

# Project Development in Eclipse

- [Before You Start](#)
- [Installing Eclipse](#)
  - [Configuring Plugins](#)
    - [Git & GitHub](#)
  - [Creating a New Java Project](#)
  - [Importing Maven Projects](#)
  - [Working Sets](#)
  - [Maven Run Configuration](#)
- [Project Source Code Formatting](#)
  - [Disabling Tabs](#)
    - [Disabling Tabs in Java Files](#)
    - [Disabling Tabs in XML Files](#)
    - [Disabling Tabs in Text Files](#)
    - [Configuring Other Editors](#)
- [Working with Subversion](#)

An Integrated Development Environment (IDE) should be used to develop source code. [Eclipse](#) is a good choice, as it is free, widely used, and well supported.



## Solving Startup Problems

If Eclipse hangs during startup or while initializing the current workspace, then try starting using `eclipse -clean -clearPersistedState` which will often resolve these types of issues.

## Before You Start

Before you start to setup Eclipse, you should follow the instructions at [Installing HPS Java](#) to install locally the HPS Java trunk, which contains the various modules that will be loaded as projects.

## Installing Eclipse

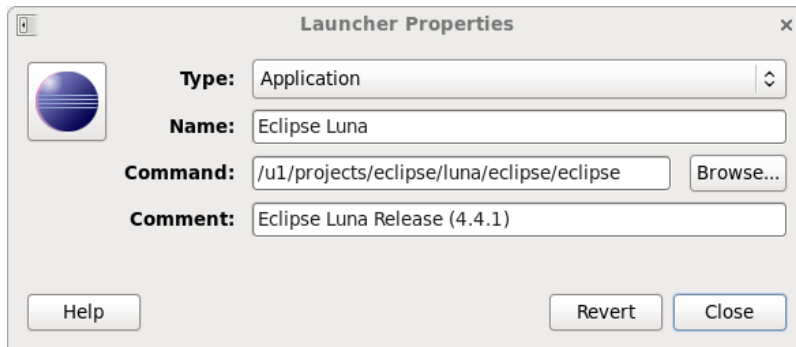
You can download an installation bundle from [the Eclipse downloads site](#).

There are many different Eclipse configurations provided here, and you will probably want the [Eclipse IDE for Java Developers](#), which provides a Java IDE and built-in support for Maven. (The link is specifically for the Luna release.)

To install the IDE, simply download the correct file for your operating system by following the links, and then use `unzip` or `tar` to unpack it into a local directory.

You should see a directory called **eclipse** and the script at **eclipse/eclipse** will start the application.

This program can also be setup as an application shortcut in your window manager, which on my Linux system looks something like this.



The actual paths will, of course, depend on where you have installed Eclipse on your system.

## Configuring Plugins

### Git & GitHub

We used to include the [Subversive Plugin](#), which provides a graphical Subversion client within Eclipse, implementing commands like commit, update, etc.

Since we now switched to GitHub, follow the instructions from here: <http://www.eclipse.org/egit/download> to get a git and GitHub plugin installed. (This is not strictly needed to use Eclipse)

# Eclipse Projects

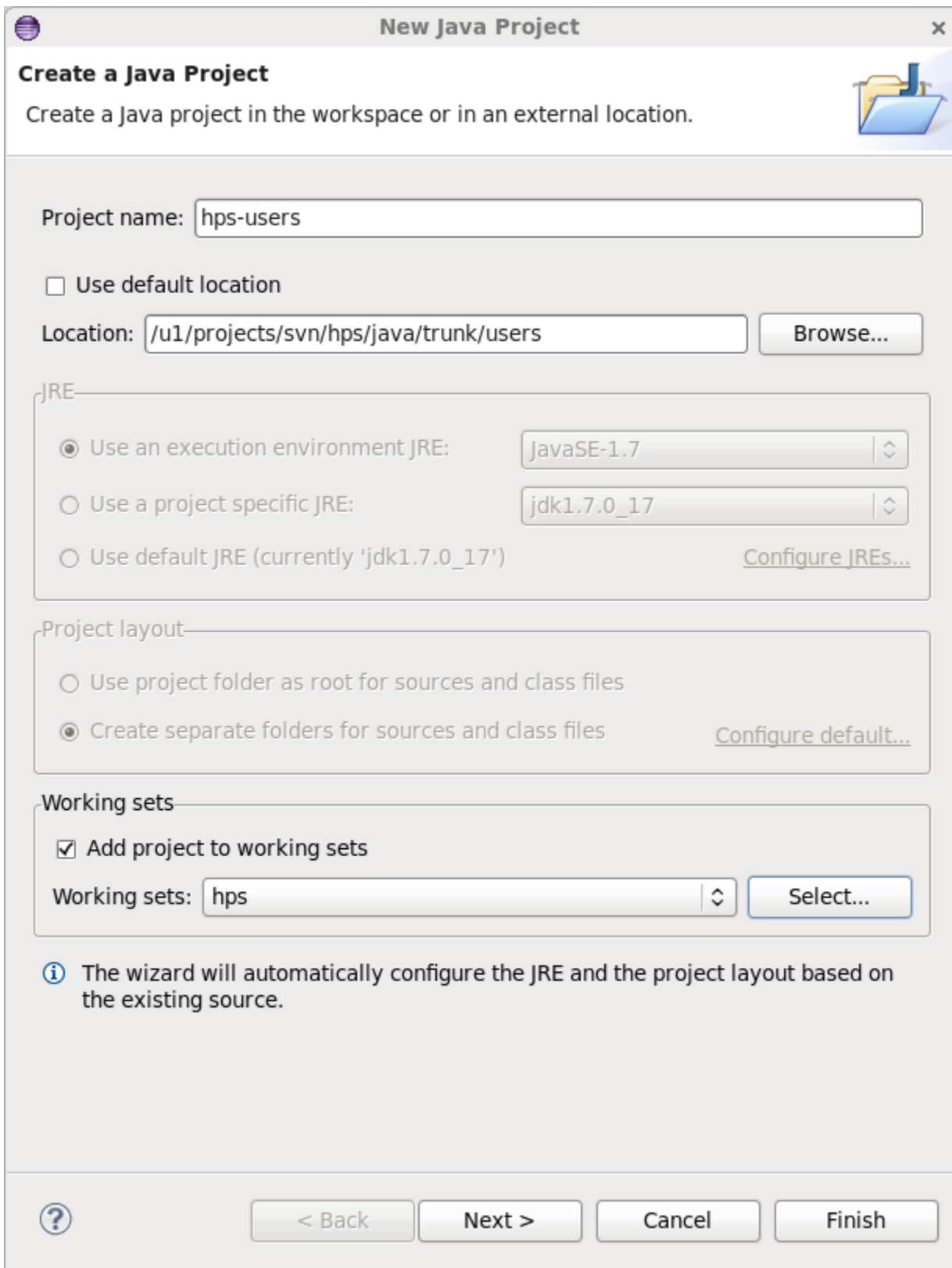
## Creating a New Java Project

To create a new project in Eclipse, you can select **File > New > Java Project** from the main menu. This will start a wizard for creating the new project.

We will create a project for the *users* module where miscellaneous user analysis code is located in HPS Java.

In the **New Java Project** window, you need to do the following.

1. Type the name of the project in the *Project name* text box, e.g. "hps-users".
2. Uncheck **Use default location** and navigate to the root directory of the module for the project using *Browse*.
3. Add the project to a working set (optional).



The screenshot shows the 'New Java Project' dialog box. At the top, it says 'Create a Java Project' and 'Create a Java project in the workspace or in an external location.' The 'Project name' field contains 'hps-users'. The 'Location' field contains '/u1/projects/svn/hps/java/trunk/users' with a 'Browse...' button next to it. Under the 'JRE' section, 'Use an execution environment JRE:' is selected, with 'JavaSE-1.7' in the dropdown. Other options are 'Use a project specific JRE:' (with 'jdk1.7.0\_17') and 'Use default JRE (currently 'jdk1.7.0\_17')'. There is a 'Configure JREs...' link. Under 'Project layout', 'Create separate folders for sources and class files' is selected, with a 'Configure default...' link. Under 'Working sets', 'Add project to working sets' is checked, and the 'Working sets' dropdown shows 'hps' with a 'Select...' button. A note at the bottom states: 'The wizard will automatically configure the JRE and the project layout based on the existing source.' The bottom of the dialog has a help icon, '< Back', 'Next >', 'Cancel', and 'Finish' buttons.

**New Java Project**

**Create a Java Project**  
Create a Java project in the workspace or in an external location.

Project name:

☐ Use default location

Location:

**JRE**

☒ Use an execution environment JRE:

☐ Use a project specific JRE:

☐ Use default JRE (currently 'jdk1.7.0\_17') [Configure JREs...](#)

**Project layout**

☐ Use project folder as root for sources and class files

☒ Create separate folders for sources and class files [Configure default...](#)

**Working sets**

☒ Add project to working sets

Working sets:

**i** The wizard will automatically configure the JRE and the project layout based on the existing source.



Then click on the **Finish** button and the project should now show up in the **Package Explorer** within the Java View.

## Importing Maven Projects

Instead of checking out and creating individual projects, Eclipse can also be configured to import multiple Maven modules at once.

To do this, follow these steps.

1. Checkout the trunk using SVN.
2. In Eclipse, go to **File > Import** and select **Maven > Existing Maven Projects** from the tree menu.
3. Click **Next** and then browse to the path of the trunk under the **Root Directory** field.
4. All of the modules will show in the window now. You can leave all of these checked, unless you do not want some modules to be imported as projects, in which case uncheck those.
5. In the same panel, before continuing, click **Add projects to working set** and create an **HPS** working set to which the new projects will be assigned.
6. To create the projects, click **Next** and Eclipse will automatically import all the selected modules into your workspace as projects with the Maven and Java natures.

Now all of the Maven modules in HPS Java are imported into your workspace.

## Working Sets

To enable commands in the toolbar which will allow showing or hiding projects by their working set, go to **Window > Customize Perspective** and click the boxes for **Window Working Set** and **Working Set Manipulation** under **Command Groups Availability**.

The working set toolbar should be added to your dock now. There should be an icon (looks like two folders) where you can select the working sets to view e.g. **HPS**. This will filter the **Package Explorer** and similar views to only show projects from that working group.

The upside down triangle in the upper right of the **Package Explorer** or **Project Explorer** is the **View Menu** and can be used to filter out or include working sets in that particular component.

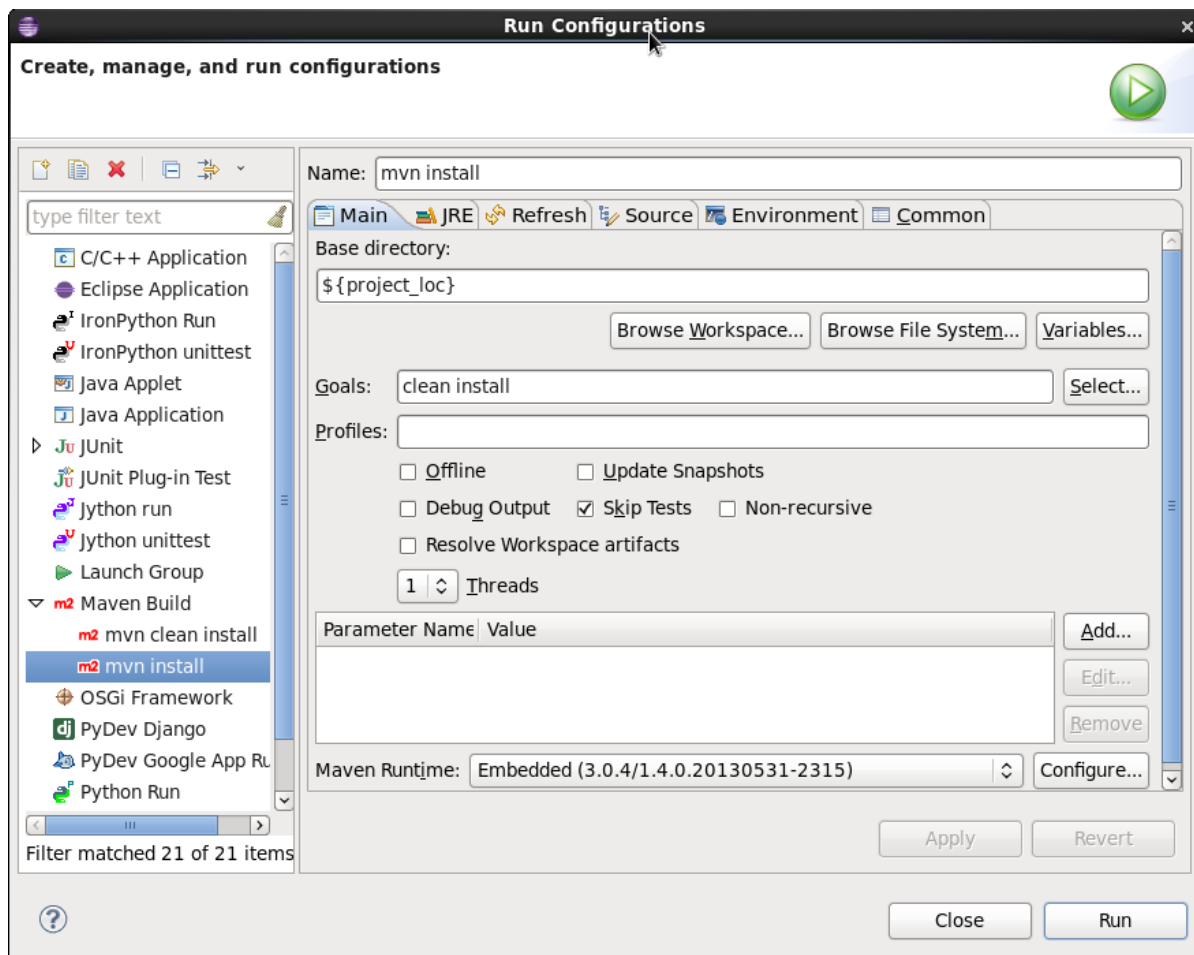
## Maven Run Configuration

It is useful to create a custom build configuration for executing Maven on the project.

Open the window **Run > Run Configurations**.

Right click on **Maven Build** and select **New**.

Setup the new configuration to look like this.



In order to use this configuration, automatic builds should be turned off, which can be done by deselecting **Project > Build Automatically** from the application menu.

To execute this build configuration, select the project in the **Package Explorer** or **Project Explorer** by left clicking on it, and then go to **Run > Run Configurations**, click on your Maven configuration, and finally click the **Run** button there.

The build log should show up in the console window as the project compiles.

#### **Resolving Dependencies in Eclipse**

Should Eclipse be unable to resolve a project's dependencies, try right-clicking on the project icon and selecting **Maven > Update Project** from the menu or hitting **Alt + F5** while the project is selected. This should force Eclipse to update its index of jar files against your local Maven repository, and hopefully this will make all the red errors in your editor go away.

## Project Source Code Formatting

#### **Formatting Source Code**

You can apply the formatting on Java files a few different ways. From the **Package Explorer**, you can select one or more files, and then right click and select **Source > Format**.

Within an editor window, you can format an entire file by right clicking and selecting **Source > Format**. This will also work on a section of selected code.

The HPS collaboration uses a standard Java code formatting convention based on the [Java Code Conventions](#).

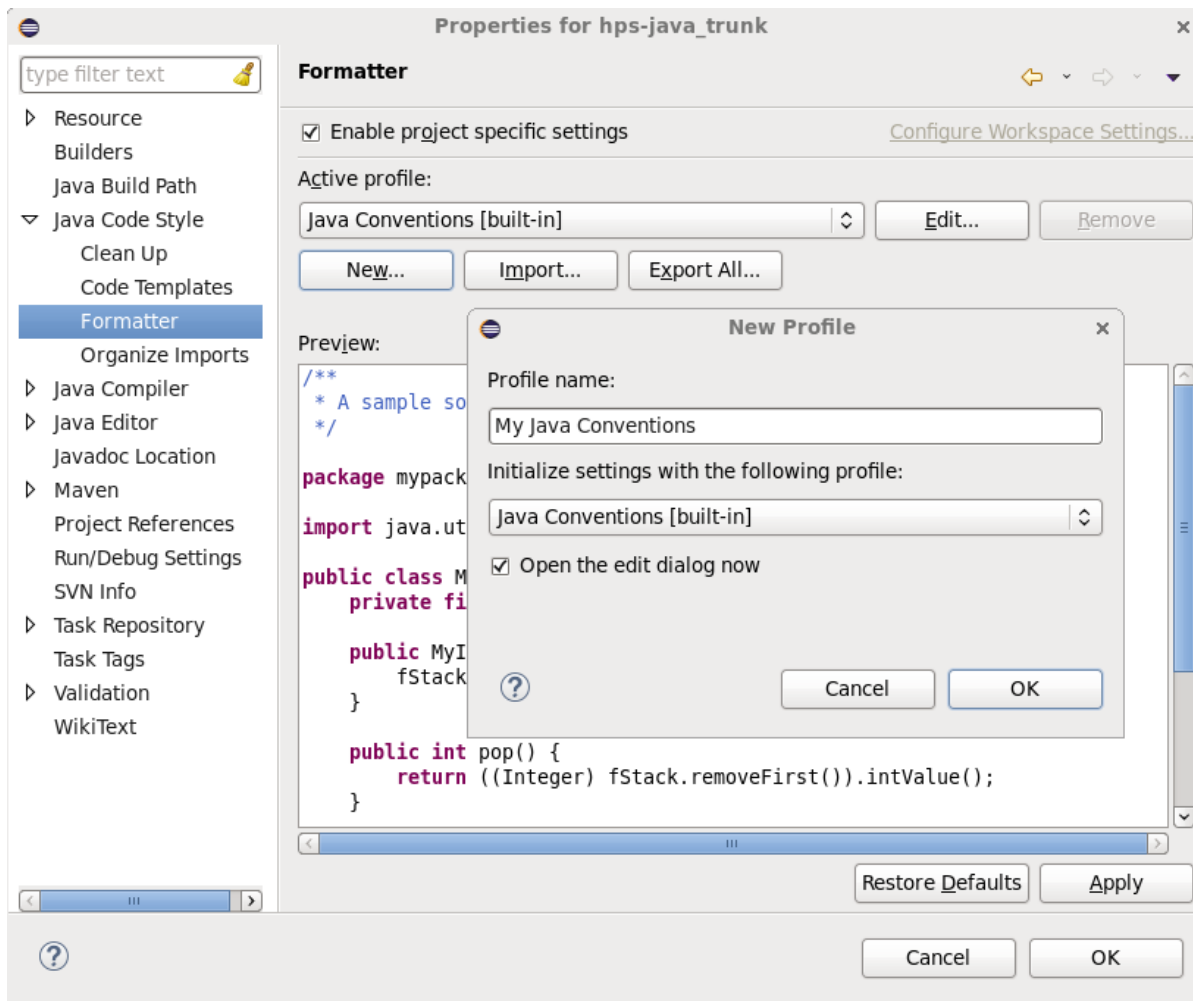
To set this up for your projects, right click on a project in Eclipse, then select **Properties**.

Navigate to **Java Code Style > Formatter** to edit the settings for this project.

Check the box marked **Enable project specific settings** and then click the *New* button.

Name the profile e.g. **My Java Settings** and select **Java Conventions** from the drop-down menu.

This configuration will appear something like the following now.



Then click **OK** to edit the settings.

You will now edit the specific formatting settings which will be used for Java projects.

There are only a few changes which will be made to the defaults.

## Disabling Tabs

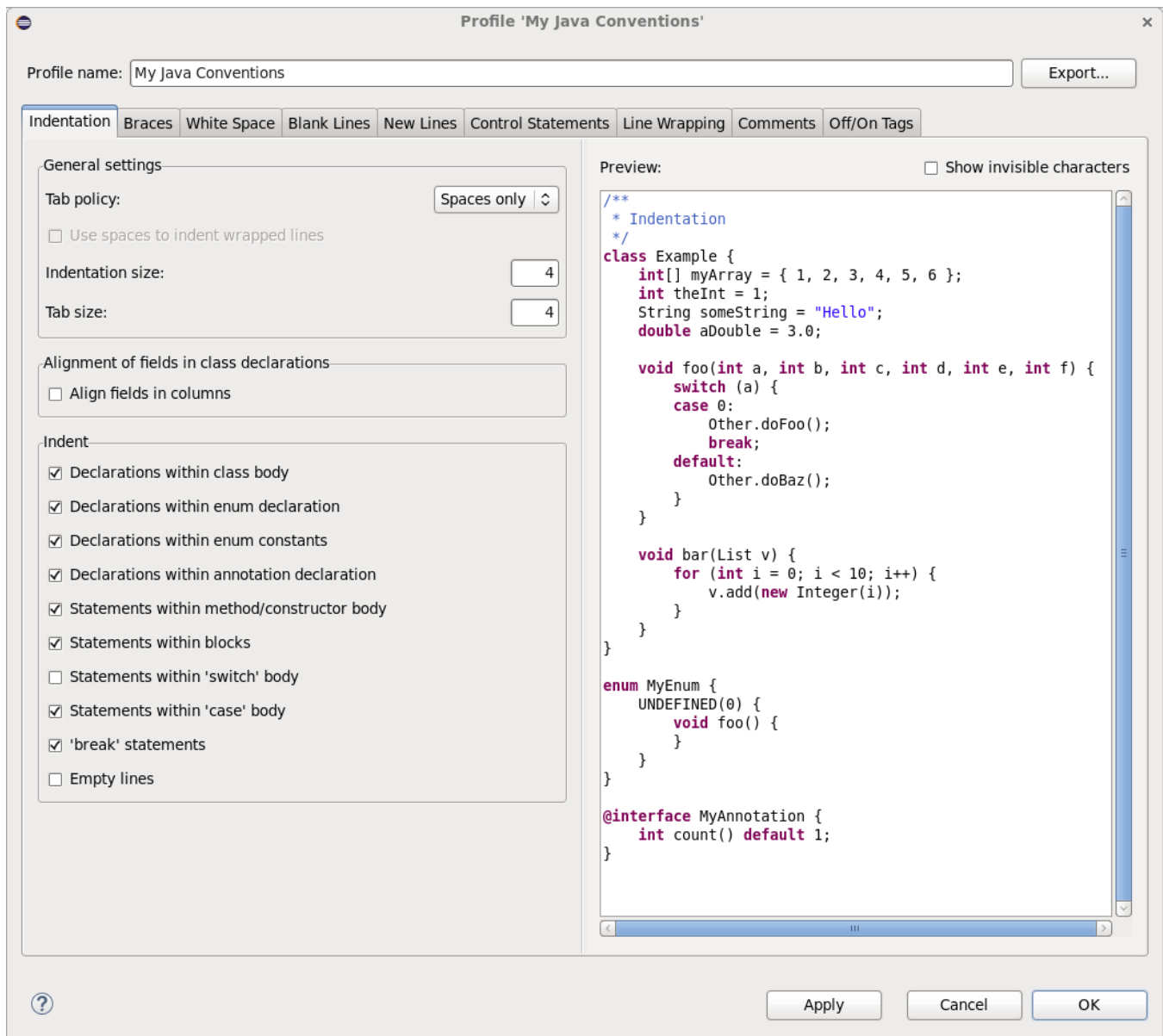


### Tabs are Evil

As a general rule, tabs should never be used in *any* files within the SVN. Mixing tabs and spaces creates a formatting mess. And tabs may appear differently depending on how someone has their editor configured (e.g. tab indentation could be 4 or 8 spaces). For these reasons, we stick to a rule of **spaces only** in all our source code.

## Disabling Tabs in Java Files

Under **Tab policy** in the formatting settings for the currently using code conventions, **Spaces only** should be selected with a **Tab size** of **4**. Next click **Apply** and then **OK**.



## Disabling Tabs in XML Files

Disable tabs in XML files by opening the **Preferences** menu and navigating to **XML > XML Files > Editor** where **Indent using spaces** should be selected and the indentation size set to **4**. Click **Apply** and **Okay**.

## Disabling Tabs in Text Files

To disable tabs in the general text editor, open **Preferences** and go to **General > Editors > Text Editors** and make sure **Insert spaces for tabs** is checked with Displayed tab width set to **4**.

## Configuring Other Editors

There are other places where this can be configured as well, for various types of editors. [This link](#) provides useful answers on where all these settings are located and how they can be configured to use spaces only.

## Working with Subversion





### Using Team Commit from Eclipse

Using Eclipse to commit files can be convenient. But you need to be very careful about this. When you select commit, Eclipse will show a window with the list of files to be committed. Make absolutely sure that this list only contains the files which you actually want included into the commit. By default, *when executing a commit from a project without any files selected, Eclipse will add every file which is not in the global ignore list to the commit.* You absolutely do not want this. So if there is a huge list of files there, then click cancel and execute the command only with the correct files selected. Or you may uncheck the boxes next to files which should not be committed. Committing files from a multiple selection in the **Project Explorer** or **Package Explorer** will also work to commit only the files you want included.

An extensive set of Subversion commands are available under the **Team** and **Compare with** sub-menus when you right click on a Java file or package in the project.

To update a file to its current version in the trunk, you can select **Team > Update**.

You may compare a file against its current remote copy by going to **Compare With > Latest from repository**.

You can add a file to SVN by selecting **Team > Add to version control**.

To commit a file or select of files, you can use **Team > Commit**.

When you remove a file using **Delete** from the right-click menu, Eclipse will remove the file and automatically mark it for deletion in the next SVN commit.