

MindTelligent, Inc.

Eclipse Tutorial

Table Of Contents

Eclipse - Important concepts.	3
Perspective.....	4
Plugins.....	5
Changing the workbench arrangement:.....	8
Eclipse installation.....	11
Adjusting the default settings	13
Structure of folders in new Java projects.....	13
Switch compiler and editor to JDK 1.4	15
Use JDK instead of JRE (or install new JRE/JDK).....	16
Set classpath variables.....	18
The first Java-Project.....	20
Importing an existing project	24

Eclipse - Important concepts.

Eclipse is a framework to integrate different types of applications. One of these applications is the Java-IDE JDT (Java Development Tooling) that ships with the Eclipse platform.

The applications are delivered in form of plugins and automatically recognized and integrated by the platform. Eclipse additionally has its own resource management mechanism. Resources are usually files on the hard drive. They reside in the workspace, a special folder in the file system. If one Eclipse-application changes one of the resources the other installed applications will be notified about these changes. (Manipulating resources with external tools)

A user always works in the workbench, the visible part of the platform (GUI). The chosen perspective determines the actual appearance of the workbench. A perspective is a collection of so called *views* and *editors* and contribute special actions to the menu- and toolbar.

Most views show special information about the resources. Depending on the view only parts or inner relationships of the resources might be shown.

An editor directly works on a resource. There's a strict load-change-save cycle built into the core of the platform. Only if an editor saves its changes, the above mentioned notification mechanism will take place.

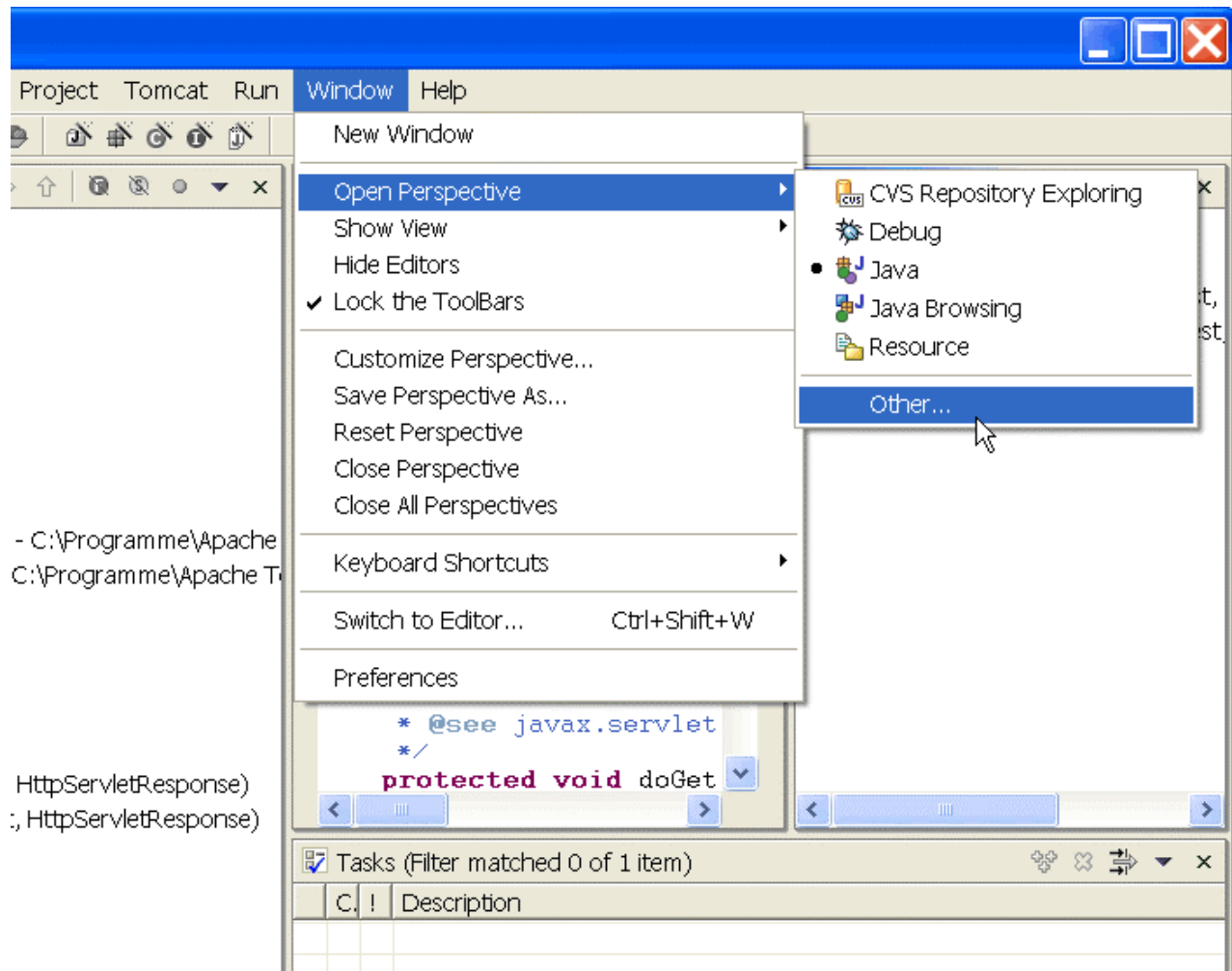
Special views can be directly connected to an editor (and not to the resource). For example the Outline *View* of the Java perspective is directly connected to the Java editor

What makes Eclipse special (among other things of course) is the extreme flexibility in which we can combine views and editors. This sometimes even leads to frustration. You can choose their arrangement in the workbench freely. You can add views and editors into an opened perspective - even if they have been defined in a totally different plugin. So we have the freedom to create our own custom-tailored development environment.

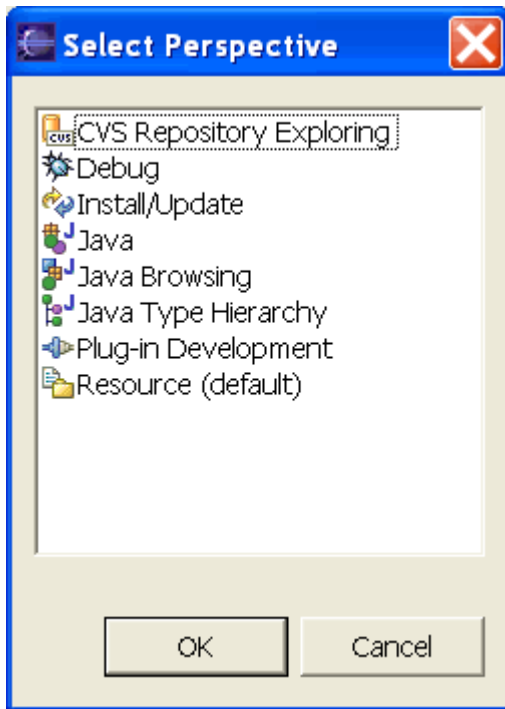
Perspective

To open a new perspective in the workbench do the following:

1. Choose 'Window -> Open Perspective' in the main menu:

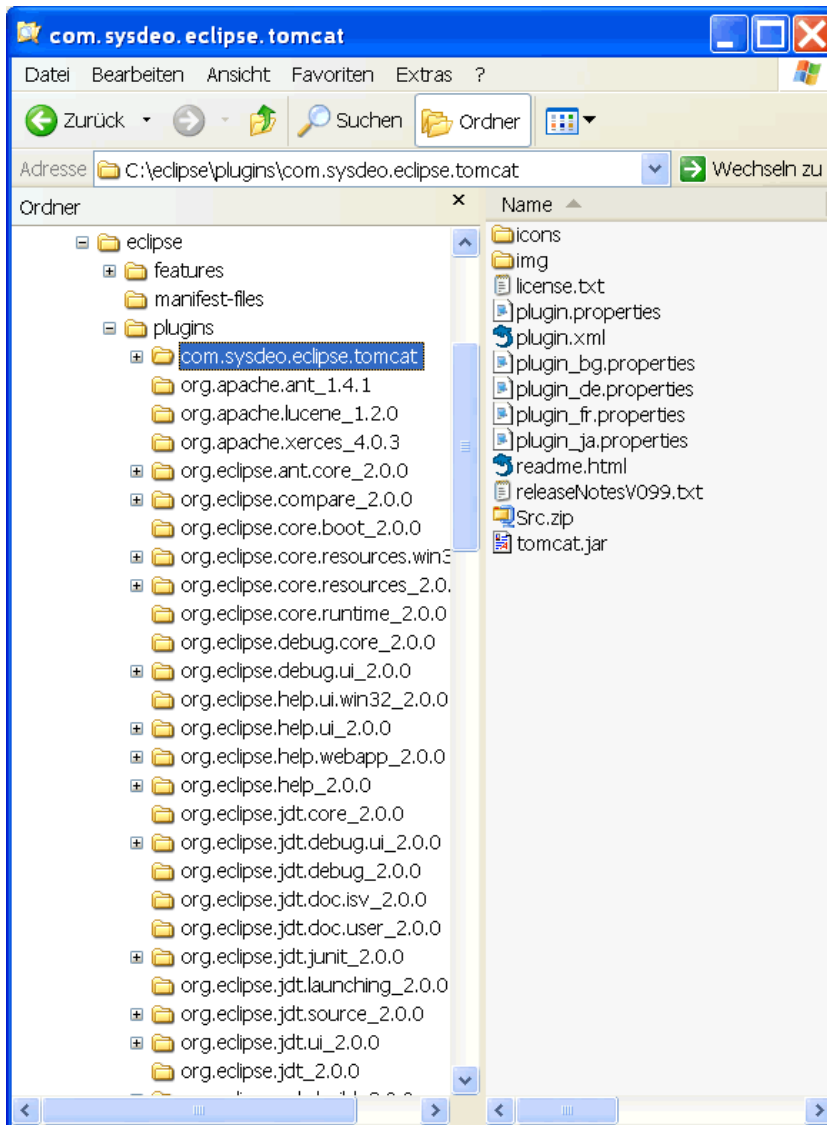


2. Chose 'Other' to see all installed perspectives:



Plugins

To install new applications, simply copy the plugins into the folder `$ECLIPSE_HOME/plugins` as shown here with the Tomcat plugin. You have to restart Eclipse to be able using the newly installed plugins. Depending on the plugin you now can choose a new perspective, find new entries in the menu and toolbar (as with the Tomcat plugin) or find new features in other structures. Hope that your plugin ships with decent documentation!

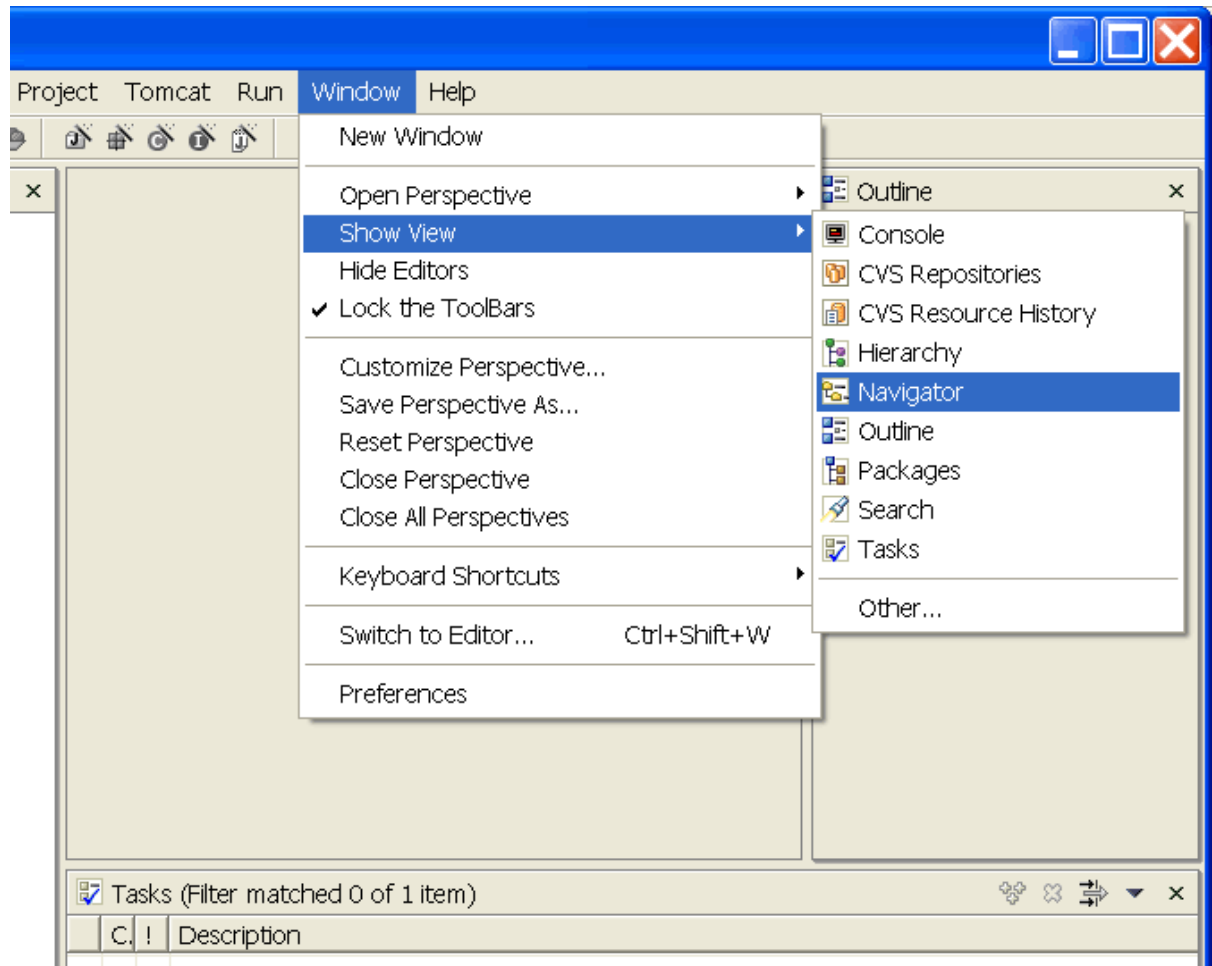


Workspace

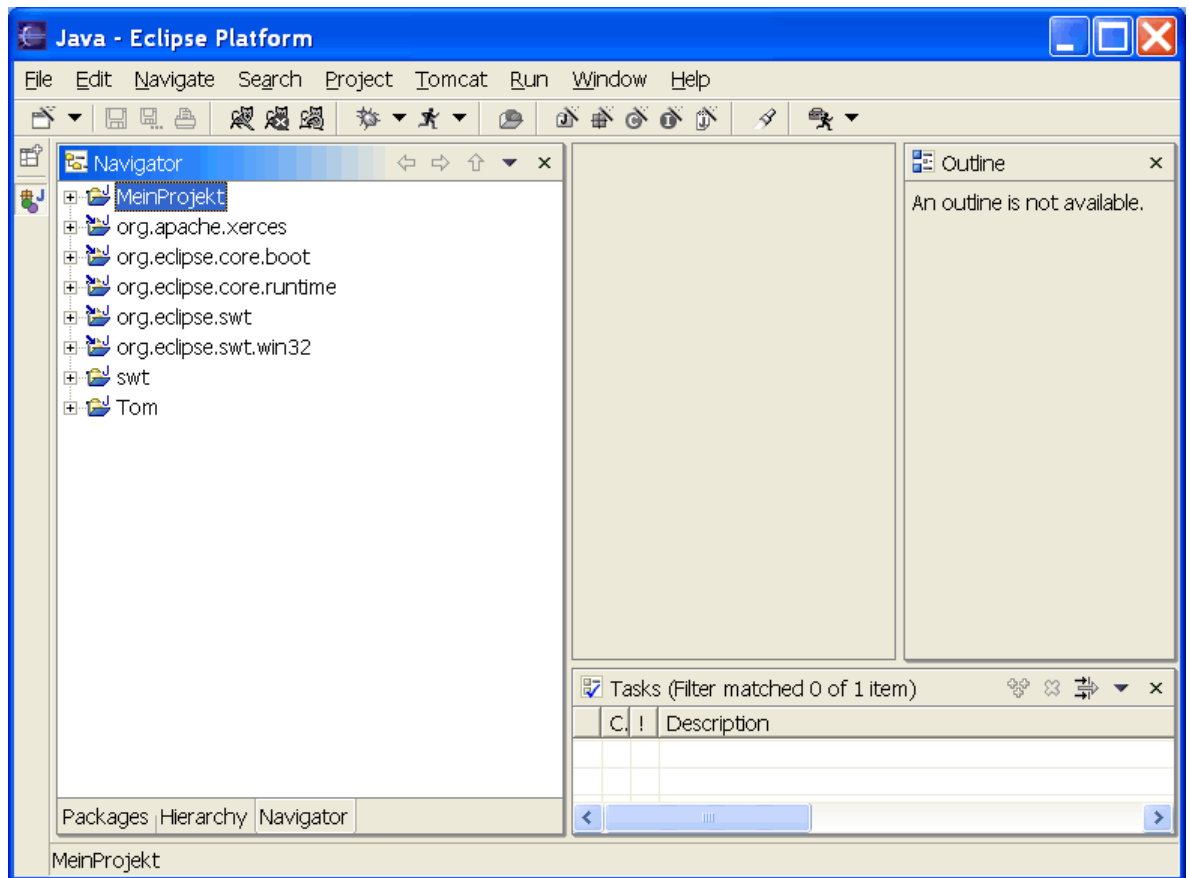
As default the root folder for the workspace is `$ECLIPSE_HOME/workspace`. If you take part in several projects or use a shared computer it makes sense to separate the workspace. You can choose an arbitrary directory for your workspace root folder with the `-data` option at Eclipse startup. For example `eclipse -data c:\MyWorkspace`.

Adding views to a perspective:

1. Open a perspective, e.g. the Java perspective.
2. Choose 'Show View -> Navigator' in the menu:

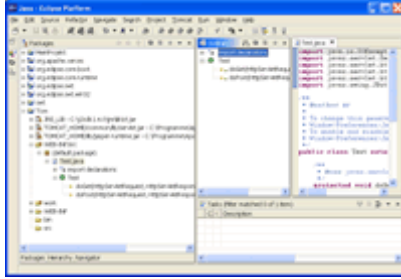


The navigator view has been added to the Java perspective.

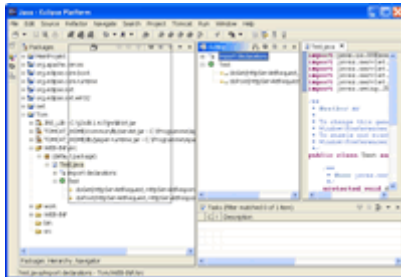


Changing the workbench arrangement:

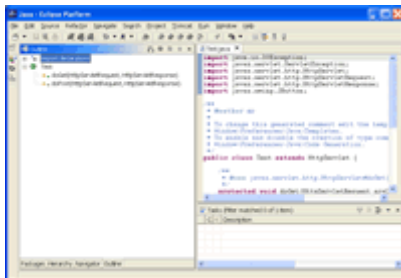
1. Left-click the title-bar of the view or editor that should be moved: (here: outline view)
2. Drag the cursor (with pressed mouse button) next to another view or editor until the cursor changes into a black arrow:
3. Release the mouse button.
The workbench layout is automatically adjusted:



- To add a view or editor as tabs do the following:
4. Left-click the title-bar of the view or editor that should be moved: (here: outline view)
 5. Drag the cursor (with pressed mouse button) on the title-bar of another view or editor until the cursor changes into the tab symbol:



6. Release the mouse button.
The workbench layout is automatically adjusted:



Manipulating resources with external tools

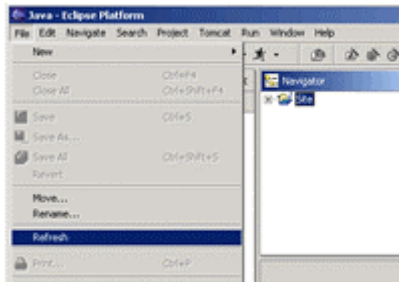
If you change resources in the workspace with external tools, there's no automatic notification.

If the resource you changed externally is opened in a Eclipse editor you will be asked if you want to load the changes from the disk:



If you create new files or folders in your project folder externally you have to tell Eclipse:

1. Select the folder in which you created the new files / folders in a view (e.g. navigator).
2. Choose 'File -> Refresh'



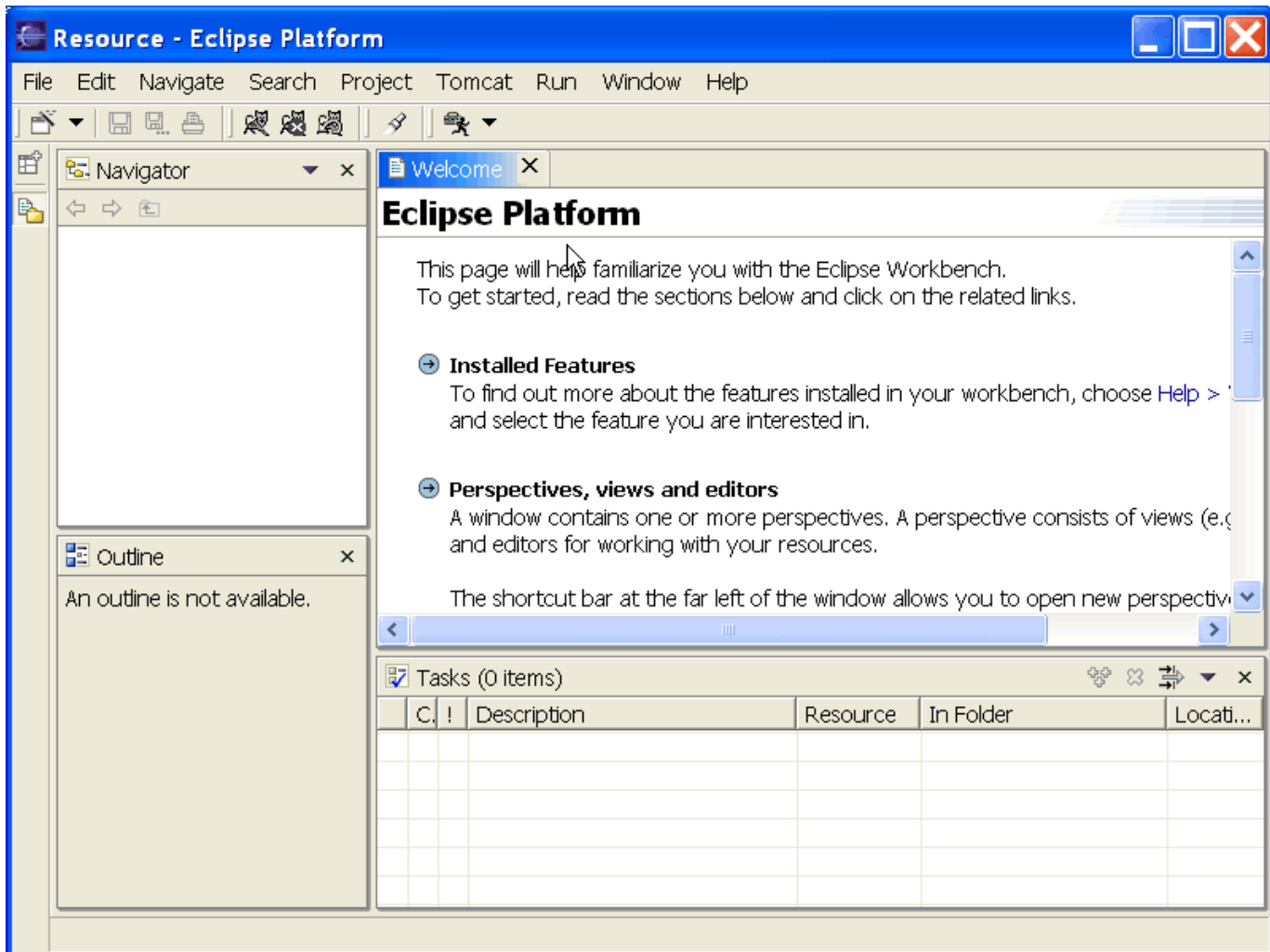
You now can see the files/folders in the according views.

Eclipse installation

You need:

- System with Windows, Linux, Solaris, QNX or Mac OS/X operating system. Preferably 256 MB RAM or more.
- A Java 2 runtime environment (JRE) or Java 2 Software Development Kit (J2SDK). Eclipse needs version 1.3 or higher .
We recommend JDK 1.4.1 as it has the least problems with hot code replacement and is the fastest. This is the Java-version that Eclipse (a Java application) runs with, not the version you develop for!
- Eclipse 2.0.1-archive. Either the Platform Runtime Binary Archive together with the JDT Runtime Binary oder the Eclipse SDK.

1. Install JRE resp. J2SDK.
2. Extract the Eclipse-archive to an arbitrary location on your hard drive. You have to keep the path settings. In Windows you could unzip directly to C:\. You will get a C:\eclipse directory (from now on referred to as *eclipse_home*).
Under Unix you could use the /opt-directory.
In Windows Eclipse finds an installed JRE/SDK automatically via the registry. Alternatively you can copy the jre-Folder (including its contents) into the *eclipse_home* directory. Another option is to specify the path to the JRE/SDK with the command-line option -vm.
3. Start Eclipse by executing eclipse.exe resp. eclipse.sh in the *eclipse_home* directory. You will see a splash screen and after some time the workbench will show up:



Adjusting the default settings

We are going to describe some useful changes of the default settings of Eclipse - as existing after the installation.

- ❑ Structure of folders in new Java projects.
- ❑ Switch compiler and editor to JDK 1.4.
- ❑ Use JDK instead of JRE (or install new JRE/JDK).
- ❑ Set classpath variables.

Structure of folders in new Java projects.

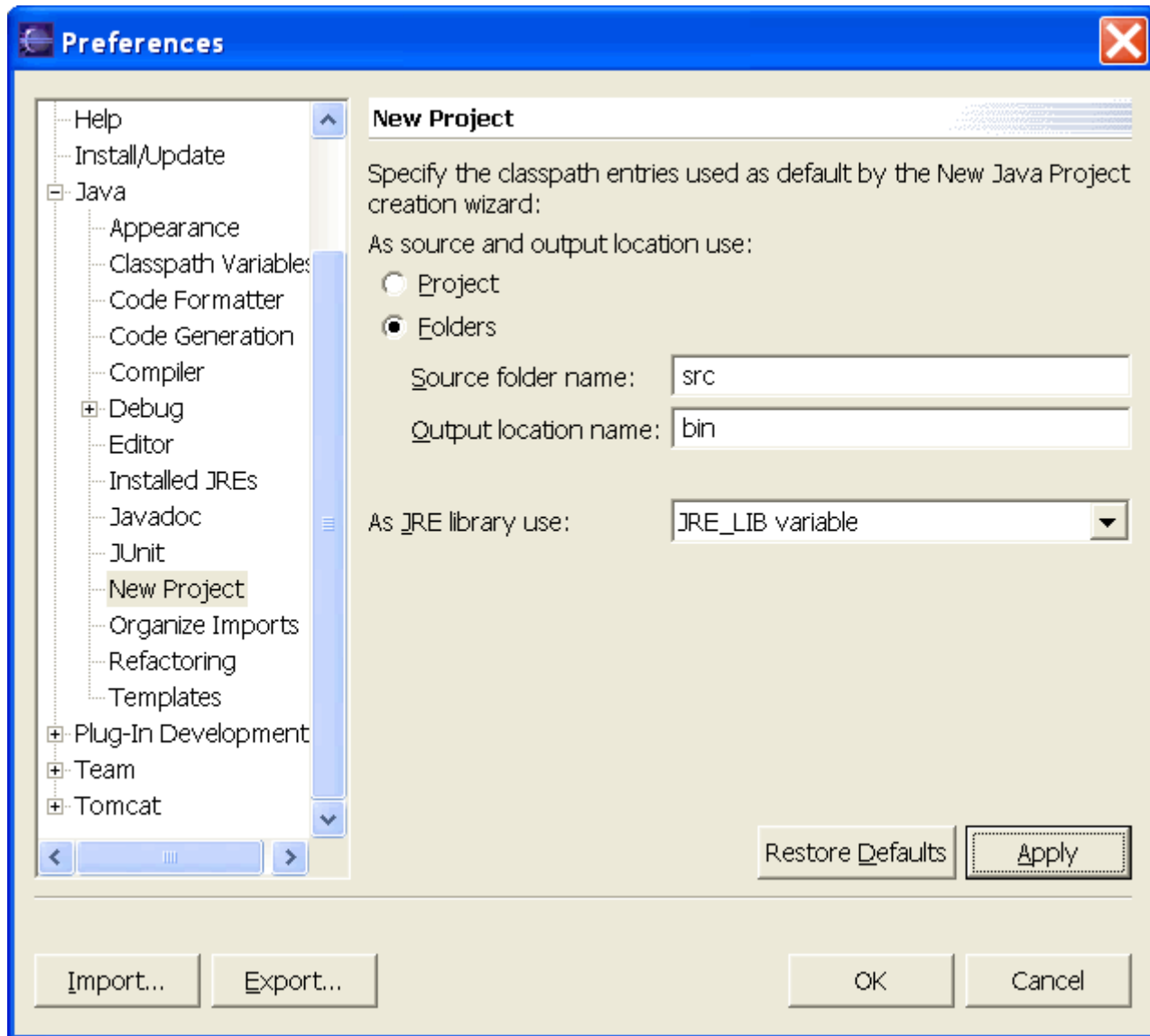
By default source- and classfiles will be saved directly in the project folder. It's better to separate source code from compiled code (.class files).

It makes sense to have a special folder (source folder) that contains `_only_` source code.

A project can grow concisely by adding additional source folders. You could give individual modules or subprojects its own folders.

1. Choose 'Window -> Preferences' from the menu-bar.
2. Expand the node 'Java' and choose 'New Project'.
3. Choose the radio-button 'Folders'.
4. Change the suggested values for the source folder ('Source folder name') and the folder where the compiled code will go ('Output location name') as needed.

In order to create new packages and classes you will need a source folder. They are shown with their own icon in the Package Explorer view.

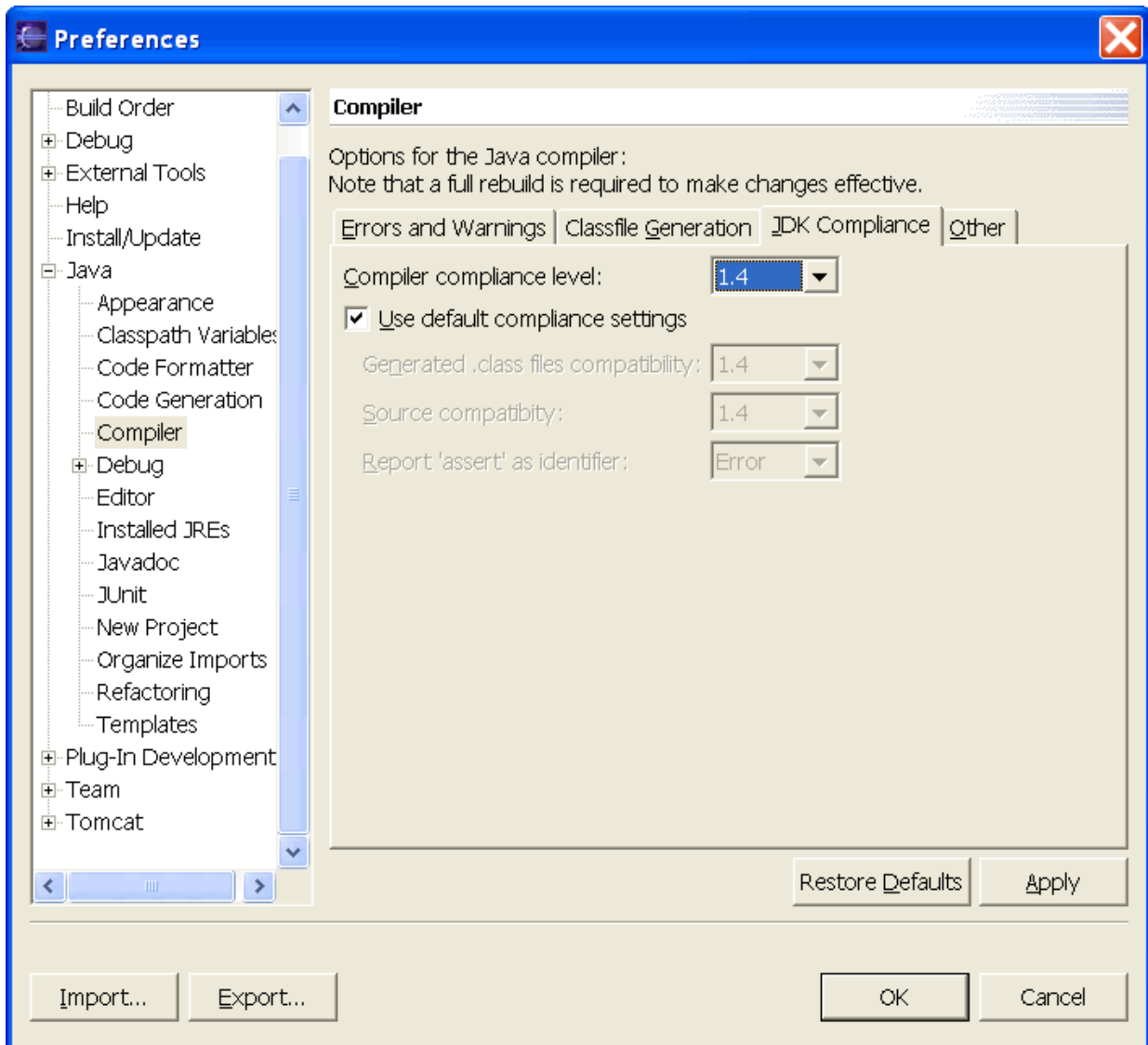


5. Press 'Apply'.
6. Press 'OK'.

Switch compiler and editor to JDK 1.4 .

The Java compiler and editor of Eclipse can be set up to work for different JDK versions.

1. Choose 'Window -> Preferences' from the menu-bar.
2. Expand the node 'Java' and choose 'Compiler'.
3. Choose button 'JDK Compliance'.
4. Choose '1.4' from selection list 'Compile compliance level'.



5. Press 'Apply'.
6. Press 'OK'.

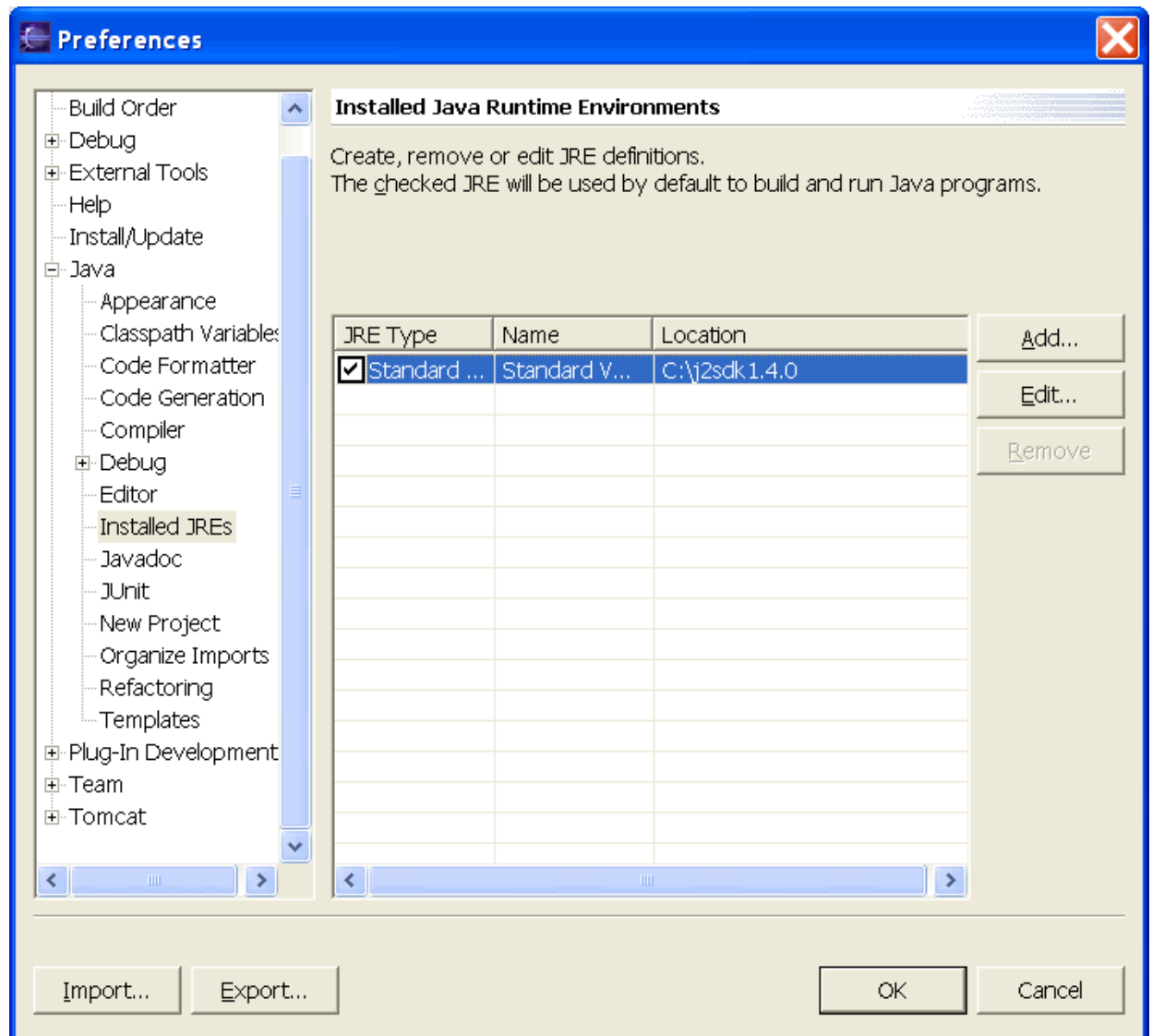
Use JDK instead of JRE (or install new JRE/JDK).

To see the source code of the standard Java classes you have to tell Eclipse to use the correspondent JDK. (By default Eclipse uses the JRE, which comes without source code (file src.zip))

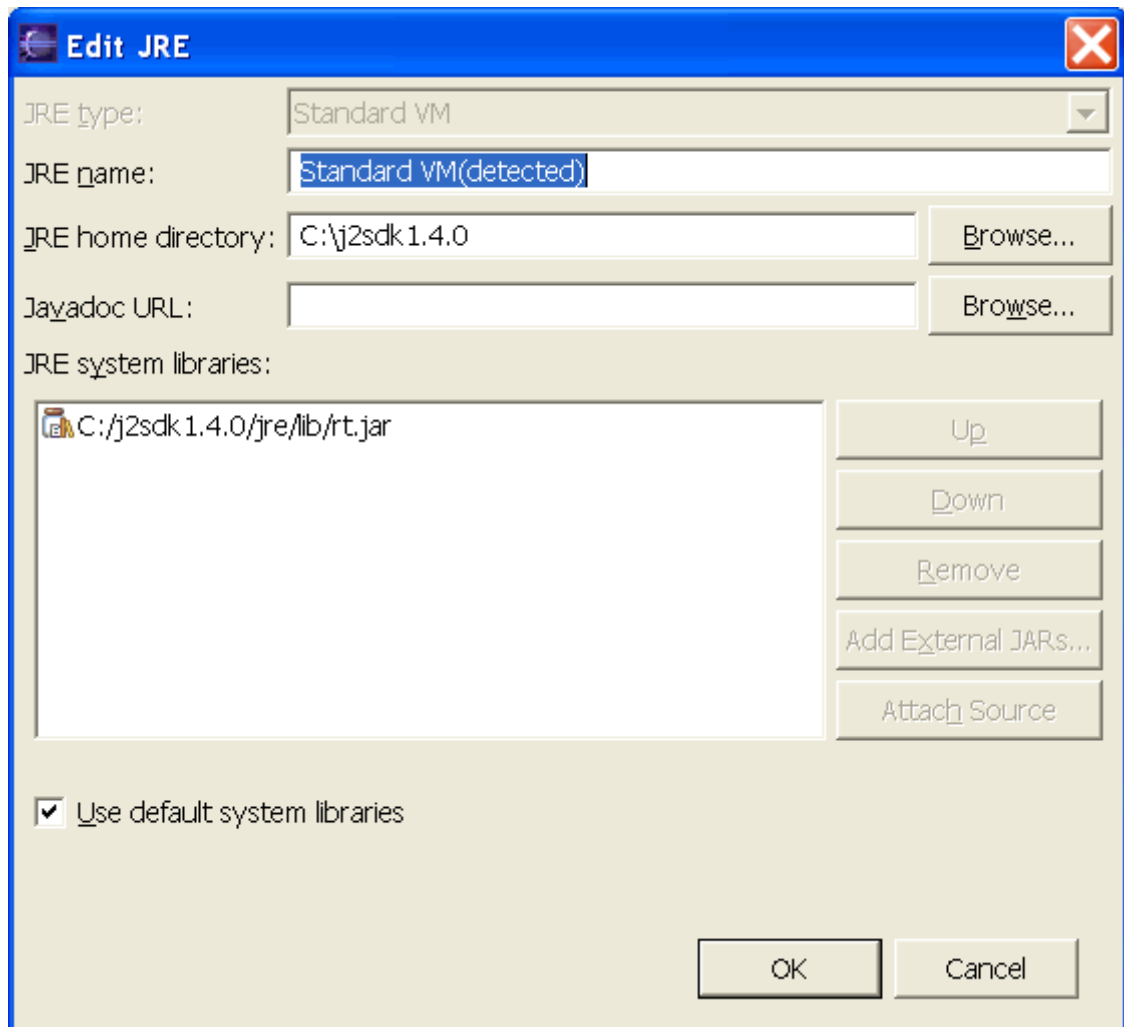
When using the JDK additional help will be offered:

Code Assist can show you method names and blend in the JavaDoc comments as hover help. Prerequisite: The JDK is installed on the system.

1. Choose 'Window -> Preferences' from the menu-bar.
2. Expand the node 'Java' and choose 'Installed JREs'.
3. Choose first line. (Name: Standard VM).



4. Press button 'Edit' (or 'Add').
The dialog 'Edit JRE' (or 'Add JRE') shows up:

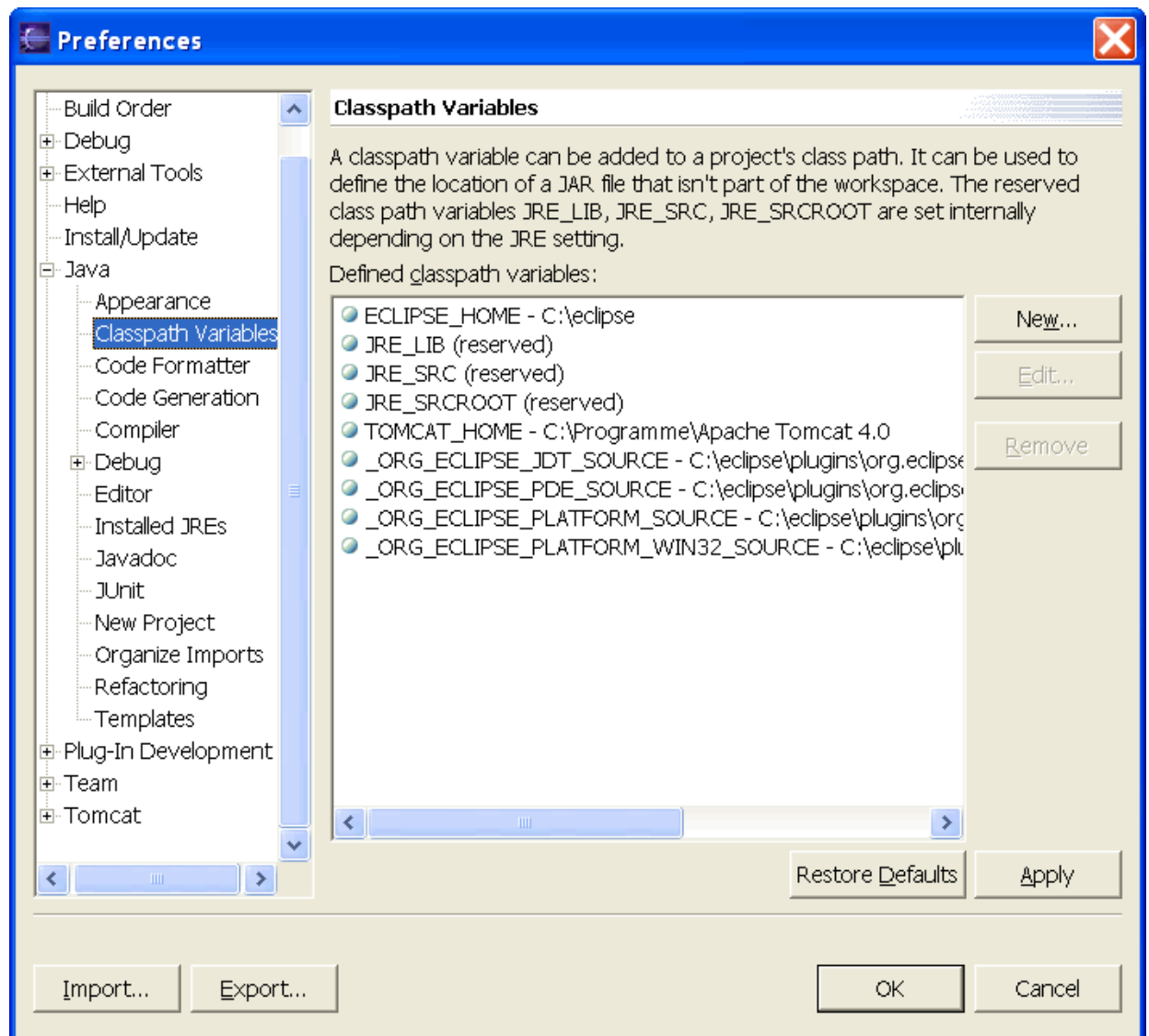


5. Press button 'Browse' in the line 'JRE home directory'.
6. Choose installation directory in the folder selection dialog. Under Windows this could be C:\j2sdk1.4.1.
7. Press 'OK' for folder selection.
8. Press 'OK' for dialog 'Edit JRE'.
9. Press 'OK' for dialog 'Preferences'.

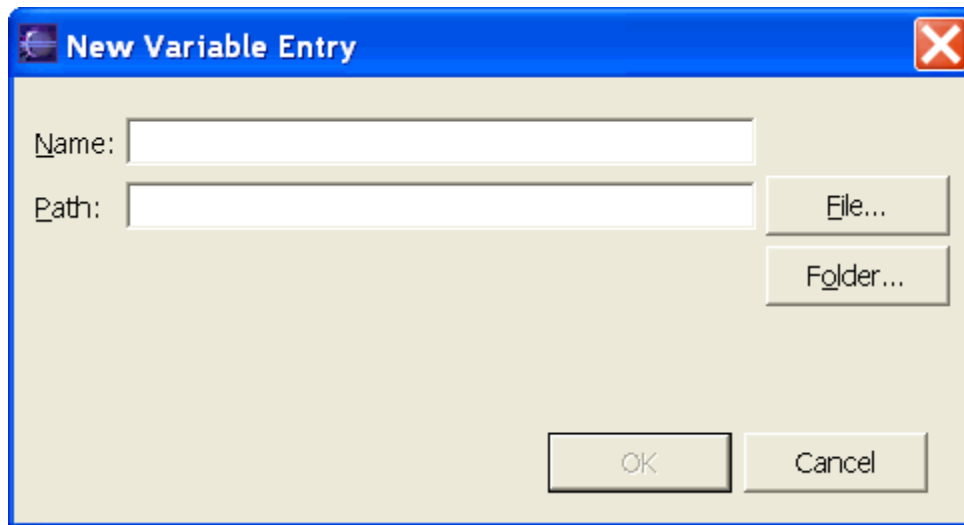
Set classpath variables.

Classpath variables are defined at workbench level. Folders and jar or zip files can be given names which makes it easier to include them in the buildpath.

1. Choose 'Window -> Preferences'.
2. Expand the node 'Java' and choose 'Classpath Variables'.
The following dialog pops up:



3. Press the button 'New' .
The dialog 'New Variable Entry' pops up:



4. Specify the variable name in the text field next to 'Name'.
5. a) If a variable should point to a file (zip- or jar-archive), press the button 'File'.
b) If a variable should point to a folder, press the button 'Folder'.

Variables pointing to a folder must be extended to a complete path from within the project where they are used. That means there have to be zip- or jar-archives in the folder that are going to be added to the build path.

An according dialog pops up.


6. Choose the file or folder in that dialog.
7. Confirm the dialog.
8. Press 'OK' in the dialog 'New Variable Entry'.
9. Press 'OK' in the dialog 'Preferences' Dialog.

The first Java-Project

We are going to create a simple Java project in which the source folder is separated from the output folder.

Creation of a Java project:

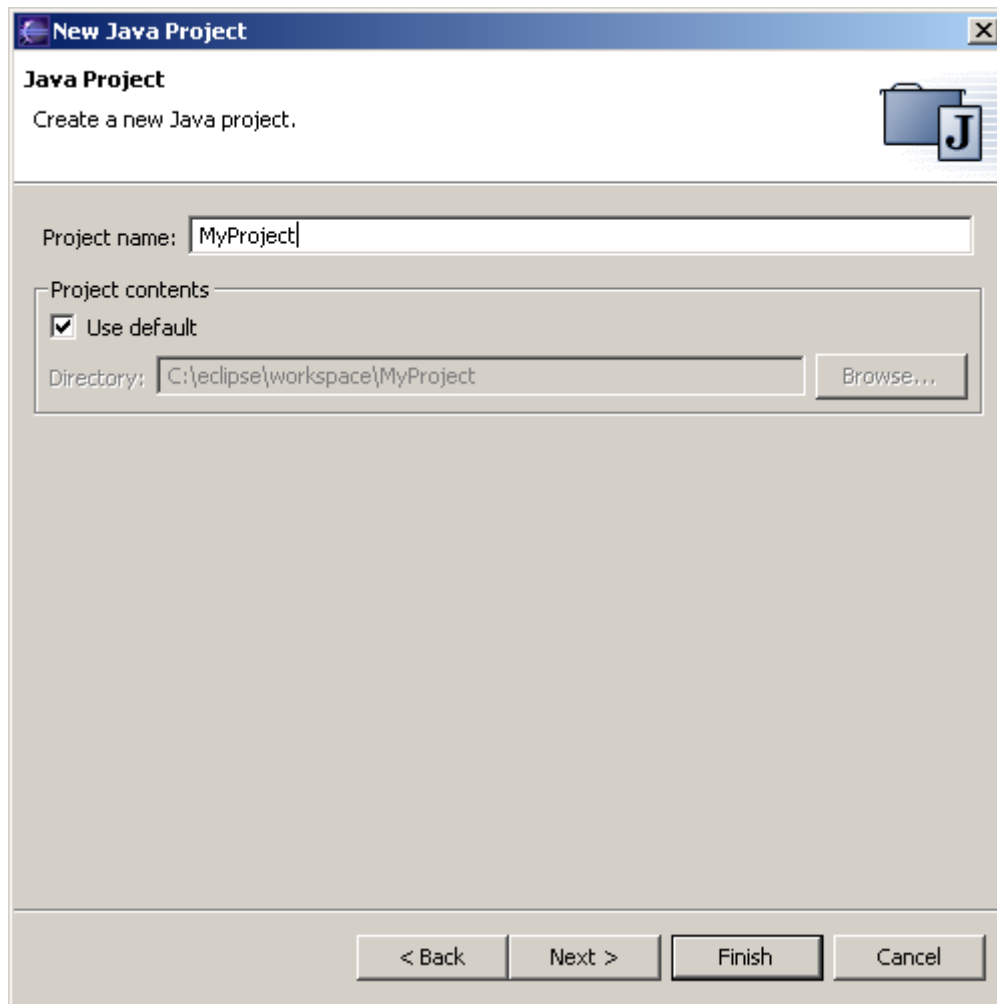
No matter if you want to start an all new project or want to import an existing one - you need to create an Eclipse Java project first.

1. Open the Java perspective.
2. Open the 'New Java Project' wizard.
There's several ways of starting this wizard. The easiest is pressing the button  in the toolbar.

Alternatively you can use the context menu (right mouse button) of the Package Explorer view and choose 'New -> Project' there.

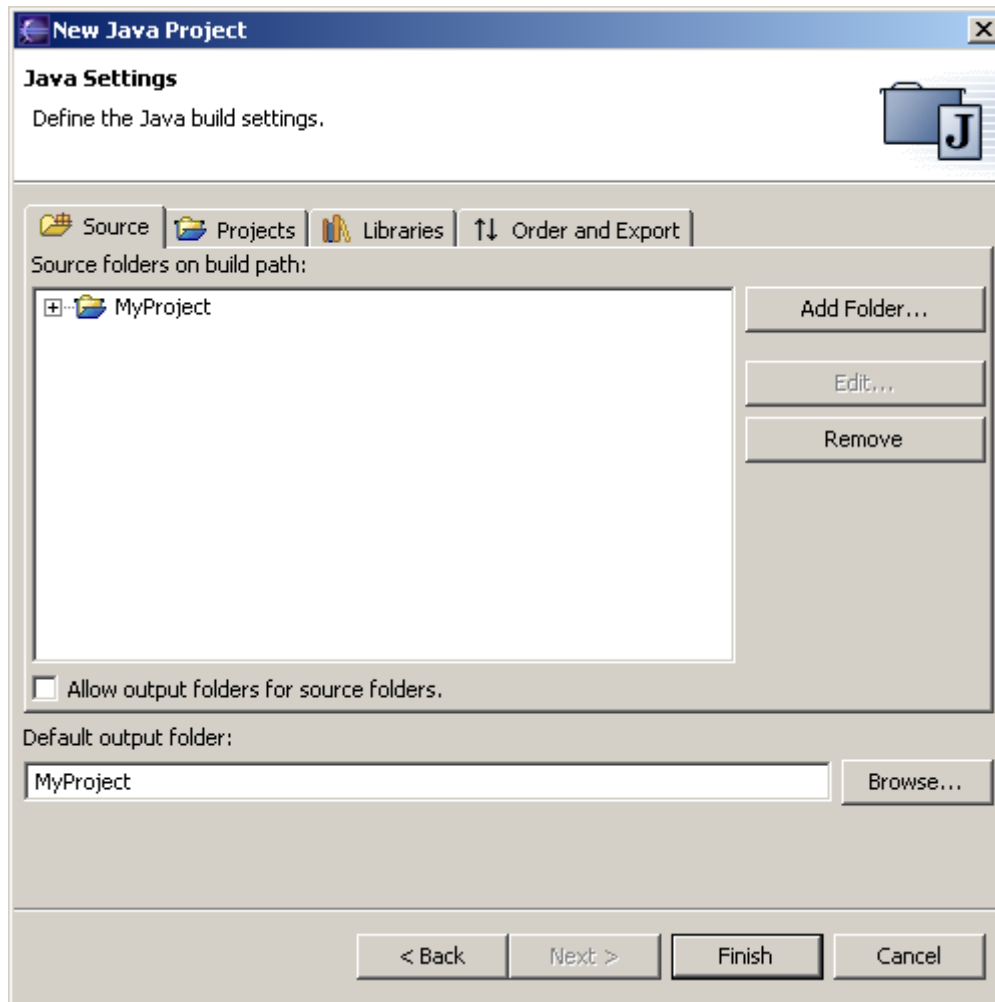
You could also choose 'File -> New -> Project' to get to a dialog that lets you choose a Java project.

The 'New Java Project' dialog pops up:



3. Choose a non existing name for your project, e.g. "MyProject" default for the project folder is *workspace_home/MyProject*
4. Press 'Next'.
The 'New Java Settings' dialog pops up:

This dialog really is editing the .classpath file. This file is stored in the project folder and can be changed any time.



The 'Source' tab lets you specify folders that contain Java sources (.java files).

Default after installation of Eclipse is the project folder but we will separate the sources from the compiled code..

If you have done the adjustments of the default settings you don't have to do steps 5-9.

5. Select the radio button 'Use source folders contained in the project'.
6. Press the button 'Create New Folder'.
The 'New Source Folder' dialog pops up:



7. Enter "src" in the text field.

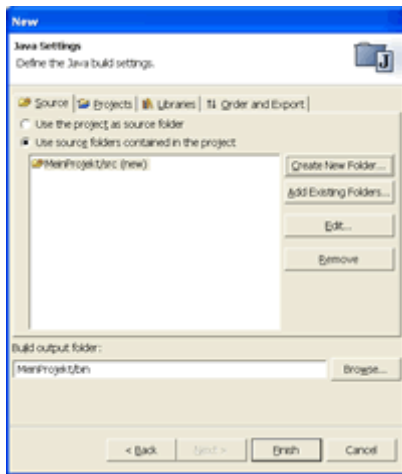
- Press 'OK'.
The 'Source Folder Added' dialog pops up.



- Press 'Yes'.

The output folder for compiled classes will be specified as *workspace_home/MyProject/bin* .

The dialog 'New Java Settings' looks like that now:

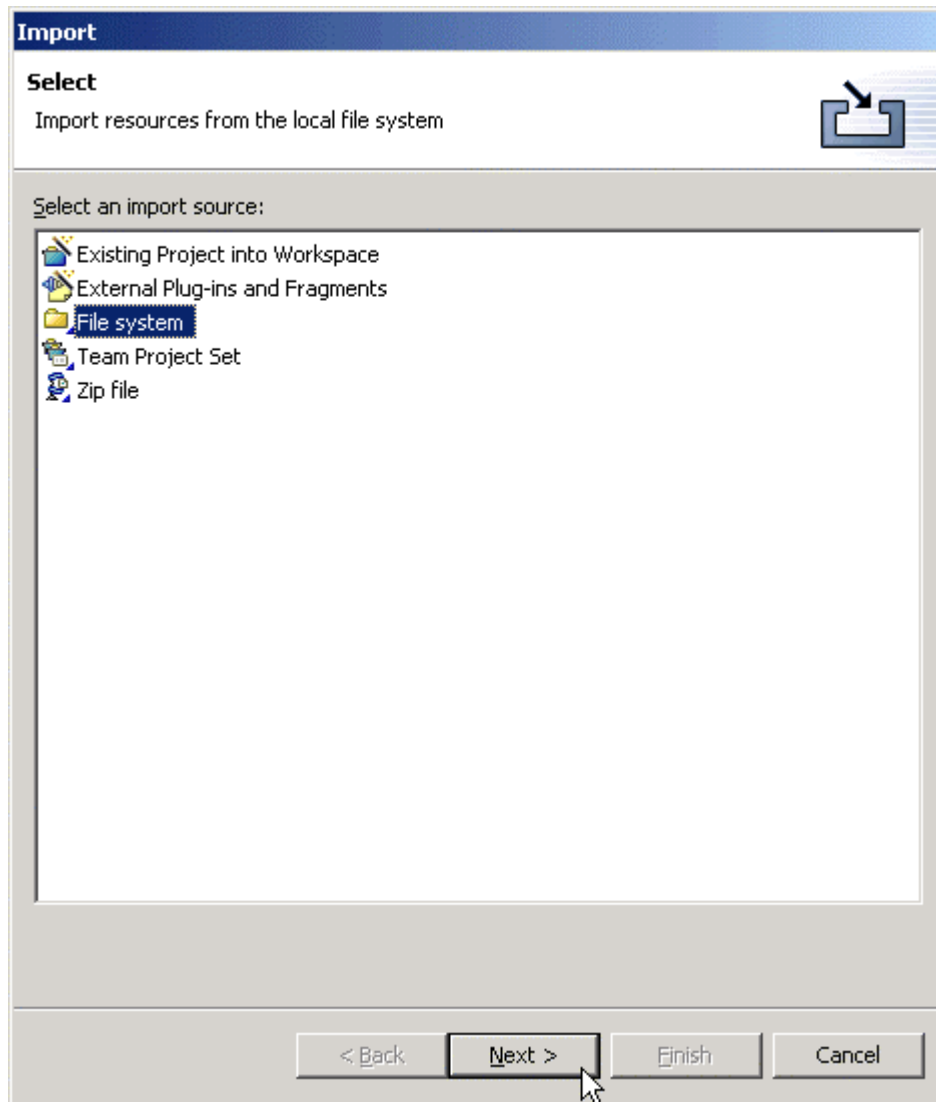


- Press 'Finish'.

Importing an existing project

To import an existing project, e.g. from JBuilder you have to:

1. Create a Java project.
2. Choose 'File -> Import'.
The 'Import Select' dialog pops up.
3. Select 'File System'.

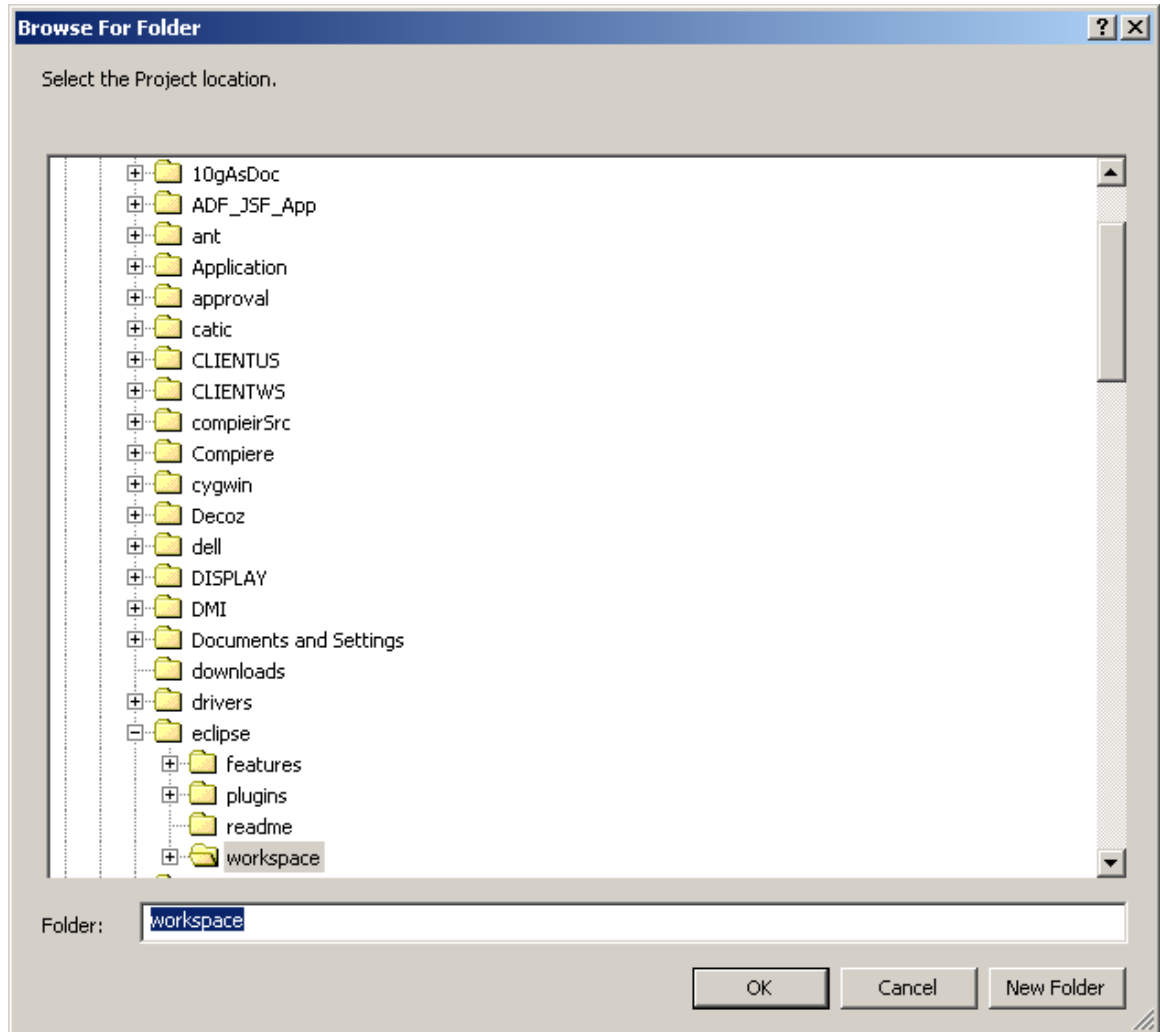


4. Press 'Next'.
The 'Import File System' dialog shows
5. Specify the directory that contains the project to import with the first 'Browse' button.
This directory is the root for the following selection dialog.
6. Select the folders and files that should be imported into the project.

7. Specify the folder in that you want to import the files with the second 'Browse' button.

You can choose here between existing project folders and their subfolder. (If the build path is already set it's important to specify the correct source folder, see below)

The option 'Create complete folder structure' also creates the folder hierarchy in the project folder.



8. Press 'Finish'.
The selected folders and files have been added to your project.
9. According to the structure of the imported Java project the build path has to be set.

The above example only imported source files of a JBuilder project. In JBuilder these reside in a folder 'src'. If the adjustments of the default settings

have been done, this folder already also is the source folder of the (Eclipse) Java project.