



by Lars Vogel

## Tutorial

394



# Eclipse IDE Tutorial

## Lars Vogel

Version 3.0

Copyright © 2007, 2008, 2009, 2010, 2011, 2012 Lars Vogel

10.04.2012

### Revision History

Revision 0.1	18.07.2007
Created	
Revision 0.2 - 3.0	18.05.2008 - 10.04.2012
bugfixes and enhancements	

## Eclipse Java IDE

This tutorial describes the usage of Eclipse as a Java IDE. It describes the installation, the creation of Java programs and tips for using Eclipse. This tutorial is based on Eclipse

### Table of Contents

1. What is Eclipse?
2. Getting started
  - 2.1. Java
  - 2.2. Install Eclipse
  - 2.3. Start Eclipse
3. Eclipse UI Overview
  - 3.1. Workspace
  - 3.2. Parts
  - 3.3. Perspective
  - 3.4. Eclipse IDE Perspectives
  - 3.5. Java Perspective and Package Explorer
  - 3.6. Linking the package explorer with the code editor
  - 3.7. Problems view
4. Create your first Java program
  - 4.1. Create project
  - 4.2. Create package
  - 4.3. Create Java class
  - 4.4. Run your project in Eclipse

FOLLOW  
ME ON



FOLLOW  
ME ON





- 4.6. Run your program outside Eclipse
- 5. Content Assist, Quick Fix and Class Navigation
  - 5.1. Content assist
  - 5.2. Quick Fix
- 6. Opening a class
- 7. Generating code
- 8. Refactoring
- 9. Eclipse Shortcuts
- 10. Using jars (libraries)
  - 10.1. Adding a library (.jar ) to your project
  - 10.2. Attach source code to a Java library
  - 10.3. Add the Javadoc for a jar
- 11. Updates and Installation of Plugins
  - 11.1. Eclipse Update Manager
  - 11.2. Manual installation of plug-ins (dropins folder)
  - 11.3. Eclipse Marketplace
  - 11.4. Share plug-in
  - 11.5. Restart required?
- 12. Efficiency Settings
  - 12.1. Eclipse Preferences
  - 12.2. Automatic placement of semicolon
  - 12.3. Imports and Source code formating
  - 12.4. Filter import statements
  - 12.5. Templates
  - 12.6. Configuring the editors for a file extension
  - 12.7. Code Templates
  - 12.8. Export / Import Preferences
  - 12.9. Task Management
  - 12.10. Working Sets
- 13. Eclipse Help and Community
  - 13.1. Eclipse Bugs
  - 13.2. Online documentations
  - 13.3. Asking (and answering) questions
  - 13.4. Webresources
- 14. Next steps
- 15. Thank you
- 16. Questions and Discussion
- 17. Links and Literature
  - 17.1. Source Code
  - 17.2. Eclipse Resources
  - 17.3. vogella Resources

## 1. What is Eclipse?

Eclipse is created by an Open Source community and is used in several different development environment for Java or Android applications. Eclipse roots go back to

Most people know Eclipse as an integrated development environment (IDE) for Java leading development environment for Java with a market share of approx. 65%.




The Eclipse project is governed by the Eclipse Foundation member supported corporation that hosts the Eclipse projects and helps to cultivate

FOLLOW  
ME ON



FOLLOW  
ME ON



 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

Eclipse can be extended with additional functionalities. Several Open Source projects have extended Eclipse with additional components. It is also possible to use Eclipse creating general purpose applications (Eclipse RCP).

## 2. Getting started

### 2.1. Java

Eclipse requires an installed Java Runtime. I recommend to use Java 7 (also known as Java 6).

Java comes in two flavors, the Java Runtime Environment (JRE) and the Java Development Kit (JDK). The JRE contains only the necessary functionality to start Java programs, while the JDK contains the development tools.

Eclipse contains its own development tools, e.g. Java compiler. Therefore for this tutorial we use the JRE.

The JDK is required if you compile Java source code outside Eclipse and for advanced scenarios. For example if you use automatic builds or if you develop web applications are not covered in this tutorial.

Java might already be installed on your machine. You can test this by opening a command prompt (using Windows: Win+R, enter "cmd" and press Enter) and by typing in the following

```
java -version
```

If Java is correctly installed you should see some information about your Java installation. If the command line returns the information that the program could not be found, you have to install Java. A Google search for "How to install Java on ..." should result in helpful links. Replace "Windows" with your operating system, e.g. Windows, Ubuntu, Mac OS X, etc.

### 2.2. Install Eclipse

To install Eclipse download the package "Eclipse IDE for Java Developers" from the <http://www.eclipse.org/downloads> and unpack it to a local directory.

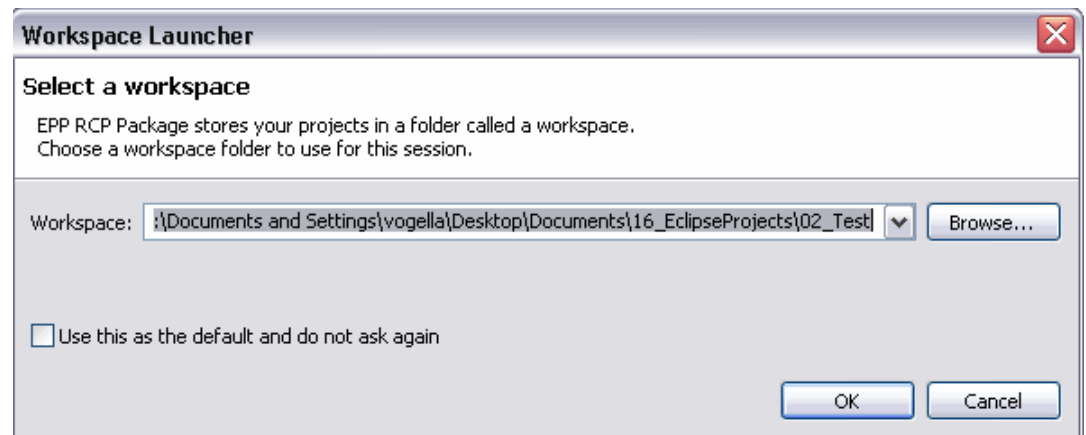
The download is a "zip" file. Most operating systems can extract zip files in their file browser. On Windows 7 via right mouse click on the file and selecting "Extract all...". If in doubt, search for "How to unzip a file on ...", again replacing "Windows" with your operating system.

Use a directory path which does not contain spaces in its name, as Eclipse sometimes fails to create the workspace if the path contains spaces.

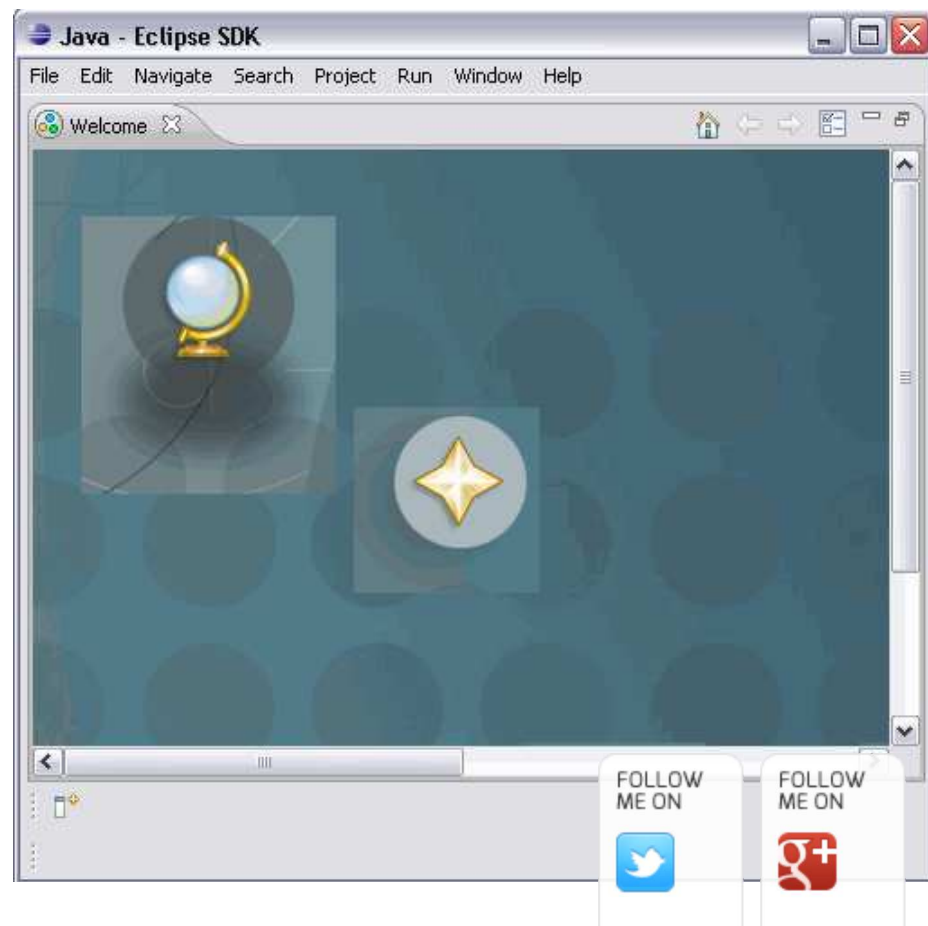


## 2.3. Start Eclipse

To start Eclipse double-click on the file `eclipse.exe` (Microsoft Windows) or `eclipse` the directory where you unpacked Eclipse. The system will prompt you for a workspace is the place where you store your Java projects (more on workspaces later). Select : and press Ok.



Eclipse will start and show the Welcome page. Close the welcome page by pressing "Welcome".



 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

[Java Persistence Tools](#) OpenJPA, Toplink, Hibernate Suppt No Lock-in, Eclipse-Based [www.myeclipseid](http://www.myeclipseid.com)

[Free Programming Courses](#) Get started in minutes! Become a Software Engineer [code.he.net](http://code.he.net)

## 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

#### 3.1.1. What is the workspace?

The workspace is the physical location (file path) you are working in. You can choose during startup of Eclipse or via the menu ( File → Switch Workspace → Others) . All source files, images and other artifacts will be stored and saved in your workspace.

#### 3.1.2. Workspace related startup parameters

Eclipse allows that certain behavior is configured via startup parameters. The following are relevant for the Workspace.

**Table 1. Workspace Startup Parameters**

Parameter	Description
-data workspace_path	Predefine the Eclipse workspace.
-showLocation	Configures Eclipse so that it shows the current workspace directory

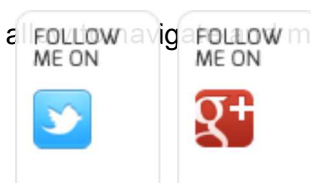
For example if you want to start Eclipse under Microsoft Windows with the workspace use the following command from the command line.

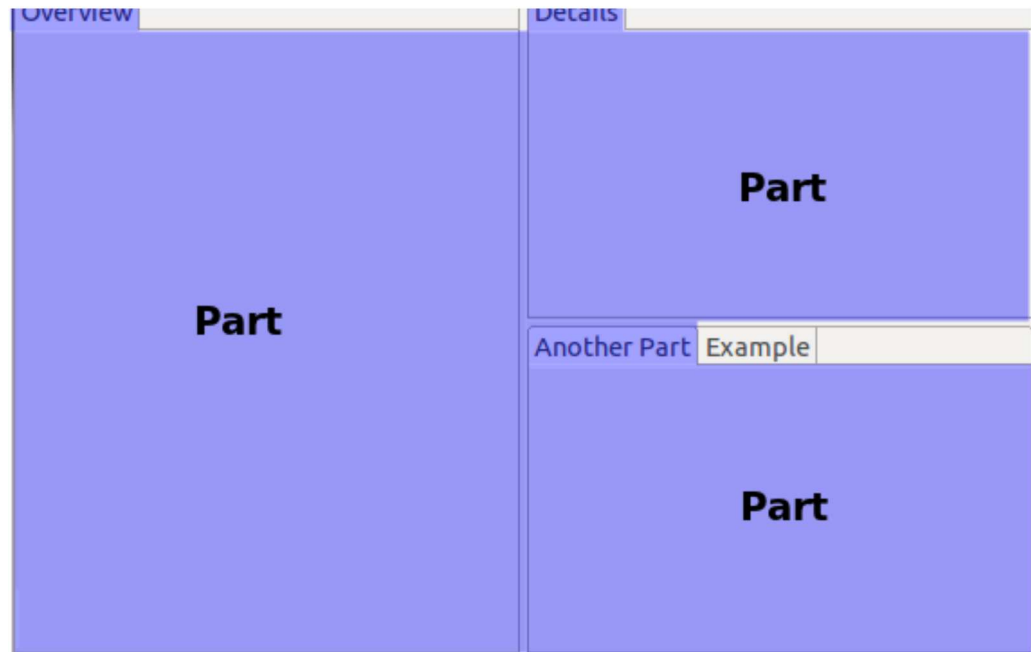
```
c:\eclipse.exe -data "c:\temp"
```

Depending on your platform you may have to put the path name into double quotes

### 3.2. Parts

Parts are user interface components which are used to display and modify data. Parts are divided into Views and Editors.





The distinction into `Views` and `Editors` is primarily not based on technical differences but on a different concept of using and arranging these `Parts`.

A `View` is typically used to work on a set of data, which might be hierarchical structure. When changed via the `View`, this change is typically directly applied to the underlying data. Sometimes, it also allows us to open an `Editor` for a selected set of the data.

An example for a `View` is the Java Package Explorer, which allows you to browse the file system. If you choose to change data in the Package Explorer, e.g. if you rename a package, the change is directly applied to the filesystem.

`Editors` are typically used to modify a single data element, e.g. a file or a data object. When changes done in an editor are applied to the data structure, the user has to explicitly select a "Save" menu entry.

`Editors` were traditionally placed in a certain area, called the "editor area". Until Eclipse 4, there was a hard limitation, it was not possible to move an `Editor` out of this area; Eclipse 4 allows `Editors` at any position in a `Perspective`.

For example, the Java Editor is used to modify Java source files. Changes to the source file are only saved once the user selects the "Save" command.

### 3.3. Perspective

A `Perspective` is a visual container for a set of `Parts`.

You can change the layout and content within a `Perspective` by opening or closing `Parts` and re-arranging them.



### 3.4. Eclipse IDE Perspectives

The Eclipse IDE ships with several default `Perspectives`.

For Java development you usually use the `Java Perspective`, but Eclipse has many `Perspectives`, e.g. `Debug`, `Git Repositories`, `CVS Repositories`.

Eclipse allows you to switch to another perspective via the menu `Window → Open Perspective → Other...`

A common problem is that you mis-configured your `Perspective`, e.g. by closing a `Perspective` to its original state via the menu `Window → Reset Perspective`.

### 3.5. Java Perspective and Package Explorer

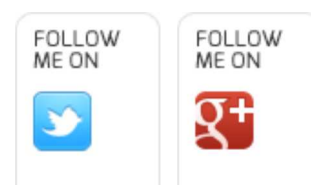
The default `Perspective` for Java development can be opened via `Window → Open Perspective → Java`.

On the left hand side, this perspective shows the "Package Explorer" `View`, which shows Java projects and to select the components you want to work on via double-click.

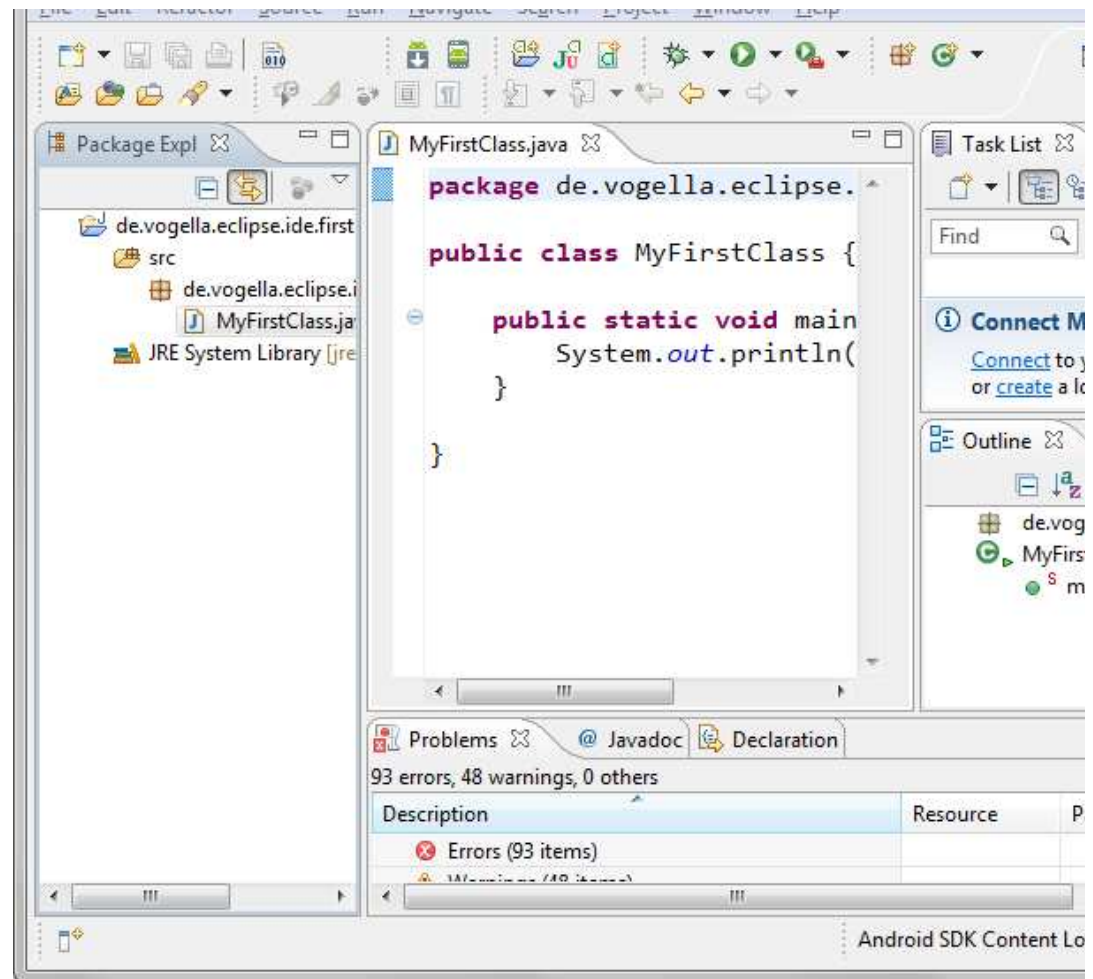
For example to open a Java source file, open the tree under `src`, select the corresponding file and double-click it. This will open the file in an `Editor`.

The following picture shows the Eclipse IDE in its standard Java perspective. The "Package Explorer" is on the left. In the middle you have the open `Editor` for a Java source file. If several files are open, they would be stacked in the same place and you could switch between them via the next `Editor`. All editors share the same part of the Eclipse IDE; this part is called the `Editor Area`.

To the right and below the editor area you find more `Views` which were considered by the developer of the perspective. For example the "Console" `view` shows the output of executed statements in your code.



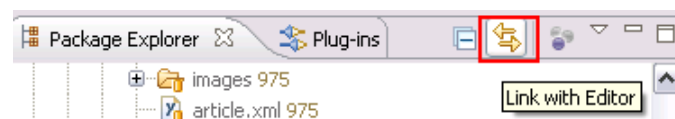




### 3.6. Linking the package explorer with the code editor

The Package Explorer allows displaying the associated file from the currently selected example if you are working on `Foo.java` and you change in the editor to `Var.java` the file will be selected in the "Package explorer" view.

To activate this behavior, press the button "Link with Editor" in the "Package explorer



### 3.7. Problems view

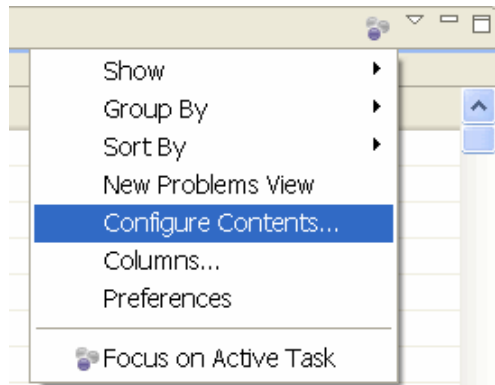
Sooner or later you will run into problems with your code or your project setup. To view your project you can use the "Problems" view which is part of the standard Java PE. If it is closed you can open it via Windows → Show View → Problems.

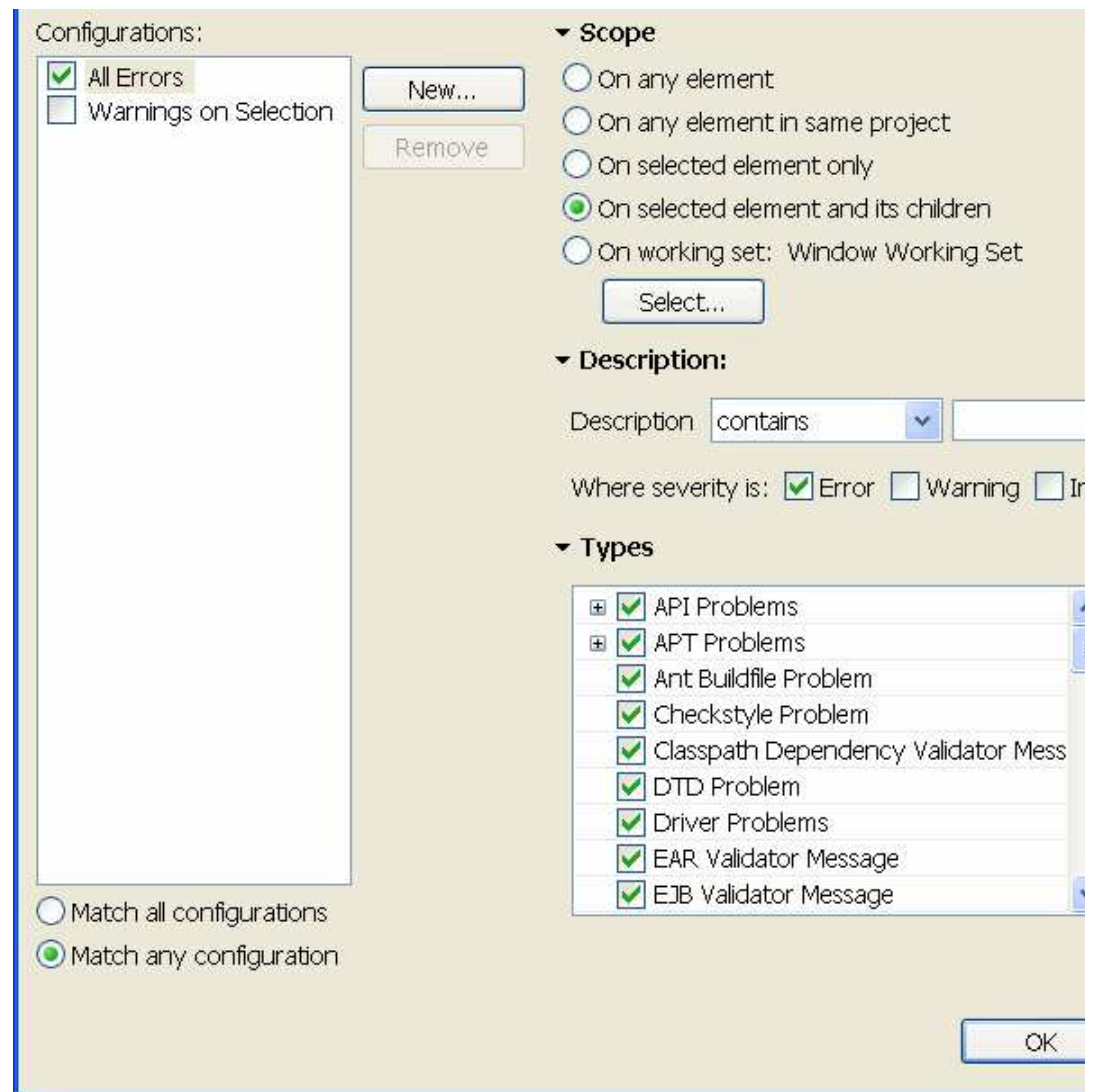




Errors (199 items)				
	Bundle 'RfcCommon' cannot be resolved	MANIFEST.MF	de.vogella...	line
	Bundle 'TMRfcConnector' cannot be resolved	MANIFEST.MF	de.vogella...	line
	LaneSelection cannot be resolved to a	ILaneDao.java	de.vogella...	line
	The import selection cannot be resolved	ILaneDao.java	de.vogella...	line
	LaneSelection cannot be resolved to a	LaneDownloadDao.java	de.vogella...	line

You can configure the content of the "Problems" view. For example, to display the currently selected project, select "Configure Contents" and set the Scope to "On an same project".



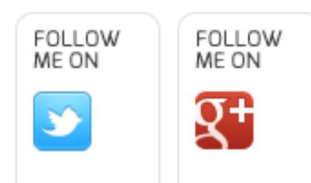


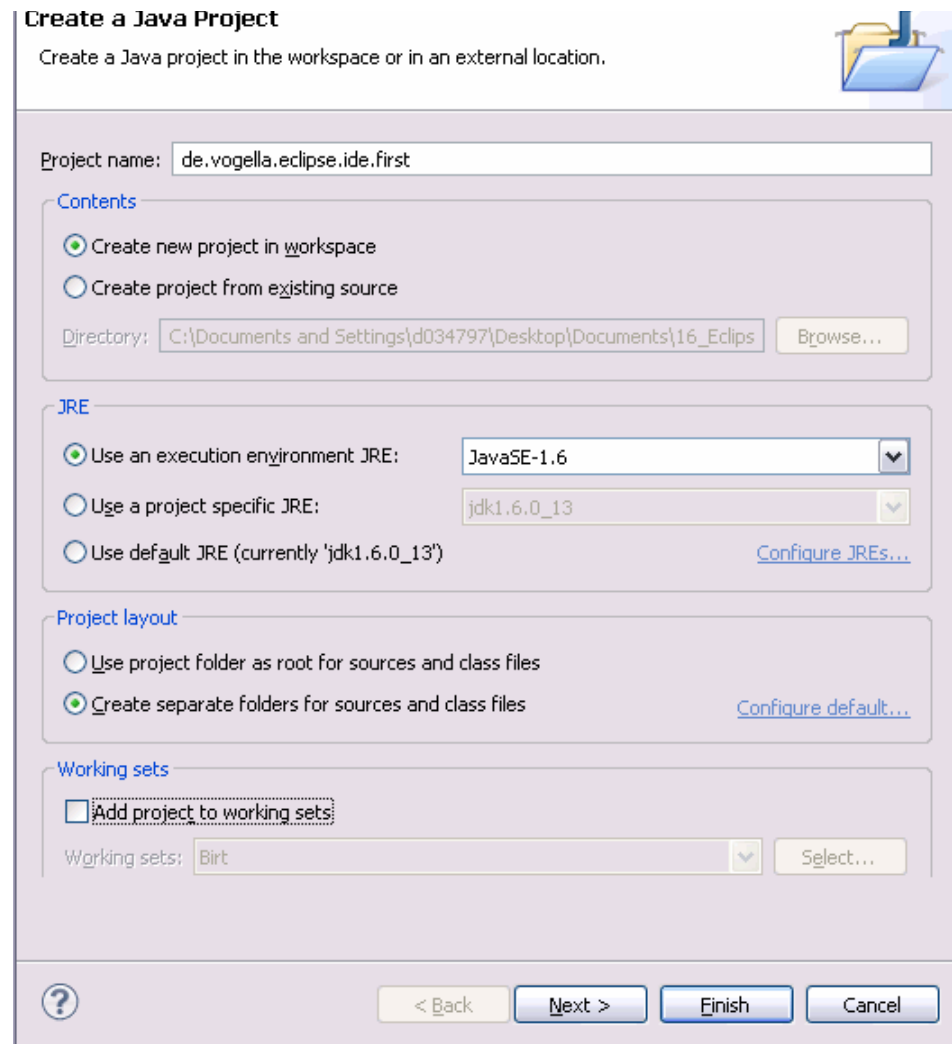
## 4. Create your first Java program

The following describes how to create a minimal Java program using Eclipse. It is tradition in the programming world to create a small program which writes "Hello World" to the console. We follow this tradition and will write "Hello Eclipse!" to the console.

### 4.1. Create project

Select from the menu File → New → Java project. Enter `de.vogella.eclipse.ide` as the project name. Select the flag "Create separate folders for sources and class files".



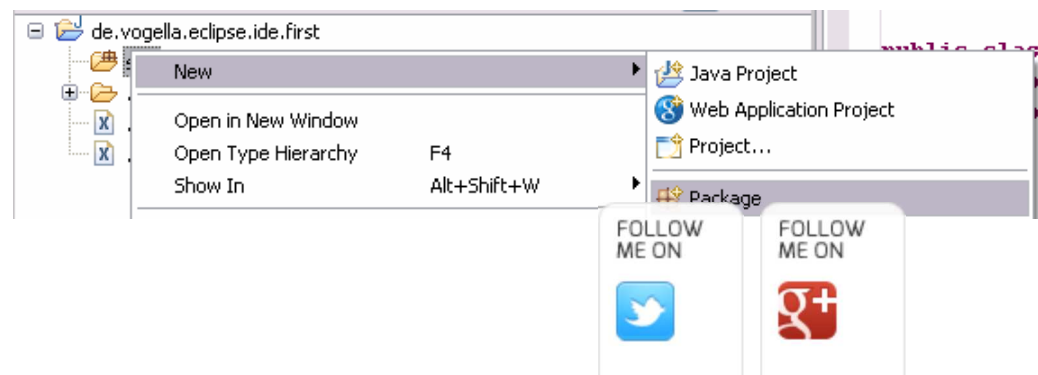


Press finish to create the project. A new project is created and displayed as a folder `de.vogella.eclipse.ide.first` and explore the content of this folder.

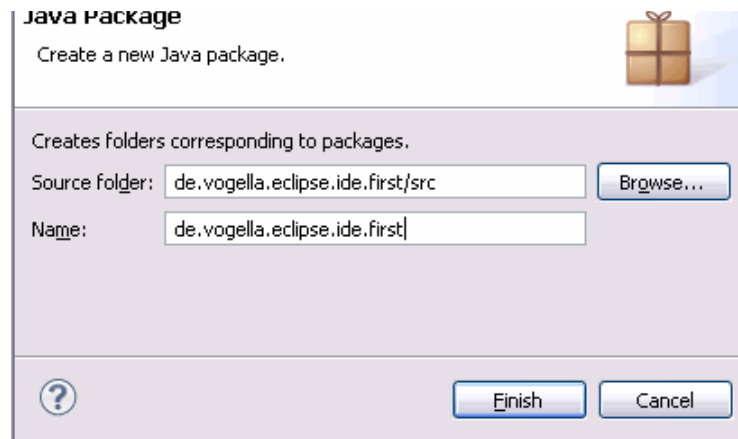
## 4.2. Create package

Create a new package. A good convention is to use the same name for the top package. Create therefore the package `de.vogella.eclipse.ide.first`.

Select the folder `src`, right click on it and select New → Package.

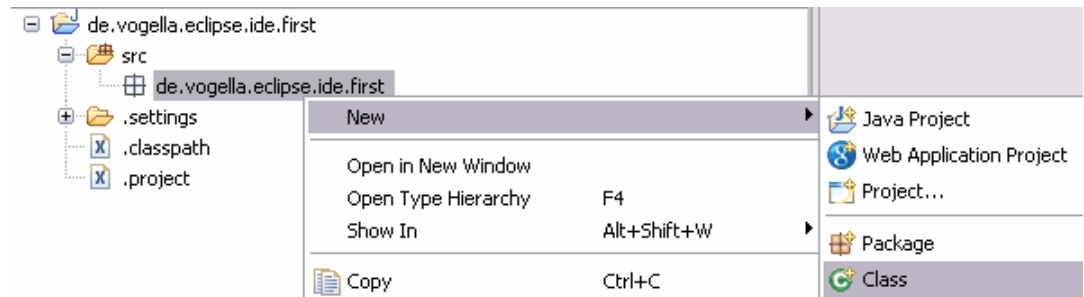


 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

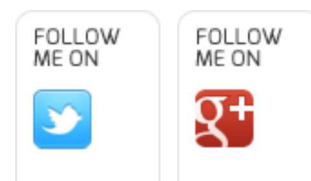


### 4.3. Create Java class

We will now create a Java class. Right click on your package and select New → Cla



Enter `MyFirstClass` as the class name and select the flag "public static void main



**Java Class**  
Create a new Java class.

Source folder:  [Browse...](#)

Package:  [Browse...](#)

☐ Enclosing type:  [Browse...](#)

Name:

Modifiers: ☒ public ☐ default ☐ private ☐ protected  
☐ abstract ☐ final ☐ static

Superclass:  [Browse...](#)

Interfaces:  [Add...](#)  
[Remove](#)

Which method stubs would you like to create?

☒ `public static void main(String[] args)`

☐ Constructors from superclass

☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))

☐ Generate comments

[?](#) [Finish](#) [Cancel](#)

This creates a new file and opens an `Editor` to edit the source code of this file. We code.

```
package de.vogella.eclipse.ide.first;

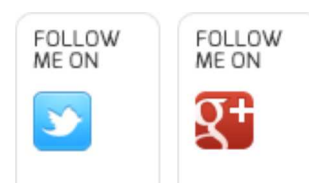
public class MyFirstClass {




    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
    }

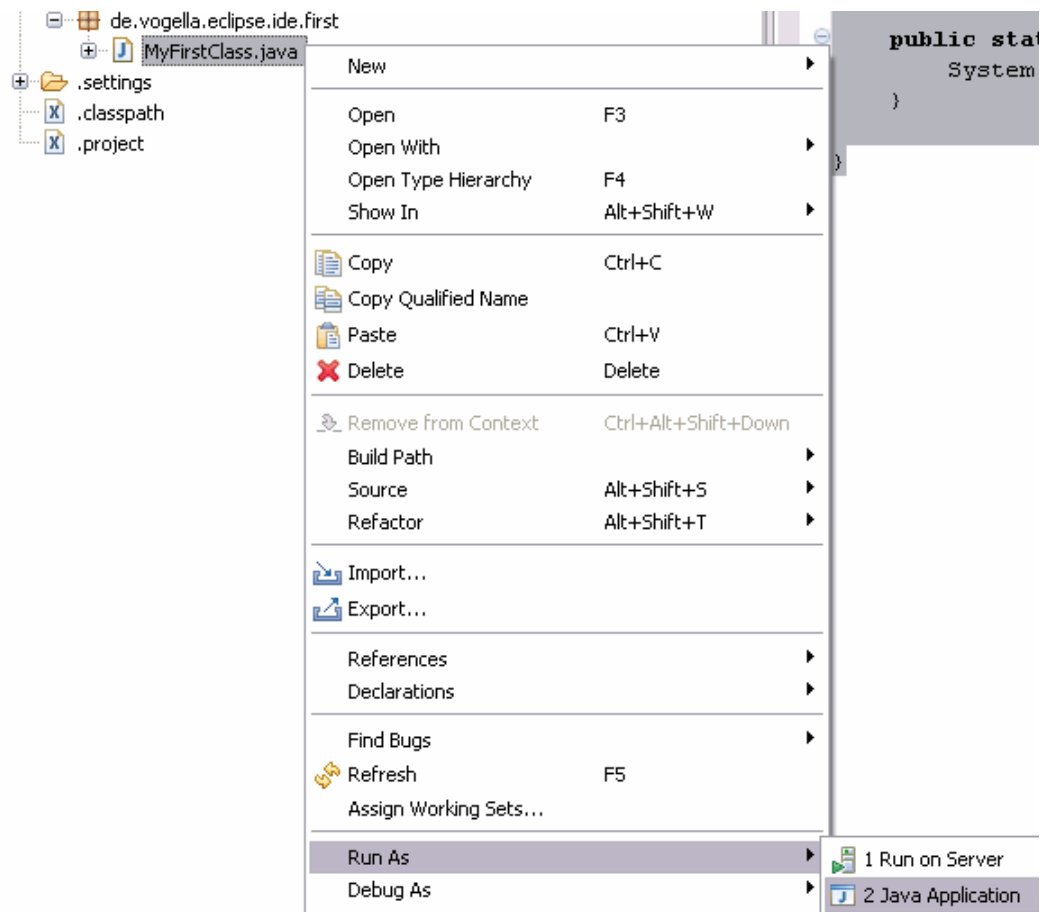
}
```

#### 4.4. Run your project in Eclipse

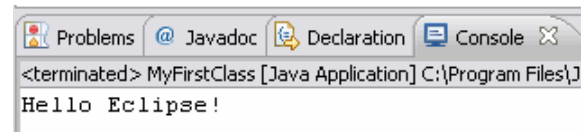
Now run your code. Right click on your Java class and select Run-as → Java applic



 Home  Tutorials  Trainings  Books  Connect



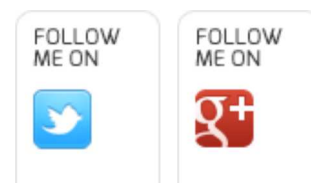
Finished! You should see the output in the console.




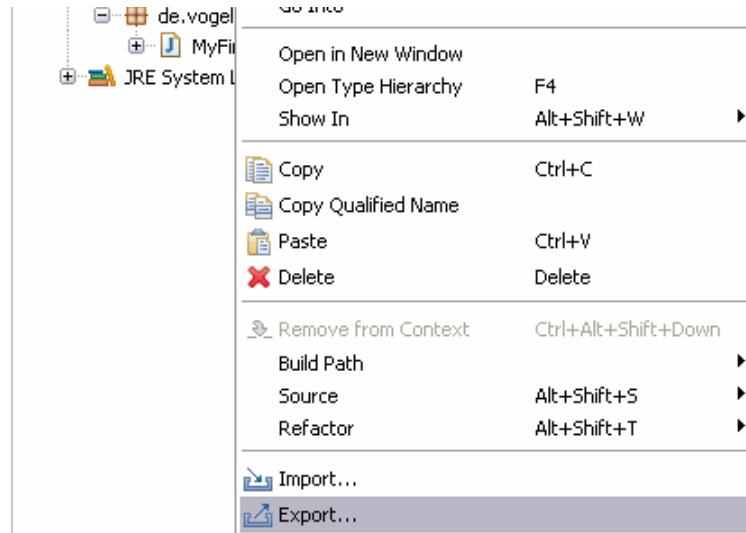
#### 4.5. Prepare to run program outside Eclipse (create jar file)

To run your Java program outside of Eclipse you need to export it as a jar file. A jar is a distribution format for Java applications.

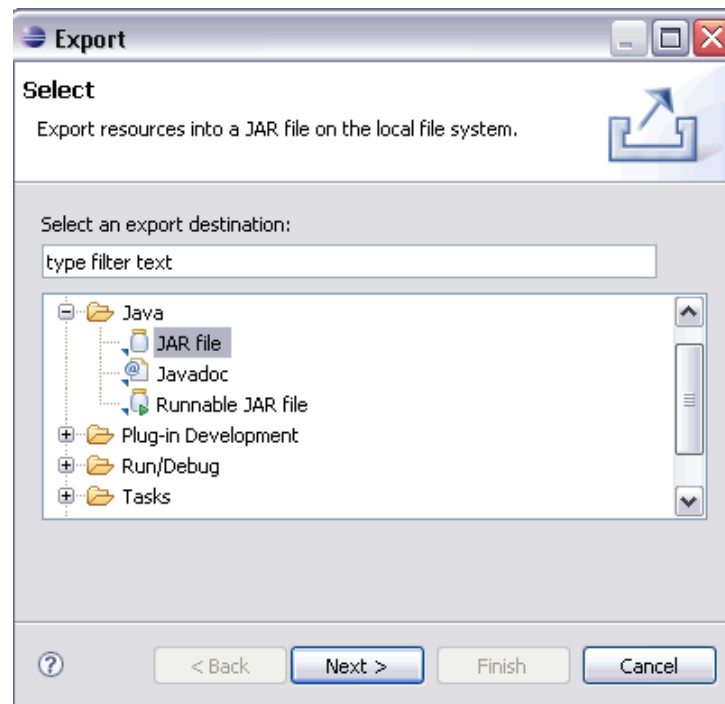
Select your project, right click on it and select **Export**.



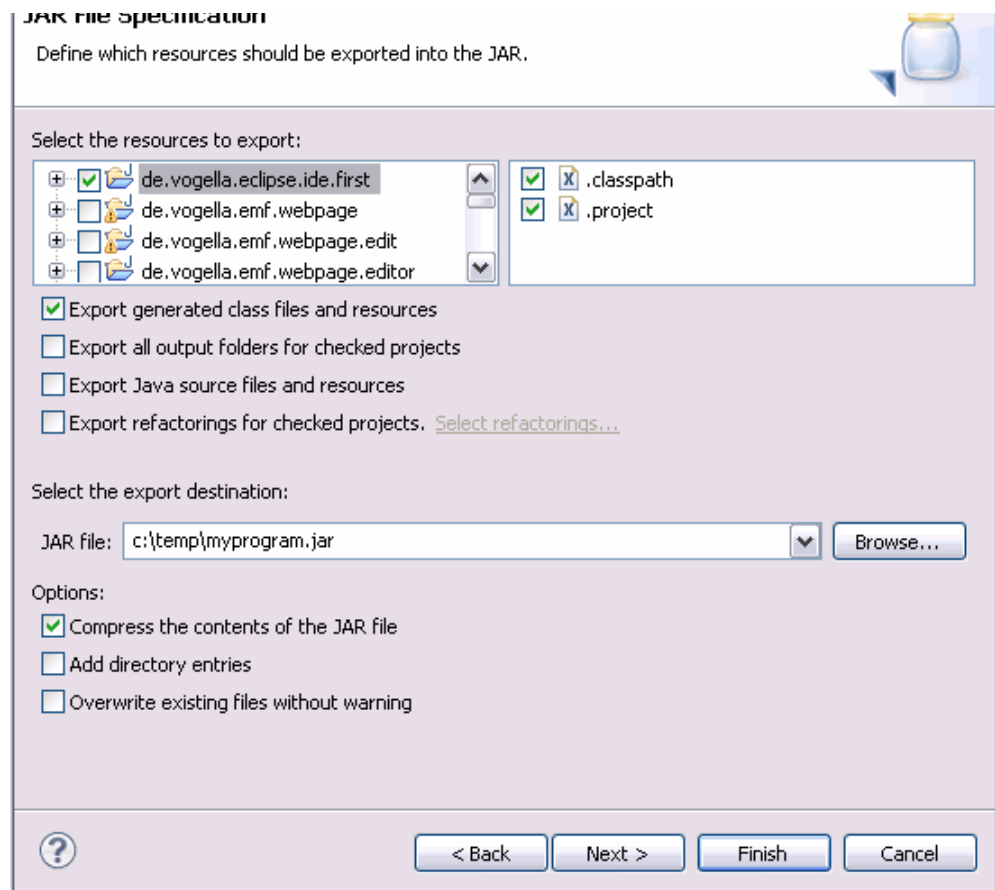
 Home  Tutorials  Trainings  Books  Connect



Select JAR file, select next. Select your project and maintain the export destination : jar file. I named it `myprogram.jar`.







Press finish. This creates a jar file in your selected output directory.

## 4.6. Run your program outside Eclipse

Open a command shell, e.g. under Microsoft Windows select Start → Run and type enter. This should open a console.

Switch to your output directory, by typing `cd path`. For example if your jar is located at `c:\temp`.

To run this program you need to include the jar file in your `classpath`. The `classpath` Java classes are available to the Java runtime. You can add a `jar` file to the `classpath` option.

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

If you type the command from above and are in the correct directory you should see output on the console.

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



Congratulations! You created your first Java project, a package, a Java class and you

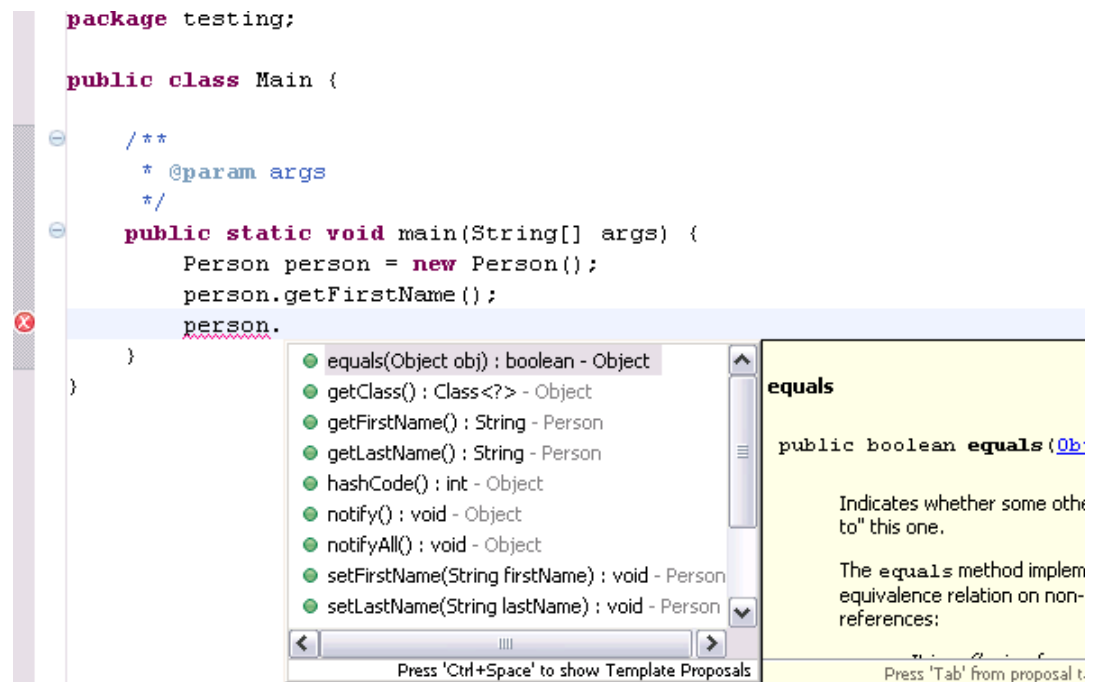
## 5. Content Assist, Quick Fix and Class Navigation

### 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` in the editor of a Java source file and then press **CTRL+Space**.  
`syso` with `System.out.println(" ")`.

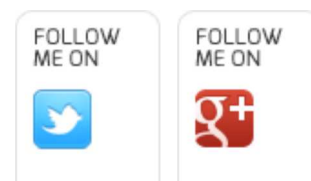
If you have a reference to an object, for example the object `person` of the type `Person` it's methods, type `person.` and press **CTRL+Space**.

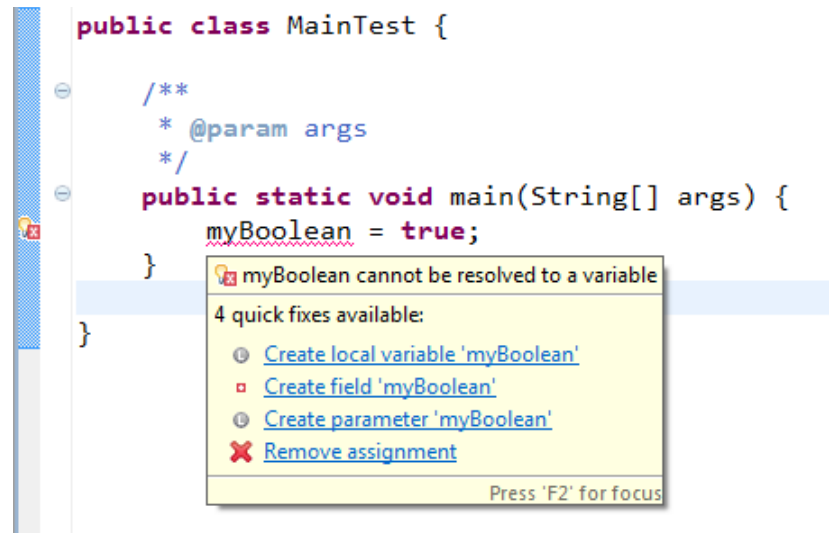


### 5.2. Quick Fix

Whenever Eclipse detects a problem, it will underline the problematic text in the editor. Select the underlined text and press **CTRL+1** to see proposals how to solve this problem.

For example type `myBoolean = true;` If `myBoolean` is not yet defined, Eclipse will show an error. Select the variable and press **CTRL+1**, Eclipse will suggest creating a field or



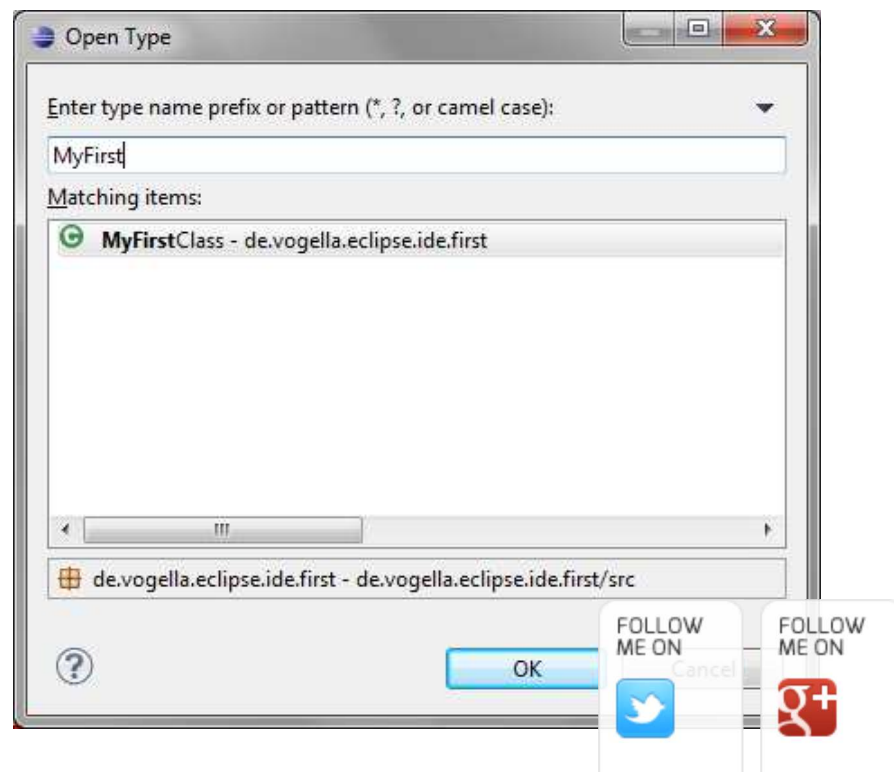





Quick Fix is extremely powerful. It allows you to create new local variables and field: methods and new classes. I can put try-catch statements around your exceptions. I statement to a variable and much more.

## 6. Opening a class

You can navigate between the classes in your project via the "Package Explorer" v.i

In addition you can open any class via positioning the cursor on the class in an edit. Alternatively, you can press **CTRL+Shift+T**. This will show a dialog in which you can name to open it.



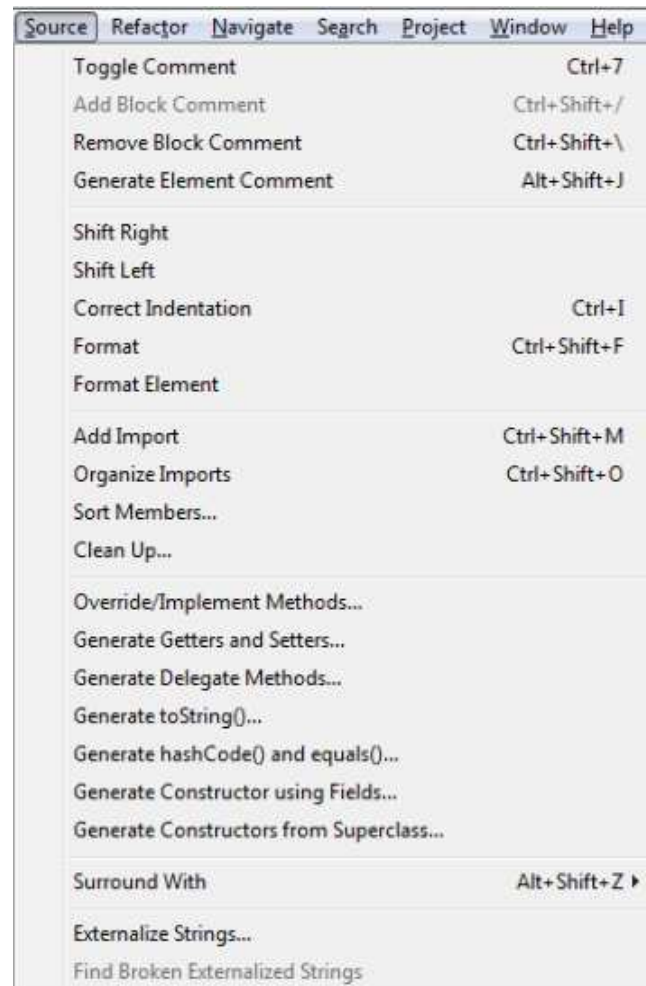
 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

## 1. Generating Code

Eclipse has several possibilities to generate code for you. This can save significant development time.

For example Eclipse can override methods from superclasses and generate the `toString()`, `hashCode()` and `equals()` methods. It can also generate getter and setter methods for your Java class.

You can find these options in the Source menu.

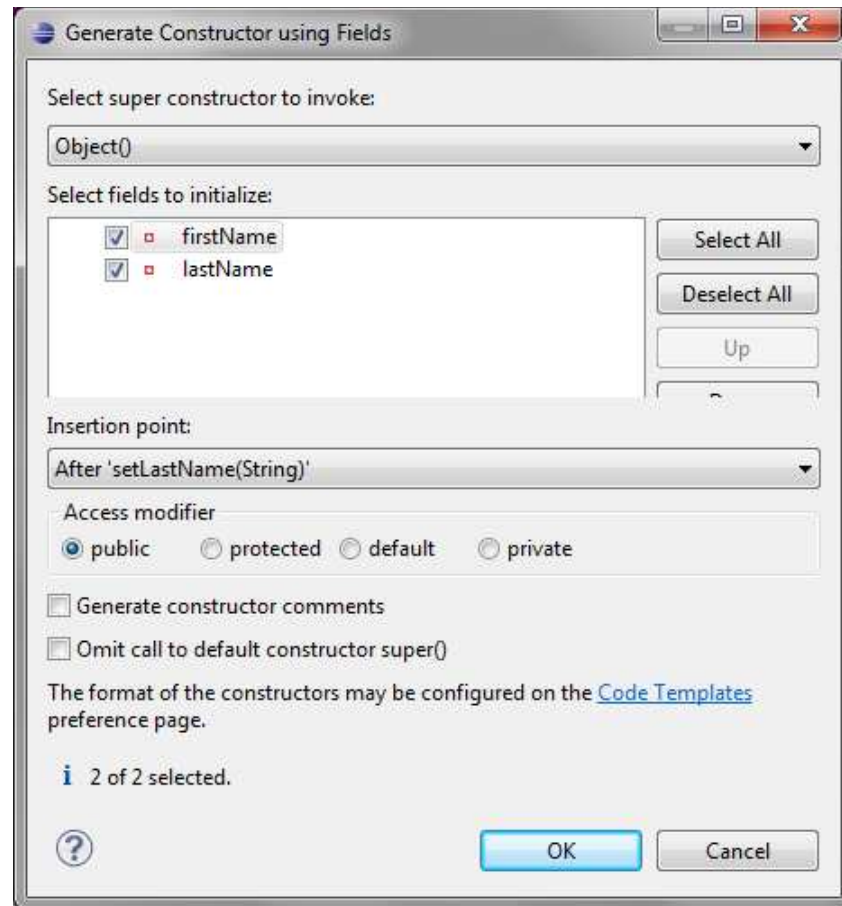


To test the source generation, create the following class in your `de.vogella.eclipse.ide` project.

```
package de.vogella.eclipse.ide.first;

public class Person {
    private String firstName;
    private String lastName;
}
```





Select Source → Generate Getter and Setter, select again both your fields and pres

Select Source → Generate toString(), mark again both fields and press "Ok".

You created the following class:

```
package de.vogella.eclipse.ide.first;

public class Person {
    private String firstName;
    private String lastName;


    public Person(String firstName, String lastName) {
        super();
        this.firstName = firstName;
        this.lastName = lastName;
    }

    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    public String getLastName() {
        return lastName;
    }
}
```



 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

```
        this.lastName = lastName,
    }

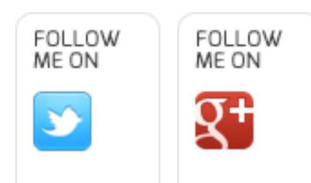
    @Override
    public String toString() {
        return "Person [firstName=" + firstName + ", lastName=" + lastName
        + "]\n";
    }
}
```

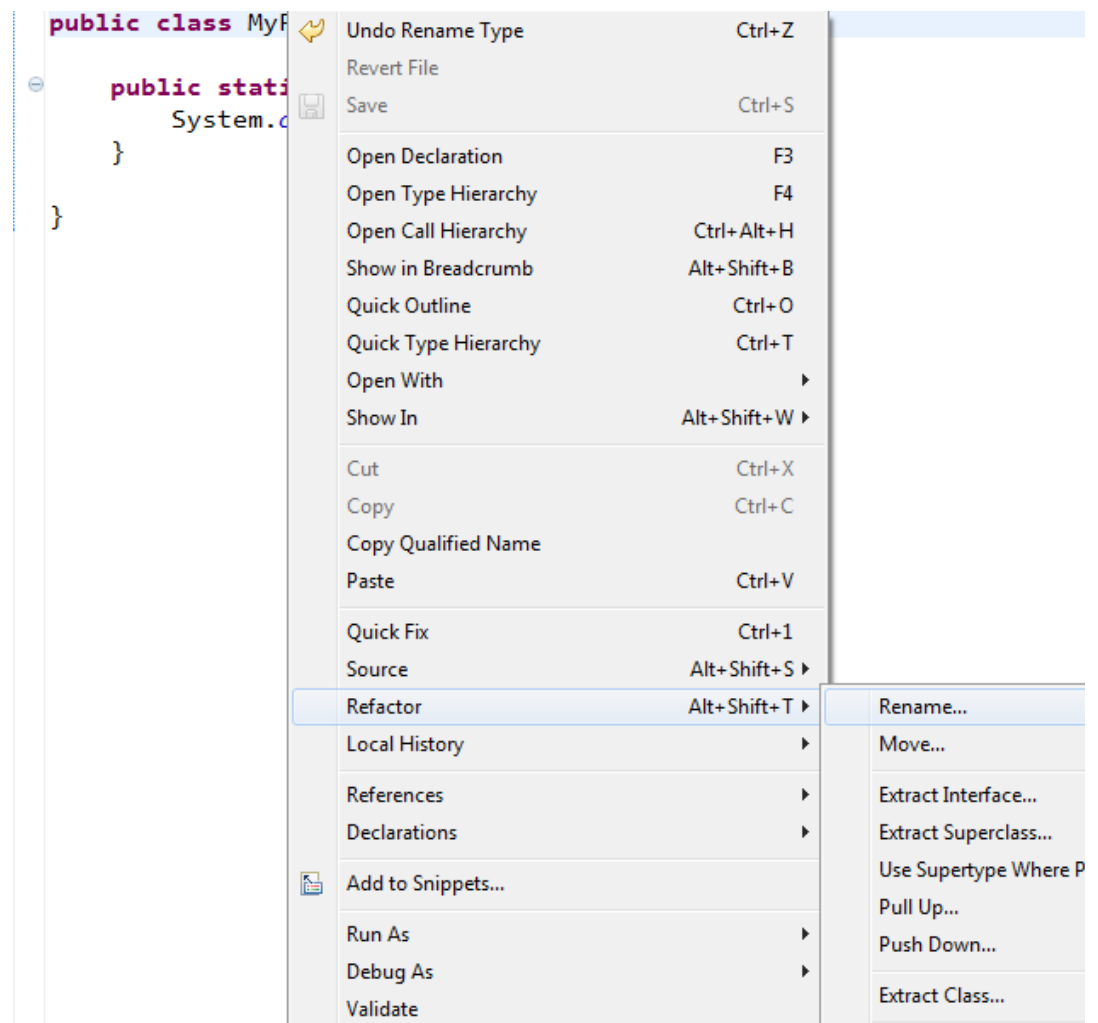
## 8. Refactoring

Refactoring is the process of restructuring the code without changing his behavior. In renaming a Java class or method is a refactoring activity.

Eclipse supports simple refactoring activities, for example renaming or moving. For example, to rename a class, select your class, right click on it and select Refactor → Rename to rename your class. Eclipse will make sure that all calls in your Workspace to your class or method are renamed.

The following shows a screenshot for calling the "Rename" refactoring on a class.





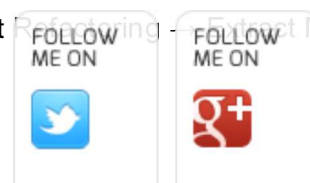
For the next examples change the code of "MyFirstClass.java" to the following.

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

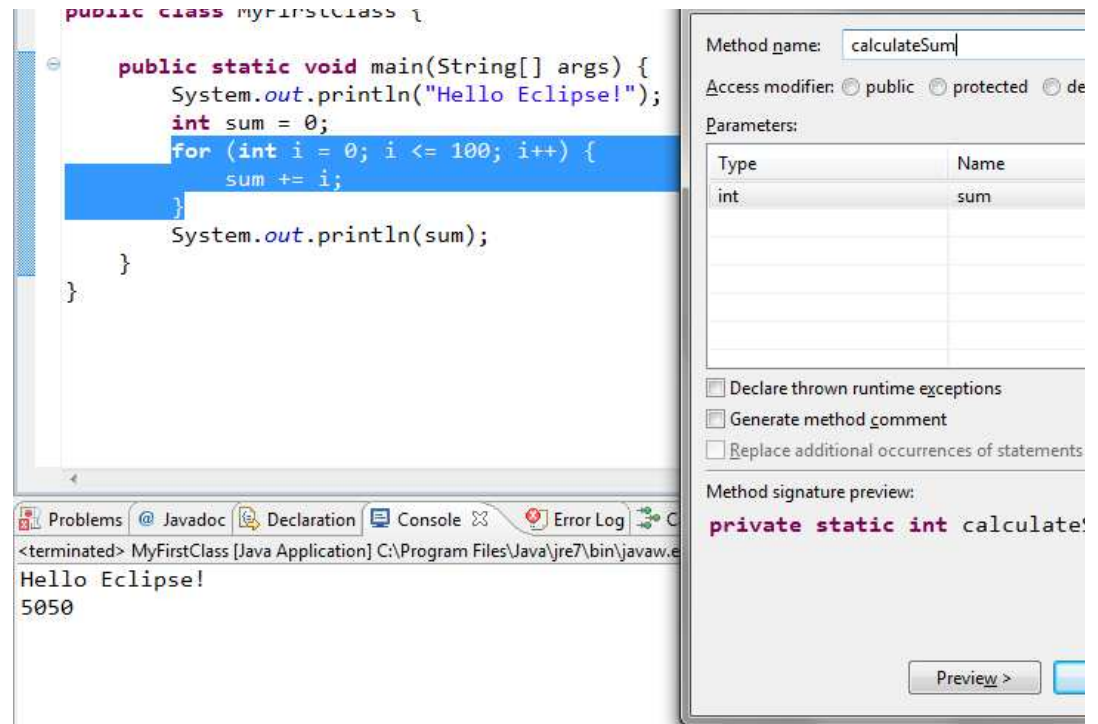
    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        System.out.println(sum);
    }
}
```

Another useful refactoring is to mark code and create a method from the selected code. In the example, we mark the coding of the "for" loop, right click and select "Extract Method...". Use "calculateSum" as the name of the new method.





 Home
  Tutorials
  Trainings
  Books
  Connect



The resulting class should look like the following.

```

package de.vogella.eclipse.ide.first;

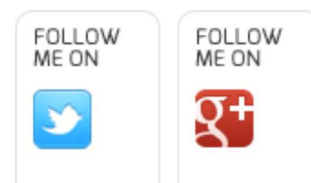
public class MyFirstClass {

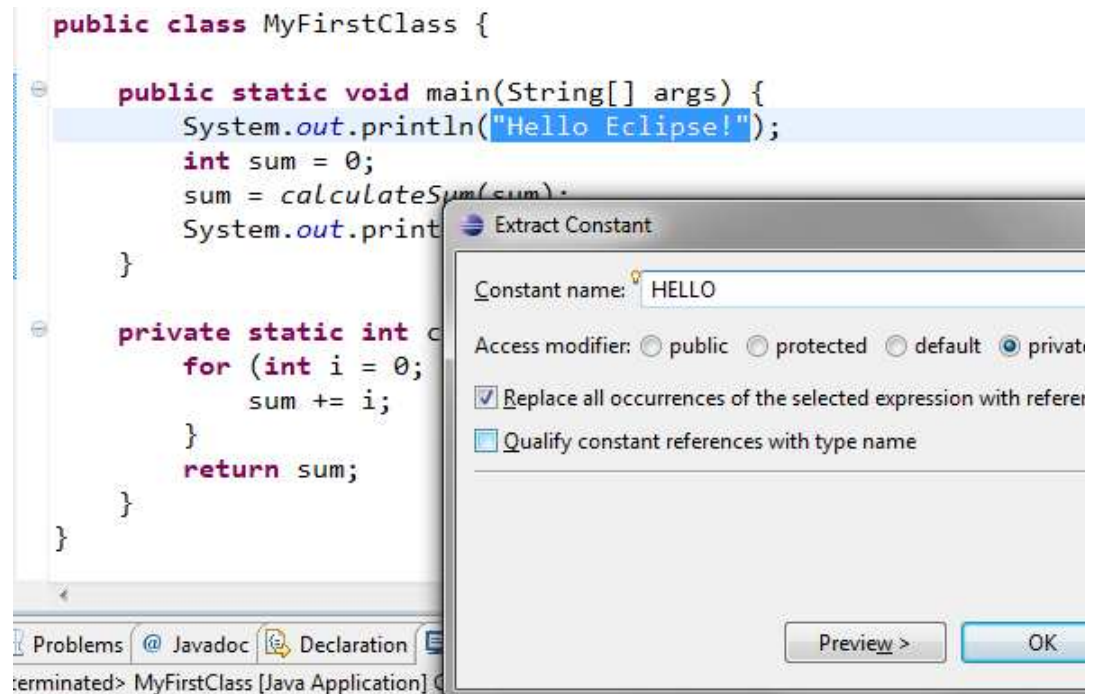
    public static void main(String[] args) {
        System.out.println("Hello Eclipse!");
        int sum = 0;
        sum = calculateSum(sum);
        System.out.println(sum);
    }

    private static int calculateSum(int sum) {
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}

```

You can also extract strings and create constants from them. Mark for this example click on it and select Refactor → Extract Constant. Name your new constant "HELLC





The resulting class should look like the following.

```
package de.vogella.eclipse.ide.first;

public class MyFirstClass {

    private static final String HELLO = "Hello Eclipse!";

    public static void main(String[] args) {
        System.out.println(HELLO);
        int sum = 0;
        sum = calculateSum(sum);
        System.out.println(sum);
    }

    private static int calculateSum(int sum) {
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}
```

Eclipse has much more refactorings, in most cases you should get an idea of the proper naming of the refactoring operation.

## 9. Eclipse Shortcuts

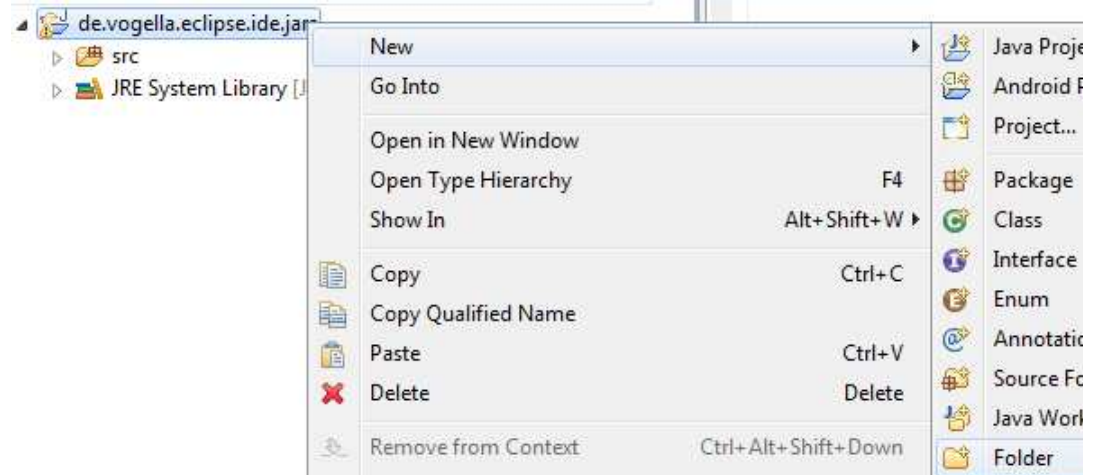
Eclipse provides a lot of shortcuts to work efficiently with the IDE. For a list of the most useful shortcuts please see [Eclipse Shortcuts](#)



## 10.1. Adding a library (.jar ) to your project

The following describes how to add Java libraries to your project. Java libraries are .jar files. It assumes that you have a jar file available; if not feel free to skip this step.

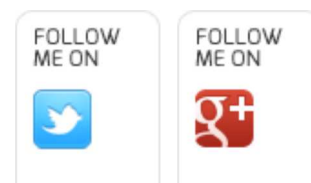
Create a new Java project `de.vogella.eclipse.ide.jars`. Then, create a new folder right clicking on your project and selecting **New → Folder**.

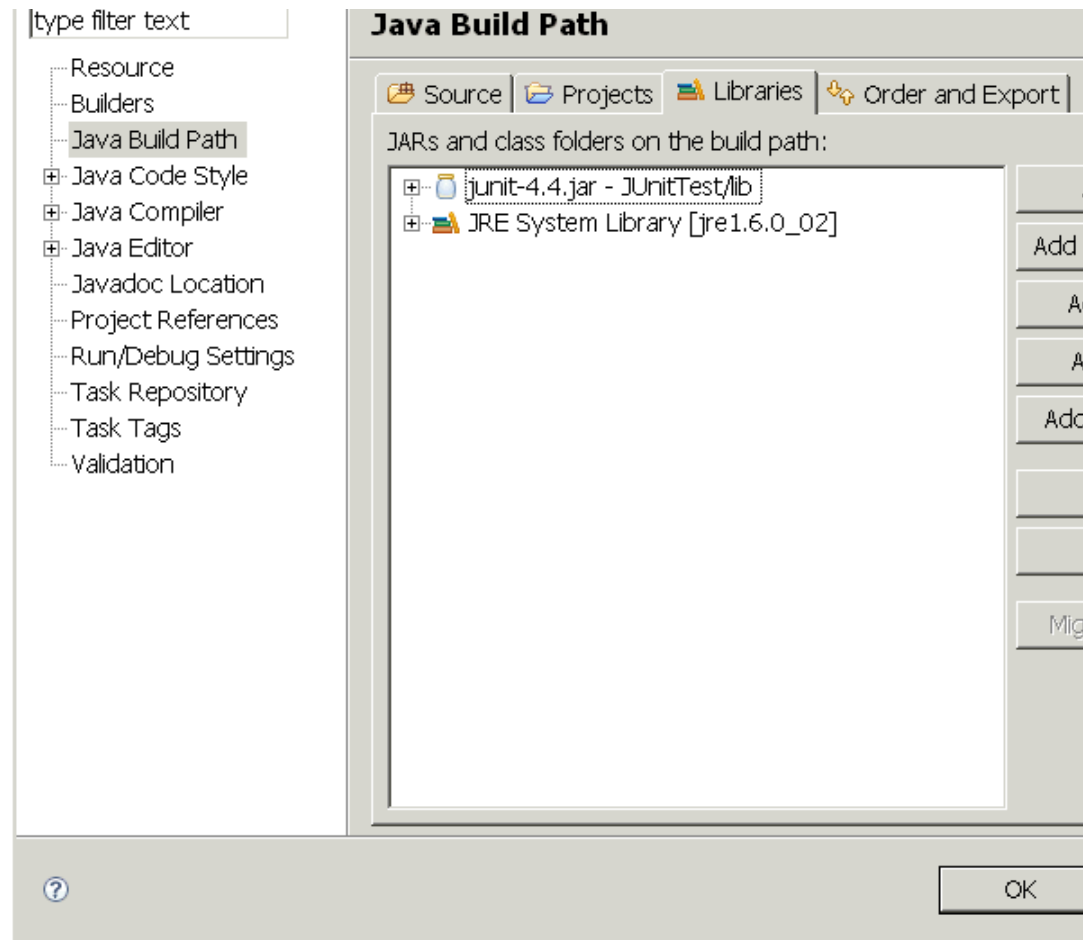


From the menu select **File → Import → General → File System**. Select your jar and the project as target. Alternatively, just copy and paste your `jar` file into the "lib" folder.

Right click on your project and select **Properties**. Under **Java Build Path → Libraries** click "Add JARs".

The following example shows how the result would look like, if the `junit-4.4.jar` had been added to the project.





Afterwards you can use the classes contained in the `jar` file in your Java source code.

## 10.2. Attach source code to a Java library

As said earlier you can open any class via positioning the cursor on the class in an editor and pressing **F3**. Alternatively, you can press **CTRL+Shift+T**. This will show a dialog in which you can enter the name to open it.

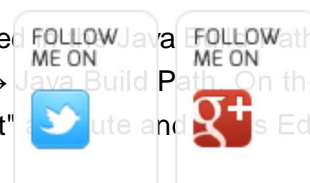
If the source code is not available, the editor will show the decompiled byte-code of the class.

This happens if you open a class from Java library and the source for this `.jar` file is not attached. The same happens if you open a class from the standard Java library without attaching the source.

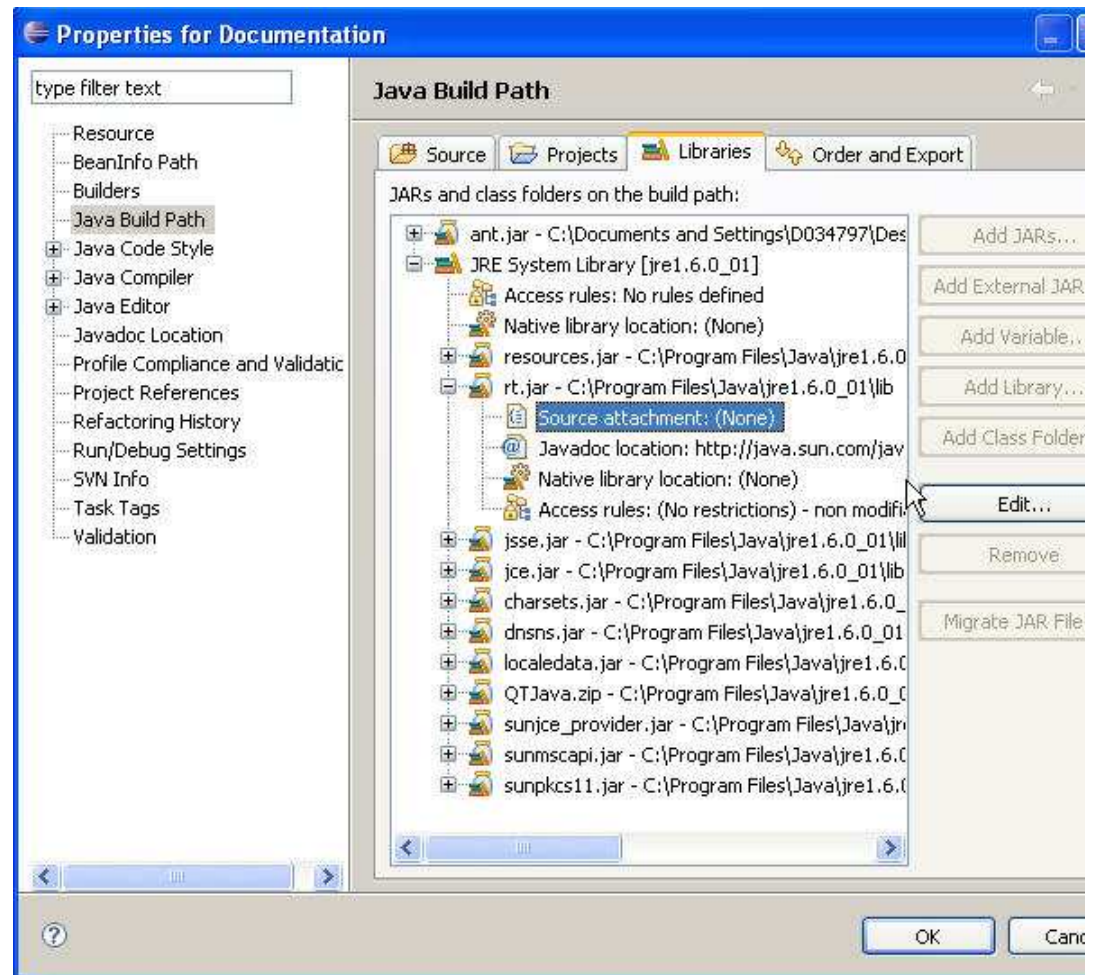
To browse the source of a type contained in a library (i.e. `.jar` file), you can attach a source folder to that library. Afterwards the editor will show the source instead of the decompiled byte-code.

In addition setting the source attachment allows debugging this source code.

The Source Attachment dialog can be reached via the 'Properties' dialog of a project or library. On the 'Libraries' tab, right click on a project → Properties → Java Build Path. On the 'Libraries' tab, select the 'Source attachment' checkbox and click 'OK'.



The following shows this for the standard Java library. If you have the Java Development Kit (JDK) installed, you should find the source in the JDK installation folder. The file is typically



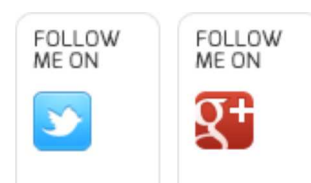
### 10.3. Add the Javadoc for a jar

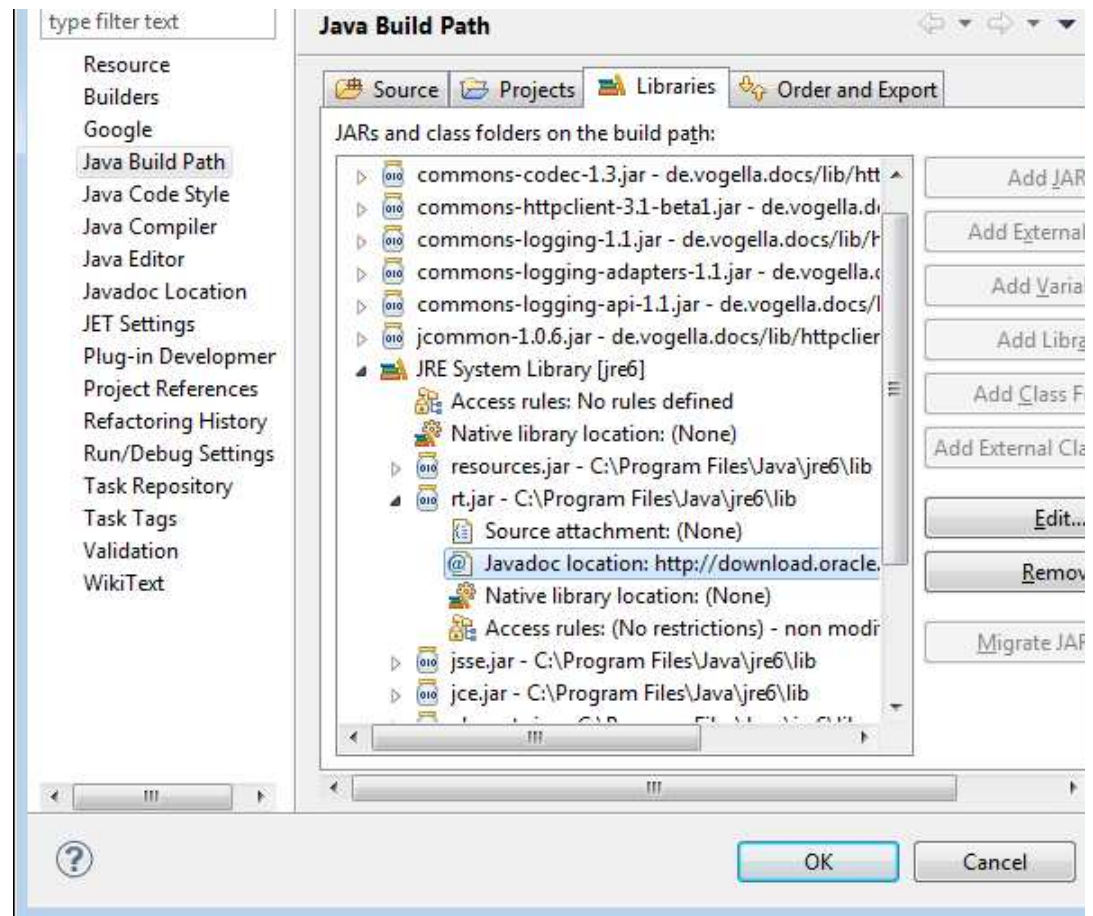
It is also possible to add Javadoc to a library which you use.

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

Open the Java Build Path page of a project via Right click on a project → Properties  
On the "Libraries" tab expand the library's node, select the "Javadoc location" attrib

Enter the location to the file which contains the Javadoc.





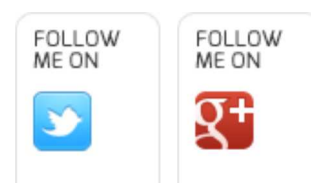
## 11. Updates and Installation of Plugins

### 11.1. Eclipse Update Manager

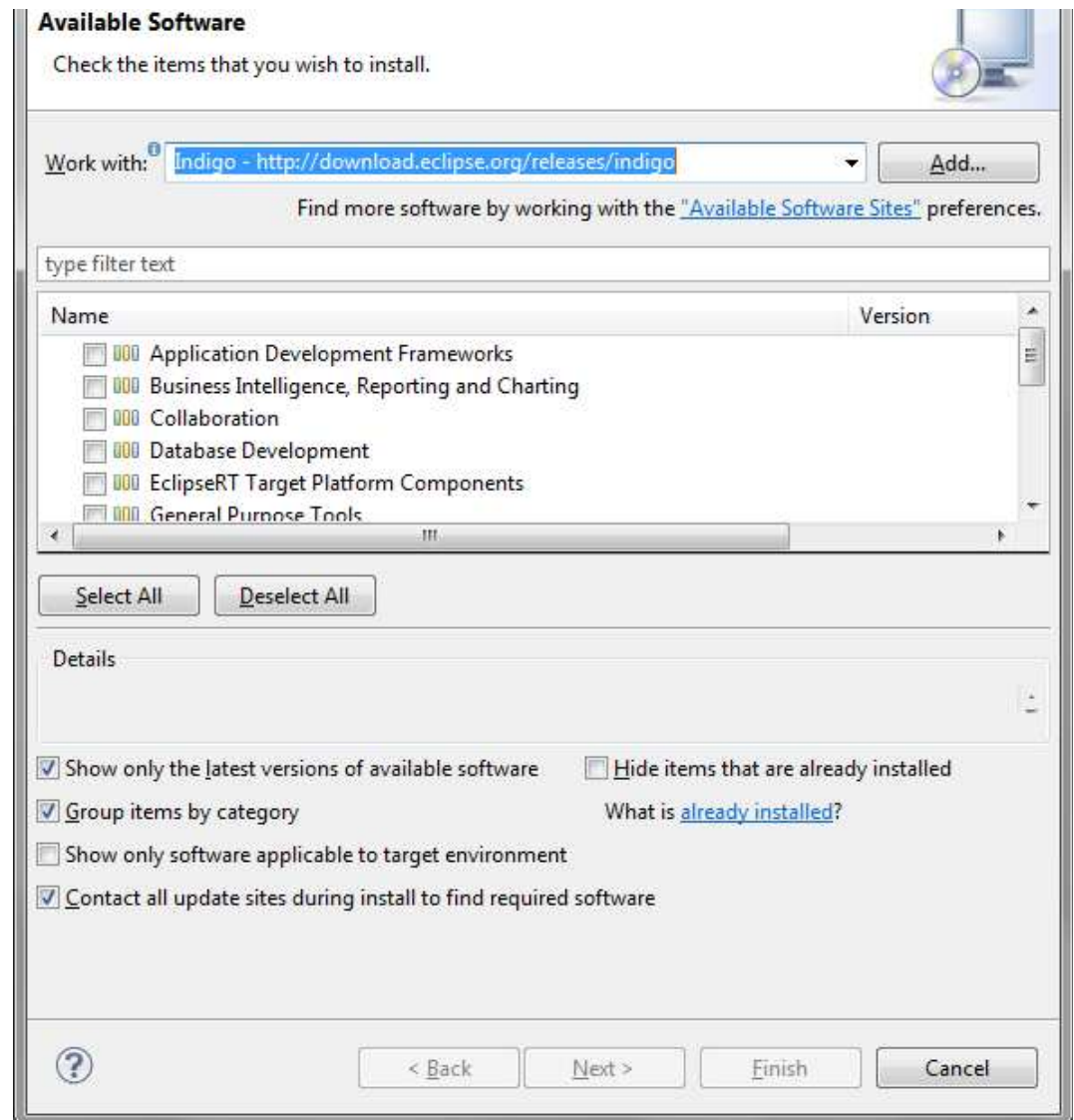
Eclipse contains an Update Manager which allows you to install and update software. Installable software components are called *features* and consists of *plug-ins*.

To update your Eclipse installation, select Help → Check for Updates. The system will check for updates for the already installed software components. If it finds updated components, you will be prompted to approve the update.

To install a new functionality, select Help → Install New Software







From the "Work with" list, select an URL from which you would like to install.

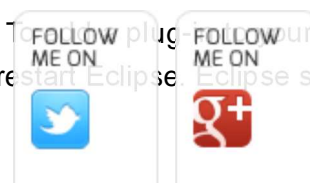
To add a new update site, press "Add" and enter the new URL as well as a name for site.

Sometimes you have to uncheck "Group items by category" because not all available items are categorized. If they are not categorized, they will not be displayed, unless the group is selected.

## 11.2. Manual installation of plug-ins (dropins folder)

If you're using a plug-in for which no Update Site is available, you can use the "dropins" folder in the Eclipse installation directory.

Plug-ins are typically distributed as .jar files. To install a plug-in manually, copy the .jar file into the Eclipse "dropins" folder and restart Eclipse. Eclipse should detect the new plug-in and install it for you.



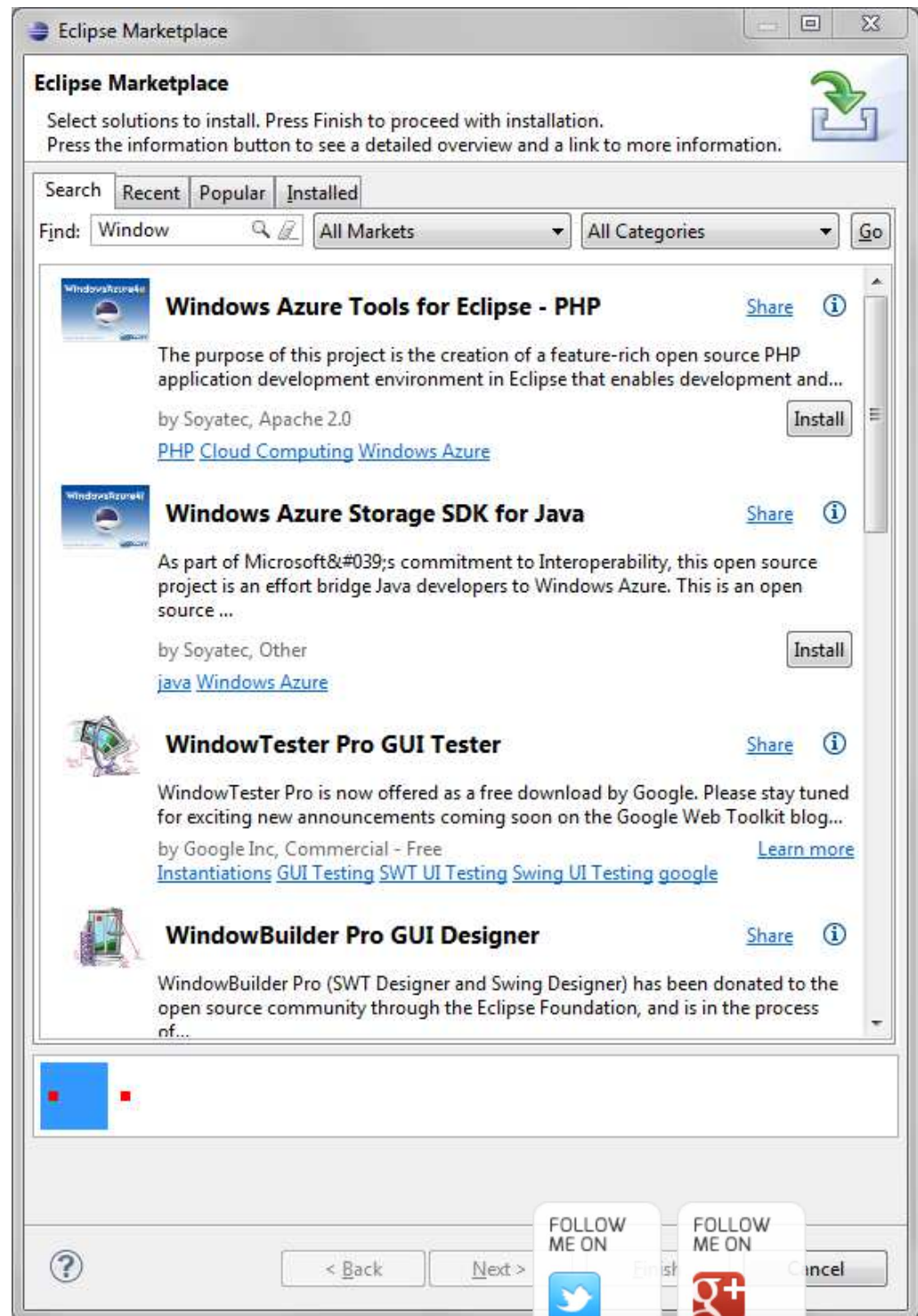


 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

Eclipse also contains a client which allows installing software components from the The advantage of this client is that you can search for components, discover popular descriptions and ratings.

Compared to the update manager you don't have to know the URL for the update si

To open the Eclipse Marketplace select Help → Eclipse Marketplace.

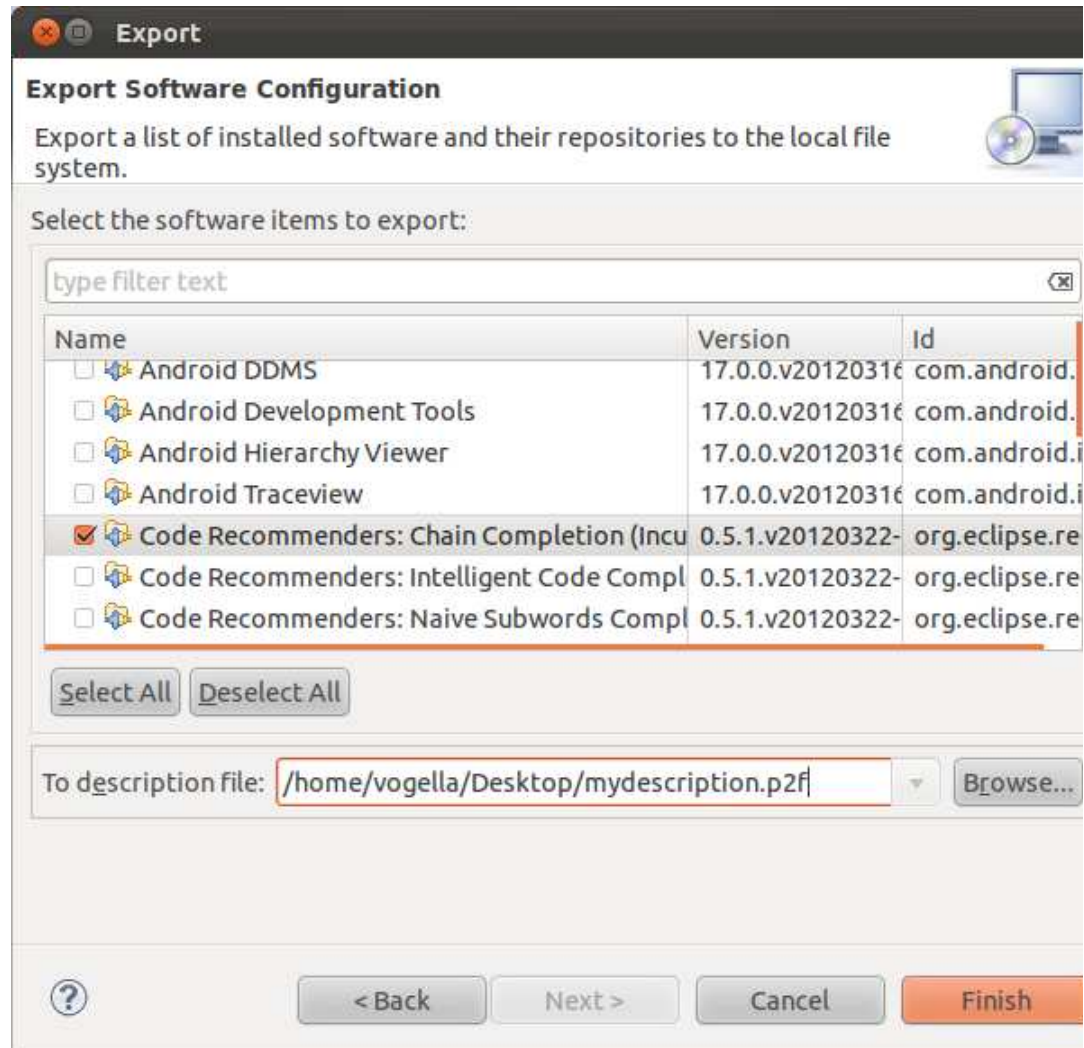


## 11.4. Share plug-in

Eclipse 4.2 also allow to export a description file which for selected Eclipse compon import this description file into their Eclipse installation and install these component

This way Eclipse installation can be kept in sync with each other.

To export a description file, select File → Export → Install → Installed Software Item the components which should be included into your description file.



To install selected components of this file in another Eclipse Installation, open it with Install → Install Software Items from File and follow the wizard.

## 11.5. Restart required?

After an update or an installation of a new sof component in Eclipse to restart Ec a good idea to restart Eclipse in this situation, otherwise some components might no





Unlimited Bandwidth  
Unlimited Mailboxes



## 12. Efficiency Settings

### 12.1. Eclipse Preferences

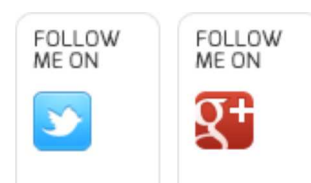
The behavior of the Eclipse IDE can be controlled via the Preference Settings. Select Preferences to open the preference settings dialog. You can use the filter box to select settings.

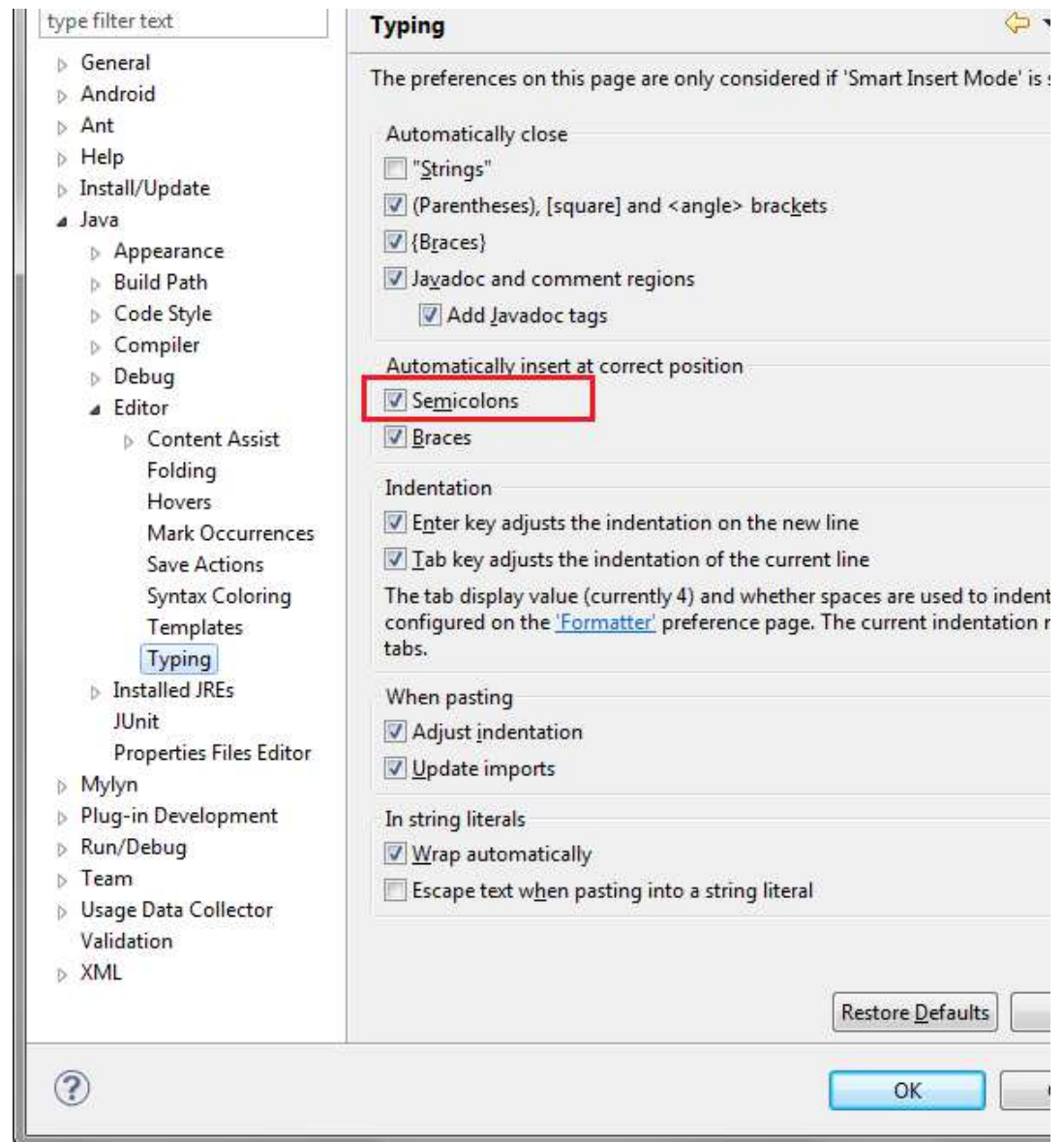
### 12.2. Automatic placement of semicolon

Eclipse can make typing more efficient by placing semicolons at the correct position

In the Preference setting select Java → Editor → Typing. In the section "Automatic placement", enable the "Semicolons" checkbox.

You can now type a semicolon in the middle of your code and Eclipse will position it at the end of the current statement.

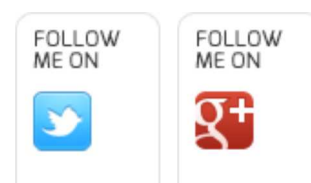


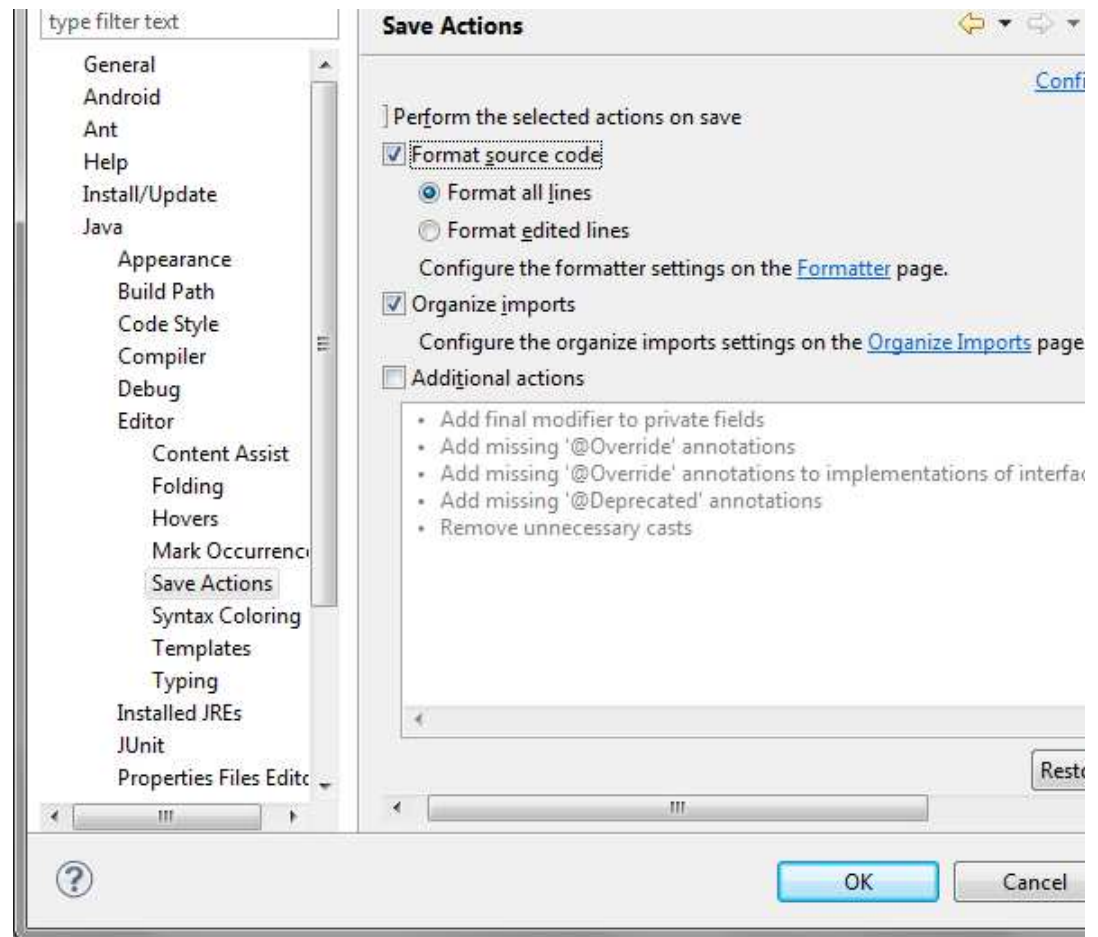


## 12.3. Imports and Source code formating

Eclipse can format your source code and organize your import statements automatic operation. This is useful as the "Save" shortcut ( **CTRL+S** ) is easy to reach.

You can find this setting under Java → Editor → Save Actions.



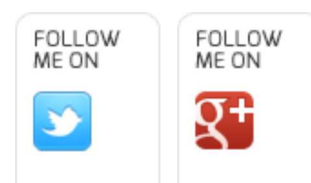


Import statements will only be automatically imported, if Eclipse finds only one valid import. If Eclipse determines more than one valid import, it will not add import statements automatically.

## 12.4. Filter import statements

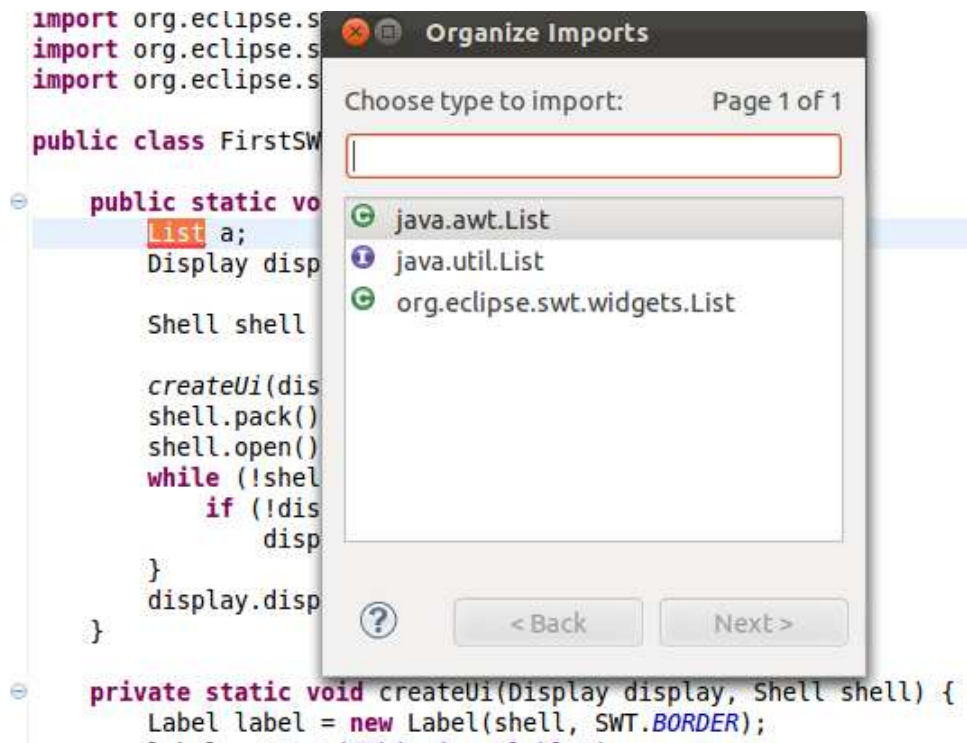
The "Organize imports" Save Action or the "Organize Imports" shortcut ( **CTRL+Sh** ) will automatically import the packages which are required. If there are several alternatives, Eclipse suggests the best one. The user has to select the right one.

The following shows the available packages for the `List` class in the "Organize Imports" dialog box.



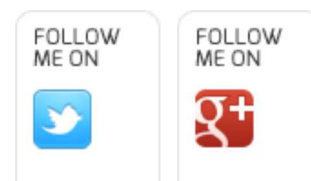


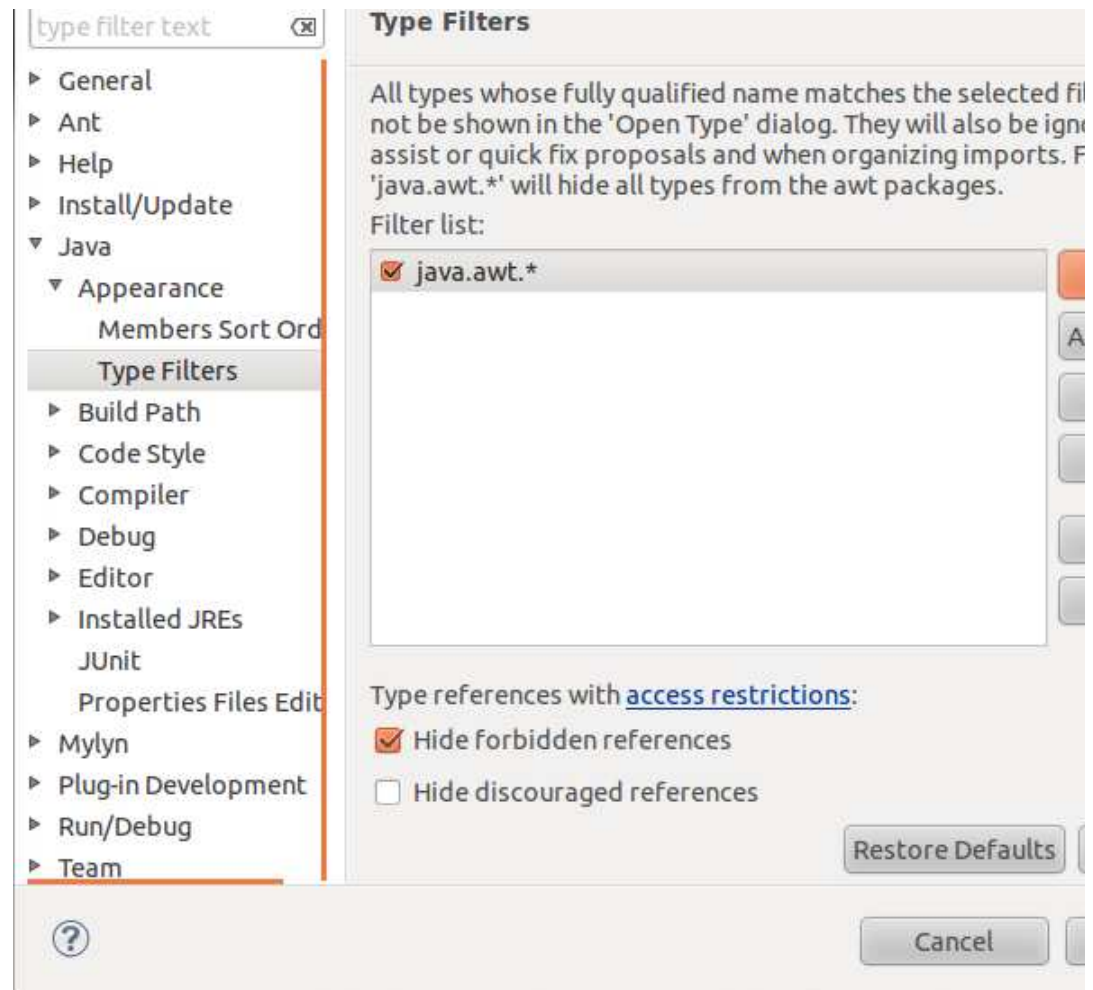
[Home](#) [Tutorials](#) [Trainings](#) [Books](#) [Connect](#)



This is annoying, if you never use certain packages. You can exclude these packages. Preferences → Java → Appearance → Type Filters

Press "Add packages" to add a specific package or "Add" to use wildcards. The following packages are added to the list to exclude all AWT packages from import.



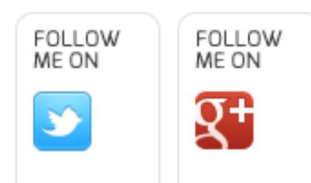


## 12.5. Templates

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

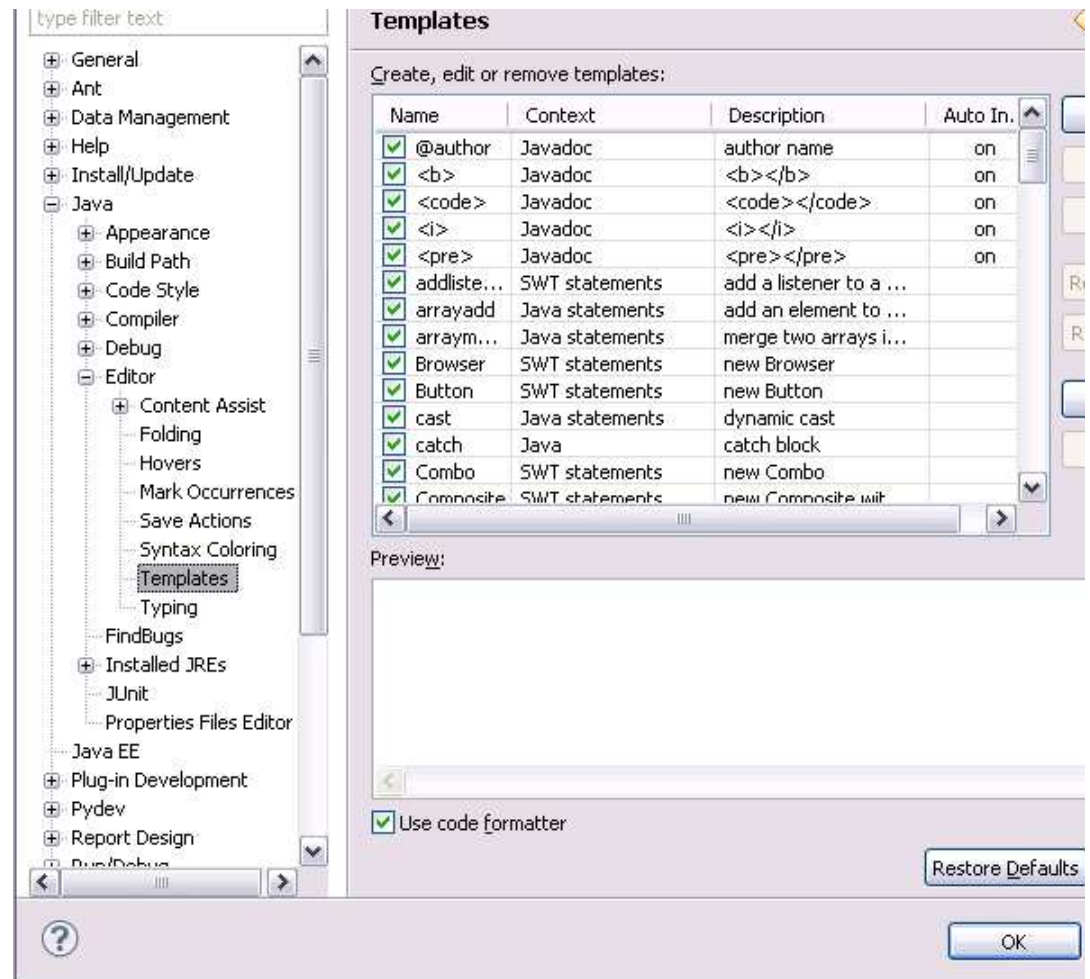
For example, assume that you are frequently creating `public void name(){} me`. You can define a template which creates the method body for you.

To create a template for this, select the menu Window → Preferences → Java → Ec

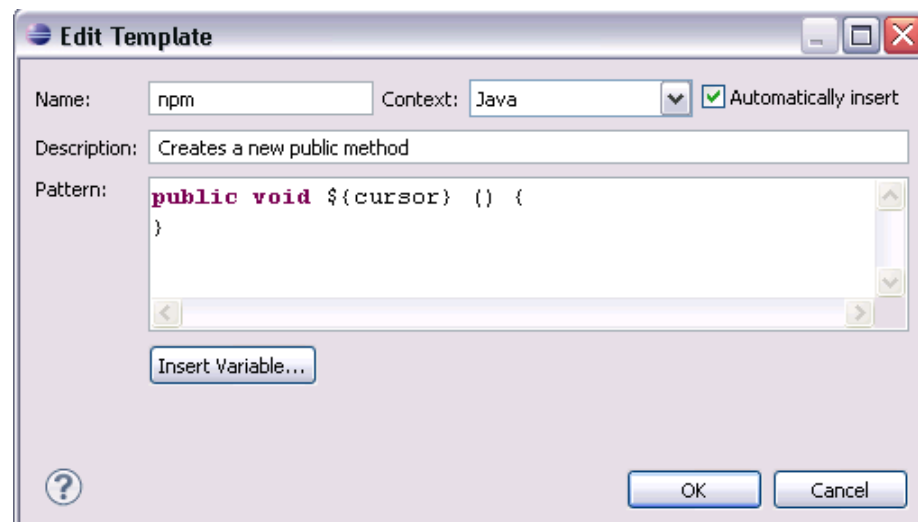




[Home](#)
[Tutorials](#)
[Trainings](#)
[Books](#)
[Connect](#)



Press New. Create the template shown in the following screenshot.



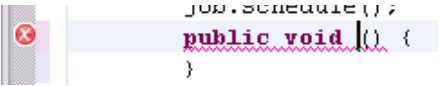
`${cursor}` indicates that the cursor should be placed at this position after applying

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

Now every time you type "npm" in the Java editor and press `Ctrl+S` the system



 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)



## 12.6. Configuring the editors for a file extension

The `Editors` which are available to open a file can be configured via `Window → Preferences → Editors → File Associations`

.

The "Default" button in this preference dialog allows to set the default editor for a file extension, e.g. this is the editor which will be used per default if you open a new file with this extension.

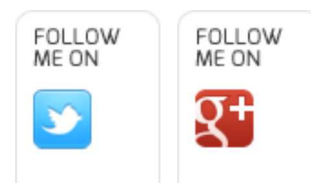
The other configured `Editors` can be selected, if you right mouse click on a file in the `Project Explorer`. Eclipse will remember the last `Editor` you used to open a file and use this `Editor` you open the file.

## 12.7. Code Templates

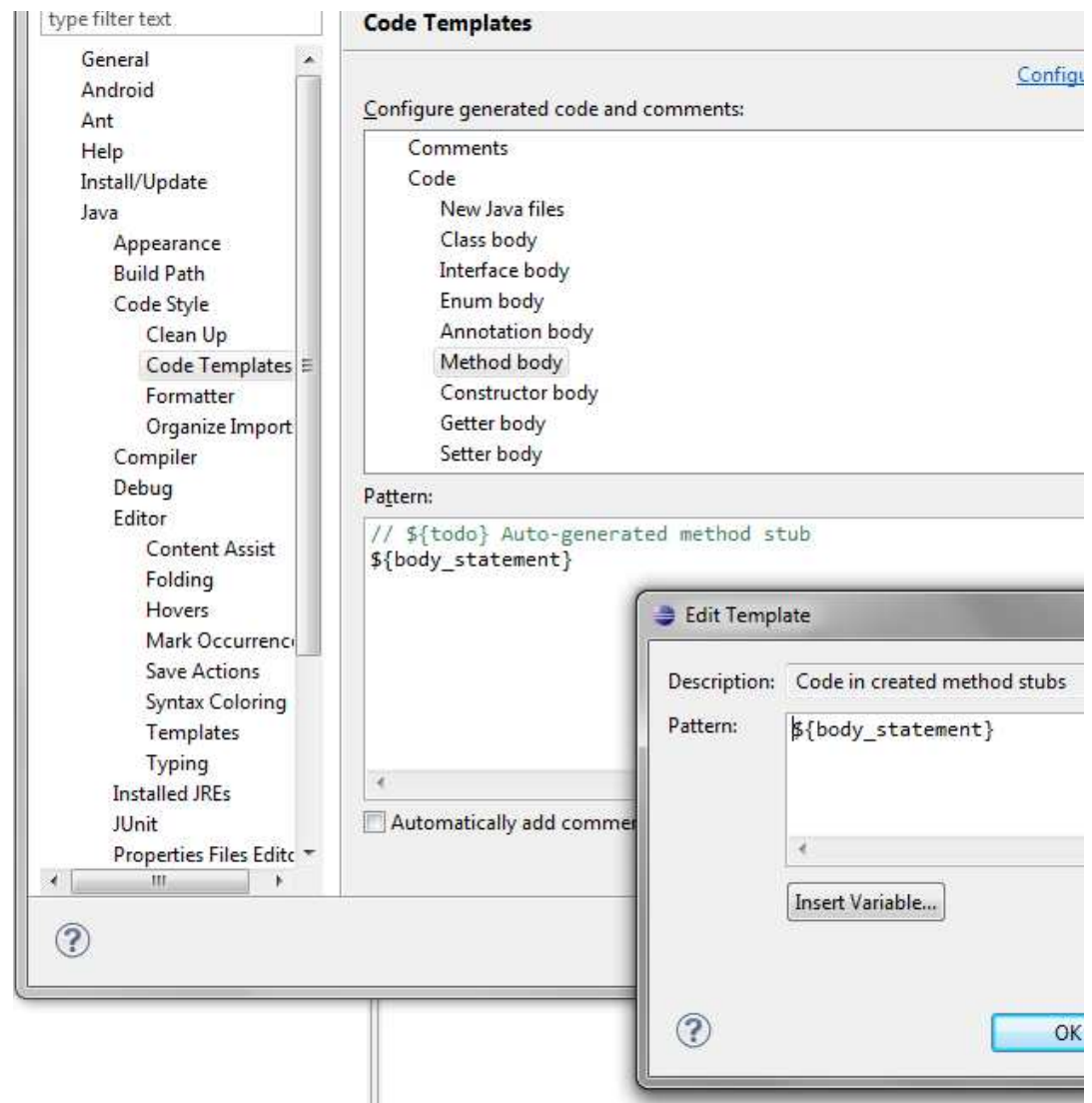
Eclipse generates lots of source code automatically. For example, in several cases it generates boilerplate code to the source code.

Select `Window → Preferences → Java → Code Style → Code Templates` to change the code templates.

In the code tree you have the templates. Select for example `Code → Method Body`; edit this template and to remove the "todo" comment.



[Home](#)
[Tutorials](#)
[Trainings](#)
[Books](#)
[Connect](#)



## 12.8. Export / Import Preferences

You can export your preference settings from one workspace via **File → Export → G Preferences**.

Similarly you can import them again into another workspace.

## 12.9. Task Management

You can use `// TODO` comments in your code to add task reminders.

This indicates a task for Eclipse. You find those in the Task view of Eclipse. Via double-click task you can navigate to the corresponding code.

You can open this view via **Window → Show View → Tasks**.

For example, add a TODO to your `MyFirstClass` in the Tasks view.



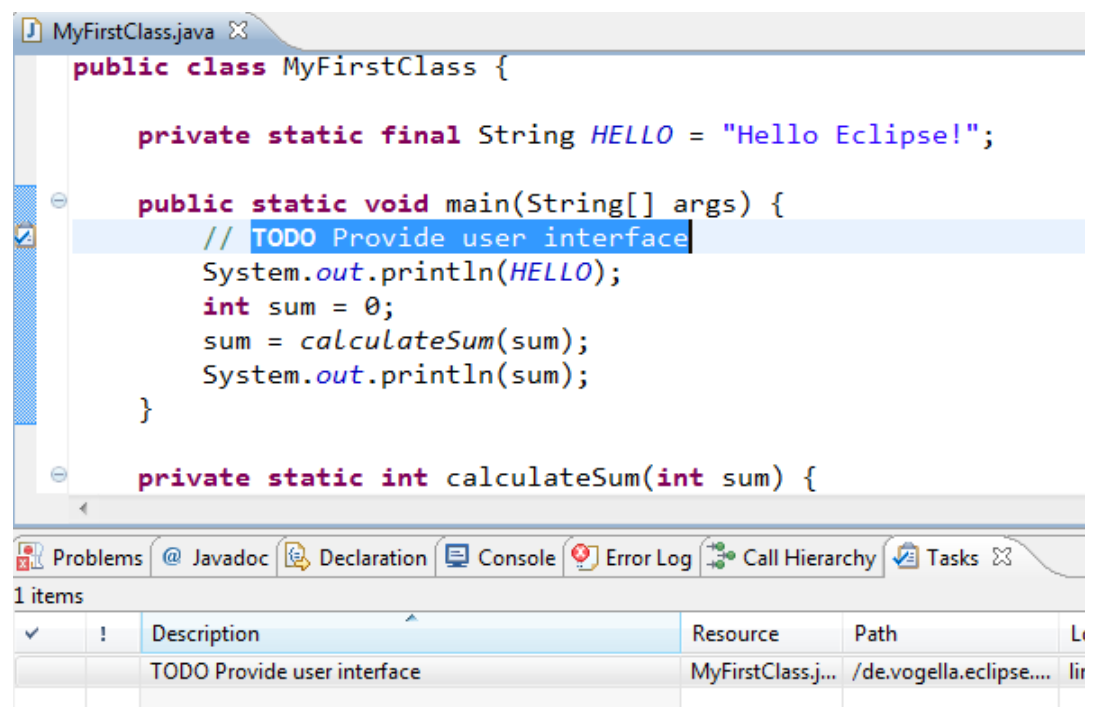
```
public class MyFirstClass {

    private static final String HELLO = "Hello Eclipse!";

    public static void main(String[] args) {
        // TODO Provide user interface
        System.out.println(HELLO);
        int sum = 0;
        sum = calculateSum(sum);
        System.out.println(sum);
    }

    private static int calculateSum(int sum) {
        for (int i = 0; i <= 100; i++) {
            sum += i;
        }
        return sum;
    }
}
```

Close the editor for the `MyFirstClass` class. If you now double-click on the tasks, again and the TODO comment is selected.



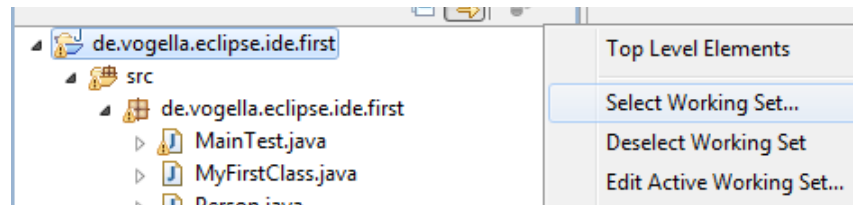
## 12.10. Working Sets

You will create more and more projects in your development career. Therefore the d workspace grows and it is hard to find the right information.

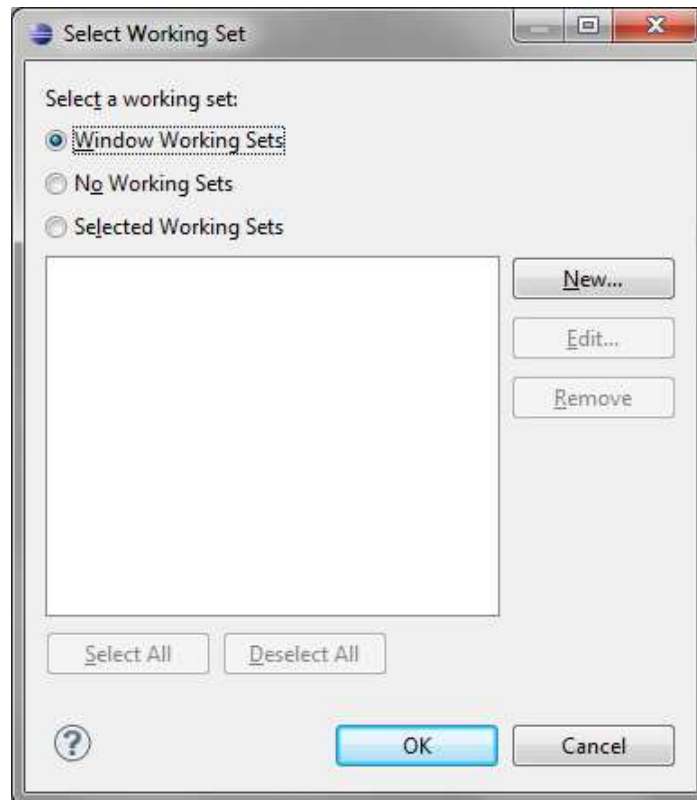
You can use working sets to organize your dis played projects / data. To set up your the Package Explorer → open the drop-down → Sele working Set...



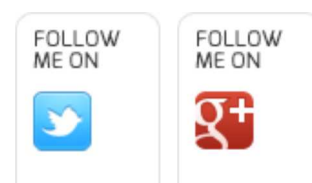
[Home](#) [Tutorials](#) [Trainings](#) [Books](#) [Connect](#)



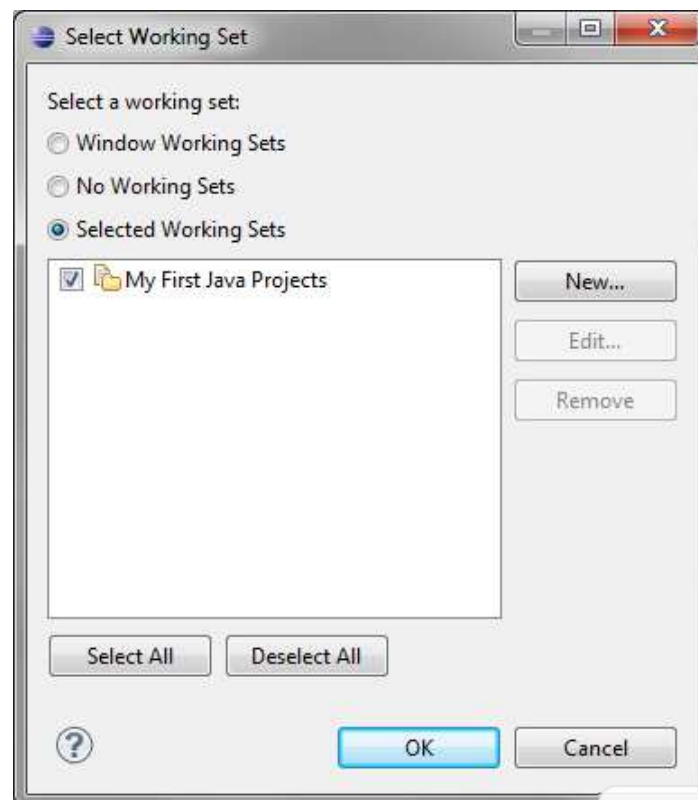
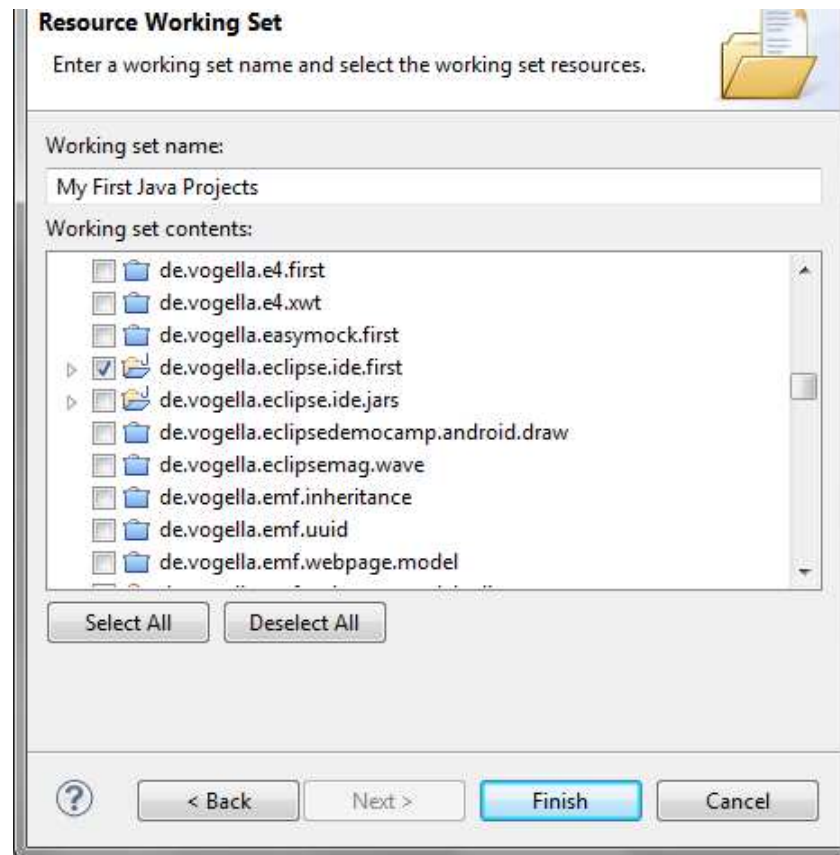
Press "New" on the following dialog to create a working set.



On the next dialog select "Resource", press Next and select the projects you would a name.

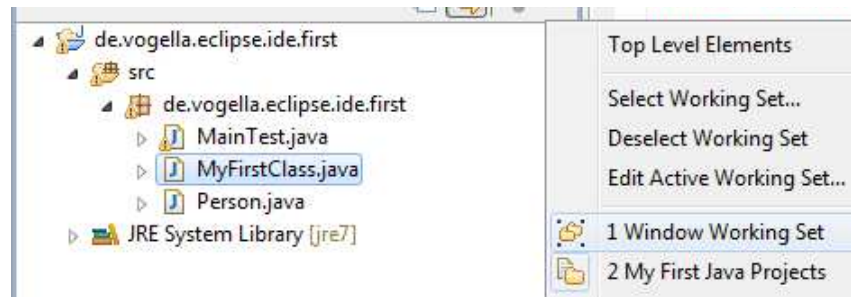


 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)



You can now easily display only the files you want to see.





## 13. Eclipse Help and Community

### 13.1. Eclipse Bugs

Eclipse has a public bug tracker based on Bugzilla. This bugtracker can be found at <https://bugs.eclipse.org/bugs/>. Here you can search for existing bugs and review them.

To participate actively in the Eclipse bugtracker you need to create a new account. By pressing the "Open a New Account" link.



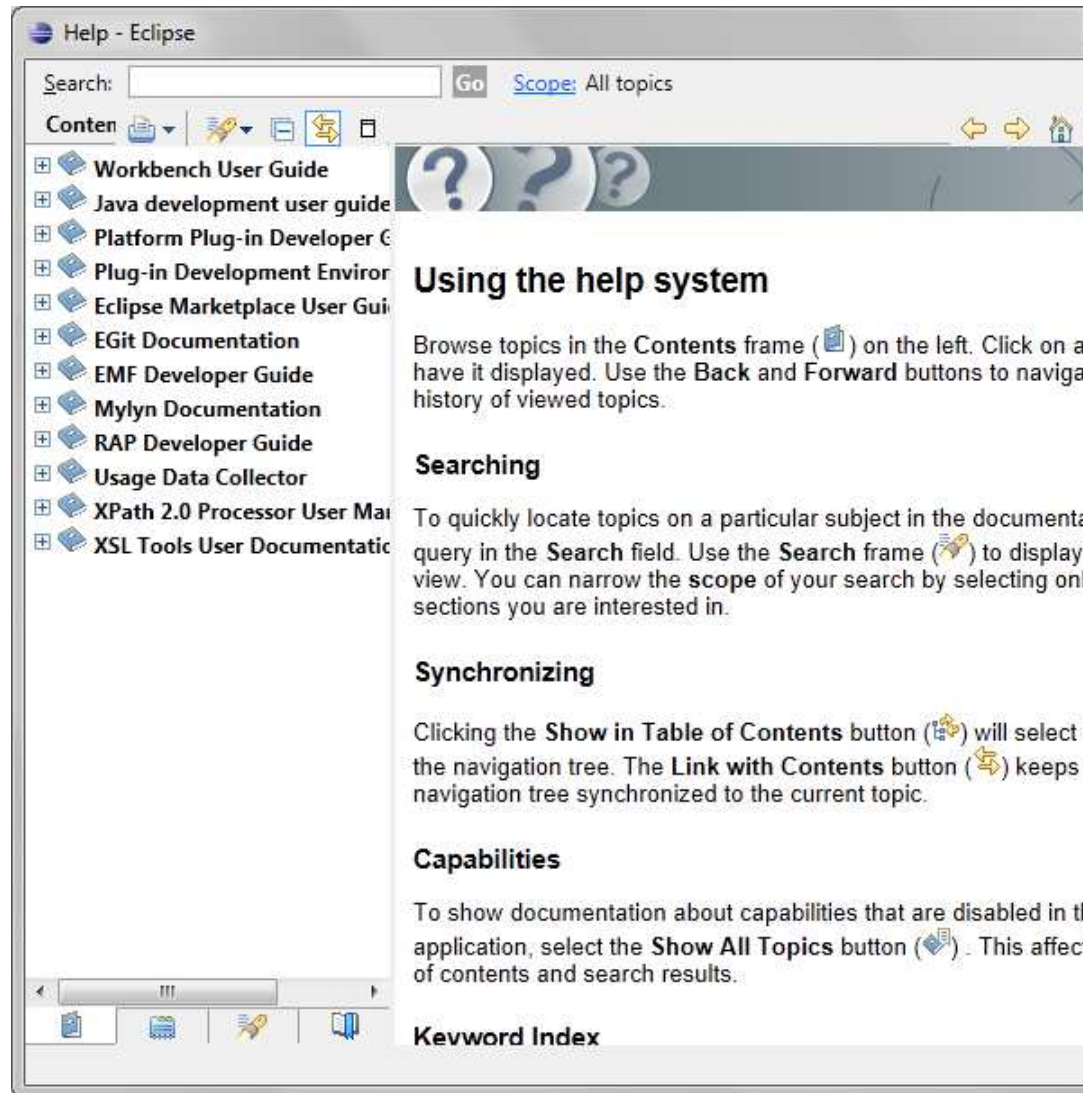
Once you have an user you can login to the Eclipse bugtracker. This allows you to



## 13.2. Online documentations

The Eclipse help system is available from within your Eclipse installation as well as

With your running Eclipse IDE you can access the online help via Help → Help Contents new window which shows you the help topics for your currently installed componen



Online you find the online help under <http://www.eclipse.org/documentation/>. The online help is independent and contains the help for all Eclipse projects included in the selected release.






## 13.3. Asking (and answering) questions

Due to the complexity and extensibility of Eclipse you will need additional resources to solve your specific problems. Fortunately the web contains several resources which can help you solve Eclipse problems.

Currently the best places to ask questions are [Eclipse forums](#), which can be found





 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

The Eclipse forums offer several topic specific forums in which you can post and answer questions. In the Eclipse forums you need a valid user in the Eclipse bugtracker. The Eclipse forums is that, depending on the topic, Eclipse committers are also active and directly answer your question.

Stackoverflow also requires a user and its community is also very active. Stackoverflow has separate forums for specific questions. In Stackoverflow you tag your questions with the e.g. "Eclipse" and people interested in these keywords search for them or subscribe.

Both places are excellent places to ask questions. If you ask a question it is in general polite and to give a good error description as this motivates people to give you high

### 13.4. Webresources

The Eclipse homepage also contains a list of relevant resources about Eclipse and programming. You find these resources under <http://www.eclipse.org/resources/>.

Also the author of this description maintains several Eclipse relevant tutorials on his all his Eclipse related articles on <http://www.vogella.com/eclipse.html>.

## 14. 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](#). To develop rich stand-alone Java clients you can use [Eclipse RCP](#). You can extend Eclipse [Plug-ins](#).

Good luck in your journey of learning Java!

## 15. Thank you

Please help me to support this article:



## 16. Questions and Discussion

Before posting questions, please see the [vogella FAQ](#). If you have questions or find an article please use the [www.vogella.com Google Group](http://www.vogella.com/GoogleGroup). I created a short list of [questions](#) which might also help you.



 [Home](#)  [Tutorials](#)  [Trainings](#)  [Books](#)  [Connect](#)

## 17.1. Source Code

### Source Code of Examples

## 17.2. Eclipse Resources

### Eclipse.org Homepage

## 17.3. vogella Resources

Eclipse RCP Training (German) Eclipse RCP Training with Lars Vogel

Android Tutorial Introduction to Android Programming

GWT Tutorial Program in Java and compile to JavaScript and HTML

Eclipse RCP Tutorial Create native applications in Java

JUnit Tutorial Test your application

Git Tutorial Put everything you have under distributed version control system

