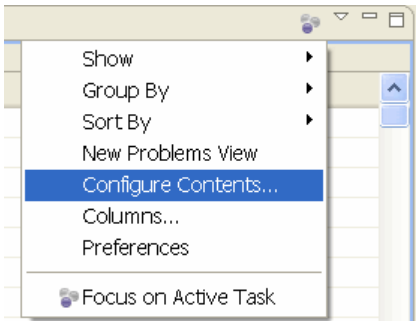
8. More Tips

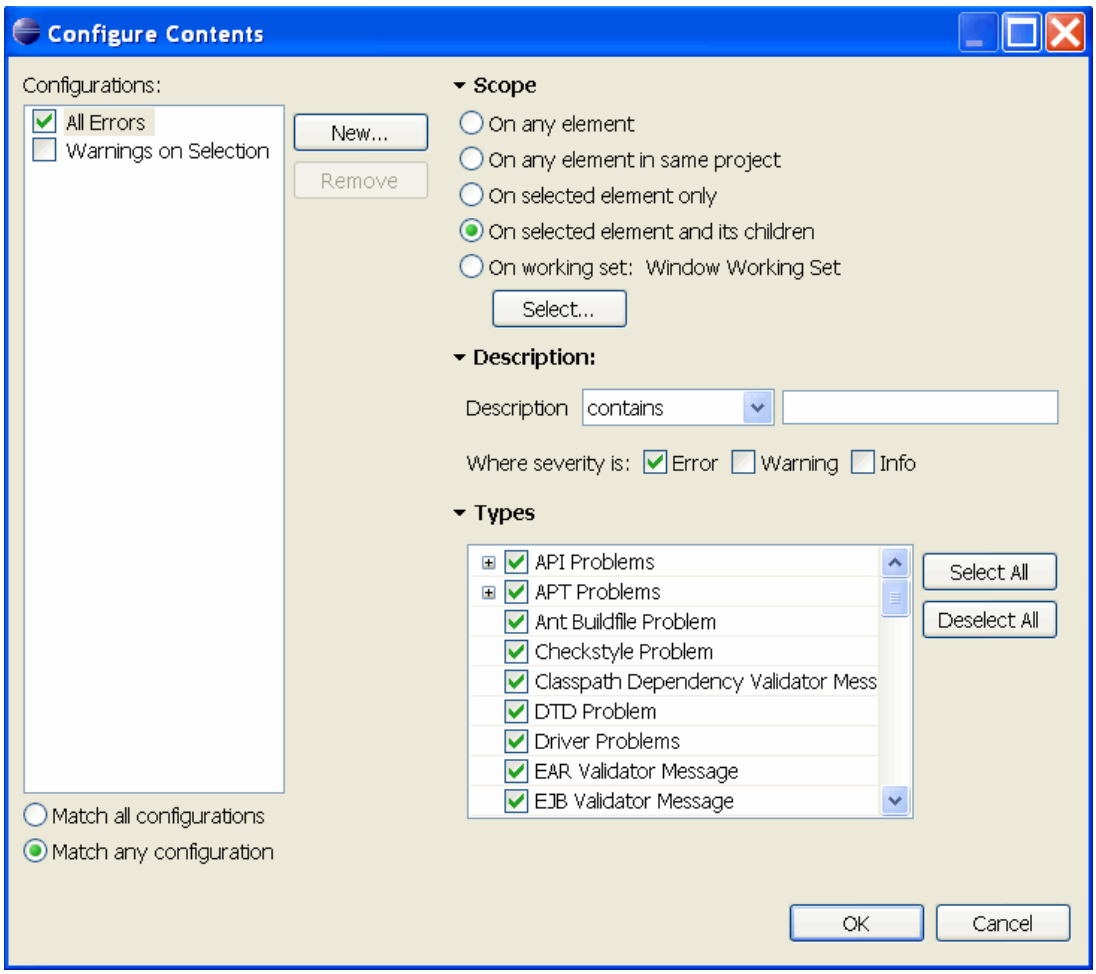
8.1. Problems view

The problems view displays problems in your projects. You can open it via Windows -> Show View -> Problems



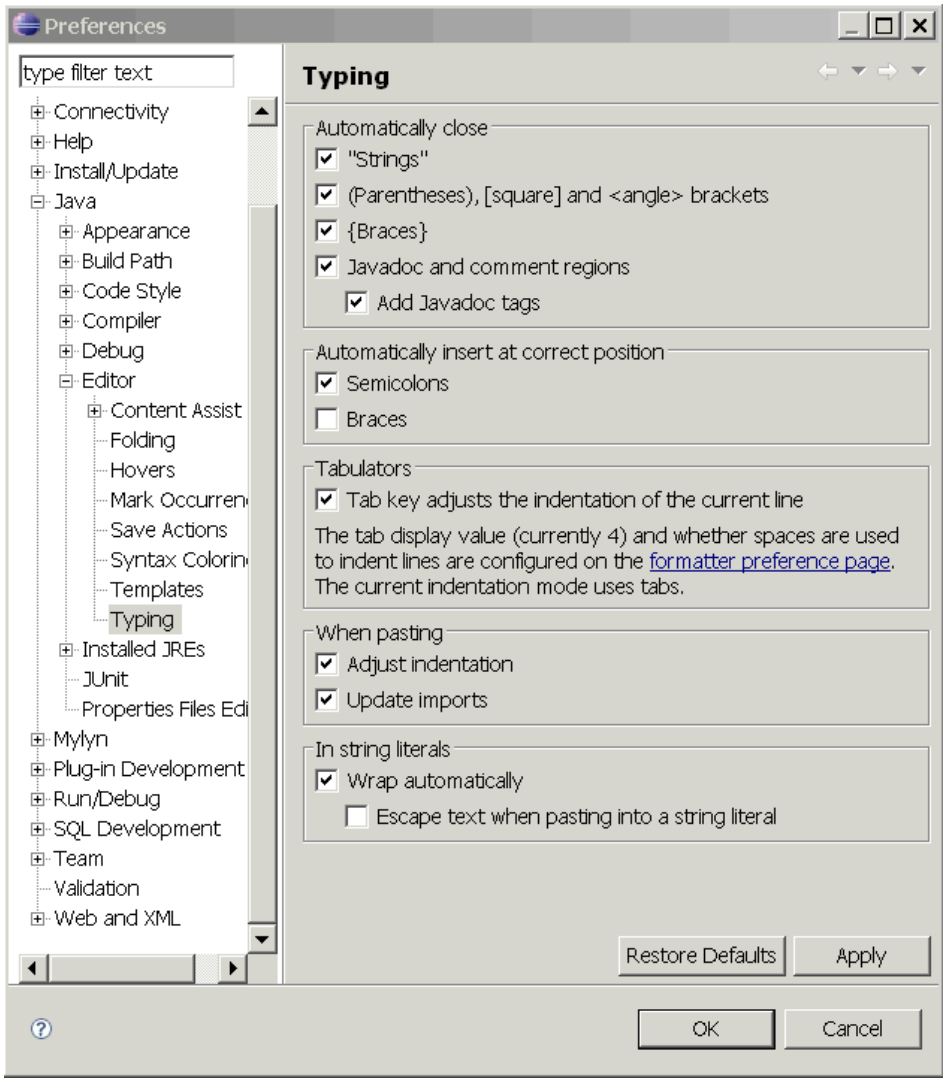
You can configure the problems view, e.g. if you only want to display the problems from the current selected project, select "Configure Contents".



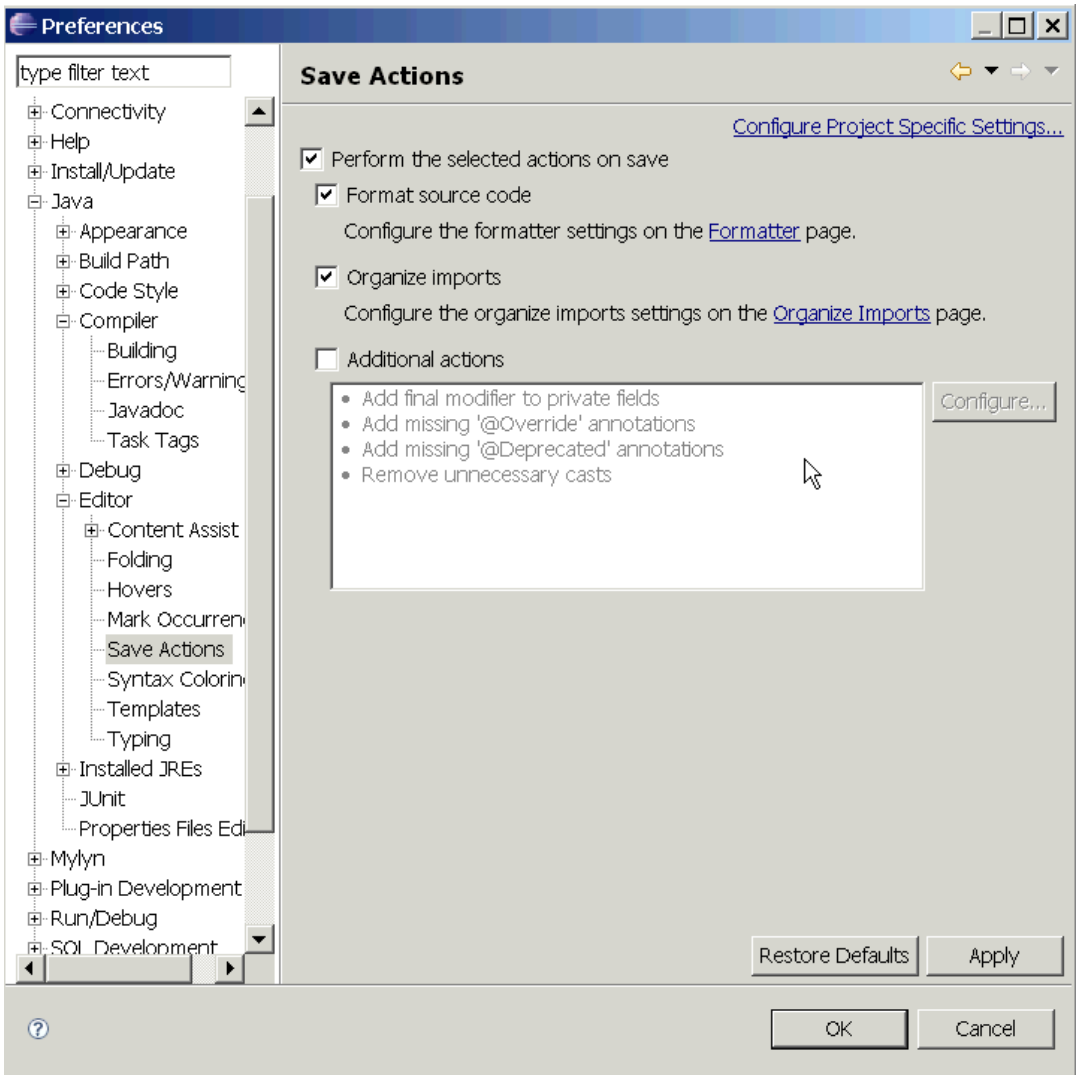


8.2. Important Preference Settings

Eclipse allows to set semicolons (and other elements) automatically.



Eclipse allows to format the source code and to organize the imports at save.



You can export your Preferences settings from one workspace via File -> Export -> General -> Preferences. Similar you can import them again into your workspace.

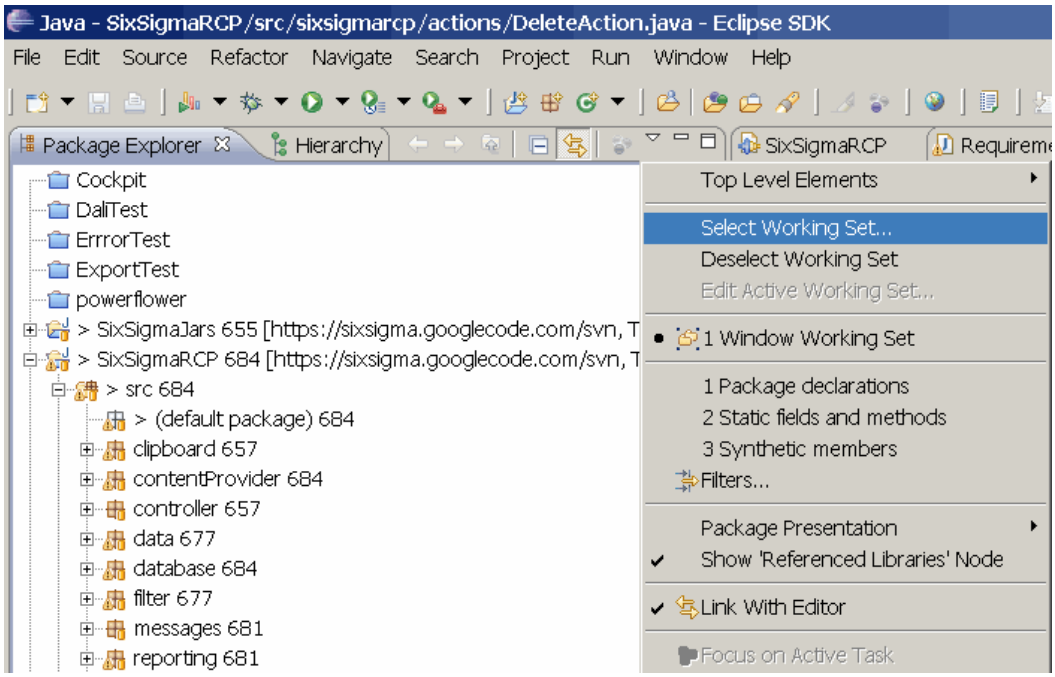
8.3. Task Management

If you use // TODO in the coding this indicates a task for eclipse and you find it in the task view of Eclipse.

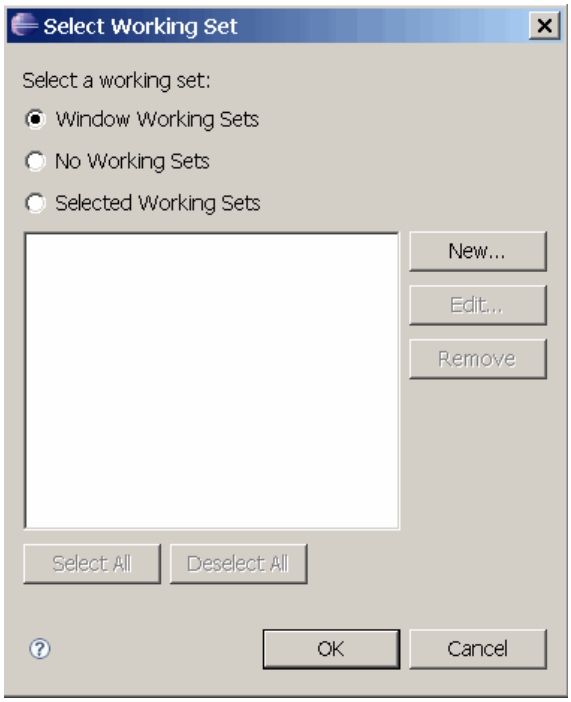
For more advanced tasks you can use Eclipse Mylyn Tutorial .

8.4. Working Sets

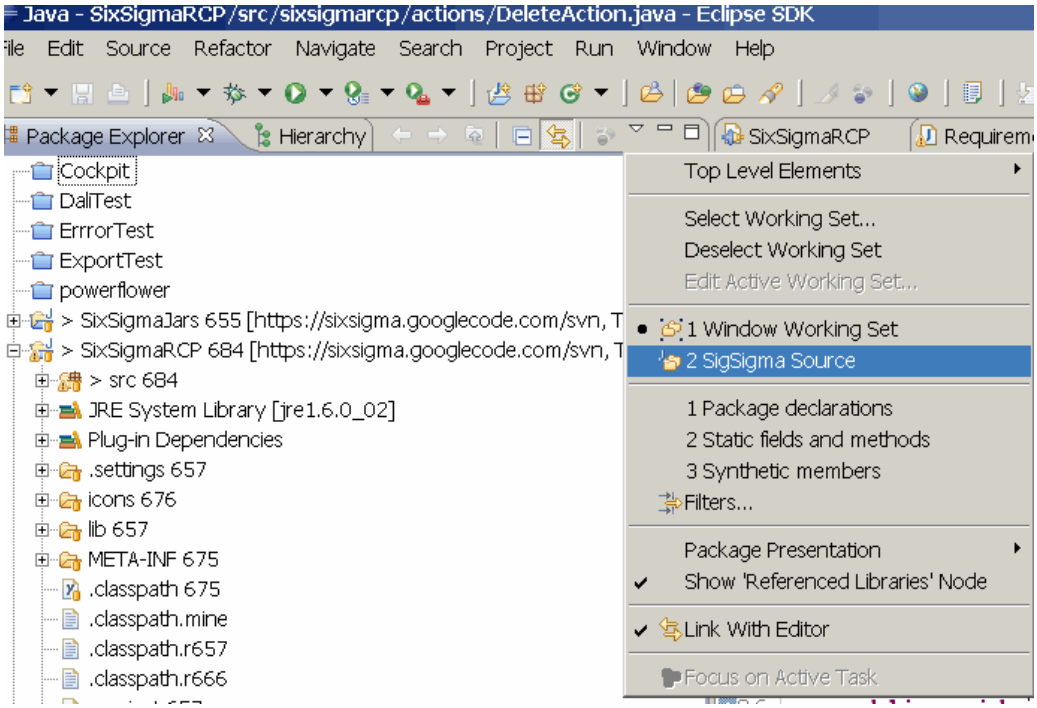
A common problem in Eclipse is that your data in your workspace grows and therefore your workspace is not well structured anymore. You can use working sets to organize your displayed projects / data. To setup your working set select in the Package Explorer -> Show -> Working Sets.



Press new on the following dialog to create a working set.



On the following dialog select java, select the source folder you would like to see and give it a name. You can now easily display only the files you want to see.



8.5. Synchronize package explorer with code display

The package explorer allows to display the associated file from the current selected editor. Example: if you working on foo.java and you change in the editor to bar.java then the display in the package explorer will change.

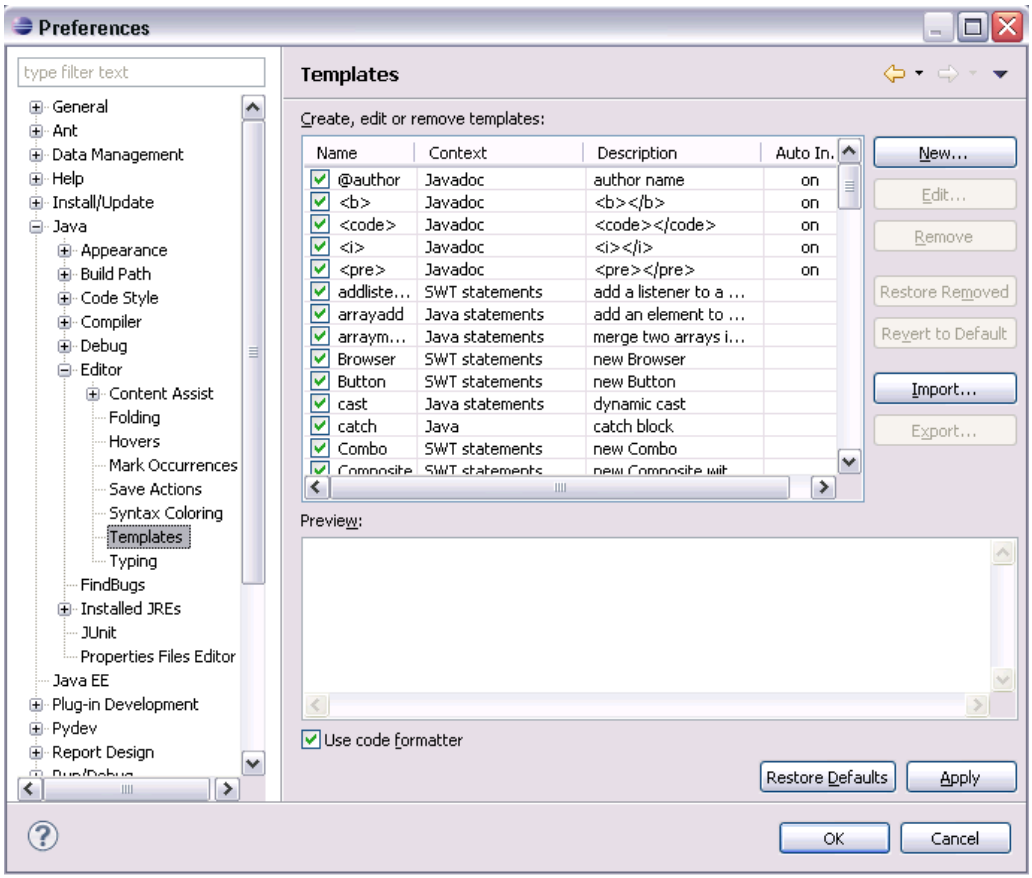
To activate this press "Link with Editor".

8.6. Code Templates

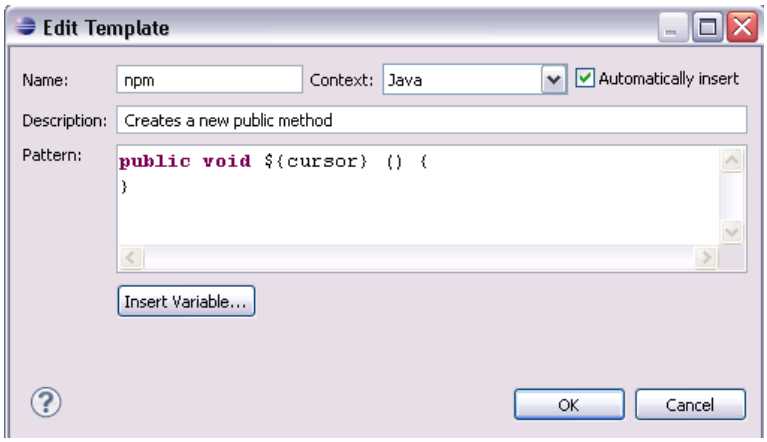
If you have to type frequently the same code / part of the document you can maintain templates which can be activate via autocomplete (Ctrl + Space).

For example lets assume you are frequently creating "public void name(){}" methods. You could define a template which creates the method body for you.

To create a template for this select the menu Window->Preferences and Open Java -> Editor -> Templates

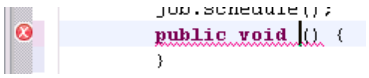


Press New. Create the following template. ${cursor} indicates that the cursor should be placed at this position after applying the template.



This this example the name "npm" is your keyword.

Now every time you type the keyword in the Java editor and press Ctrl+Space the system will replace your text with your template.



9. Next steps

To learn how to debug Eclipse Java programs you can use Eclipse Debugging

To learn Java Web development you can use with Servlet and JSP development with Eclipse Web Tool Platform (WTP) - Tutorial . If you want to develop rich stand-alone Java clients you can use Eclipse RCP - Tutorial

. Check out Eclipse Plugin Development - Tutorial to learn how to develop your own plugins.

Good luck in your journey of learning Java!

10. Thank you

Thank you for practicing with this tutorial.

Please note that I maintain this website in my private time. If you like the information I'm providing please help me by donating.

11. Questions and Discussion

For questions and discussion around this article please use the www.vogella.de Google Group. Also if you note an error in this article please post the error and if possible the correction to the Group.

I believe the following is a very good guideline for asking questions in general and also for the Google group How To Ask Questions The Smart Way.

12. Links and Literature

12.1. Source Code

http://www.vogella.de/code/codeeclipse.html Source Code of Examples

12.2. Eclipse Resources

12.3. Other Resources

Eclipse.org Homepage

Articles about Java, Eclipse and Webdevelopment from www.vogella.de

Articles about Eclipse development from www.vogella.de

Articles about Web development from www.vogella.de