


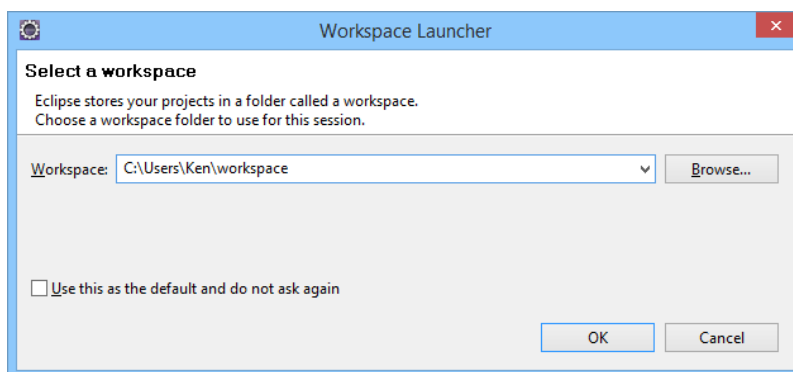
The Omni Programmer

Things I have learned writing software, teaching programming, and living!

Setting up a Workspace in Eclipse for Desktop Application Development using Maven

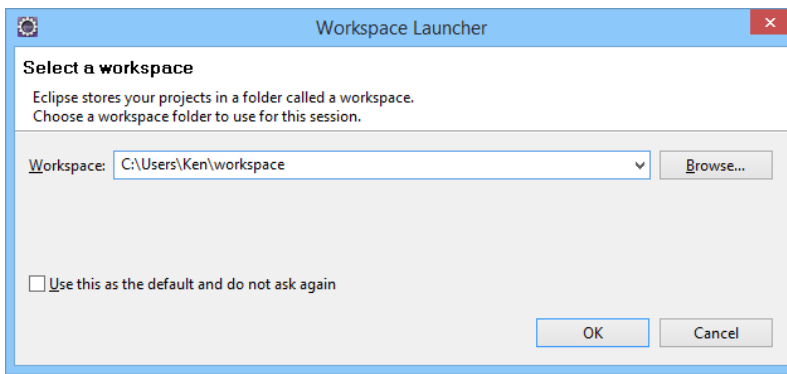
 Ken  2014-08-24  No Comments

This tutorial is primarily for the students in my 420-517 Software Development Project course at Dawson College. It explains how to set up a workspace in Eclipse for this course. These instructions are important because Eclipse stores workspace specific settings in the workspace itself. This means that should you create a new workspace you will need to redo these settings. When you start Eclipse you are asked for the name of the workspace you want to use. If this is the first time you are using Eclipse the workspace launcher will appear as:

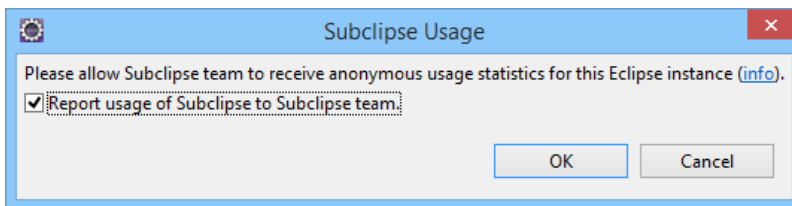


Eclipse wants to create a workspace called **workspace** in your home directory. On Linux and Mac systems you have a Home volume. In Windows the matching space is the folder with your name in the Users folder. If you have used Eclipse on the computer previously then it will show you the last workspace that you used. Pull down the combo box and you will get a list of all the workspaces that you have used. I prefer to have my workspace name indicate what I am doing. Therefore I never accept the default name and instead use a more meaningful name. **Dawson Note:** *On the lab computers the default workspace is in C:\Users\Administrator\workspace.* Here is my workspace choice for this tutorial:





After pressing **OK** Eclipse will start up. I configure Eclipse with the Subclipse plugin for Subversion. For that reason I get the following modal dialog (a dialog that demands attention).



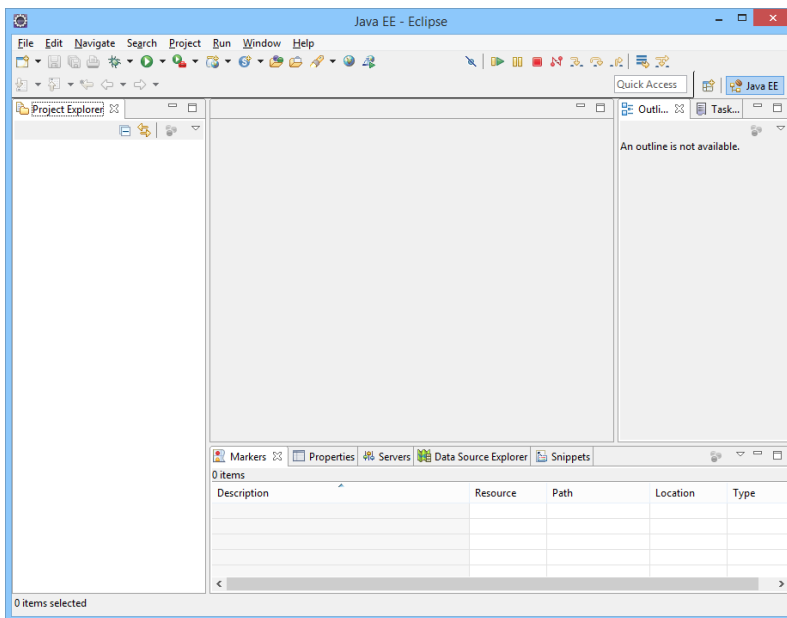
I usually un-check this simply to reduce the traffic over the network. In the school lab this is a good idea. Now we can see the Eclipse program as it appears when a new workspace is selected. Depending on the plugin that you have installed this screen may look slightly different. If you use an existing workspace then this will not appear and you will go directly to the workbench. The workbench is what we call the screen with all the windows we can use in Eclipse.



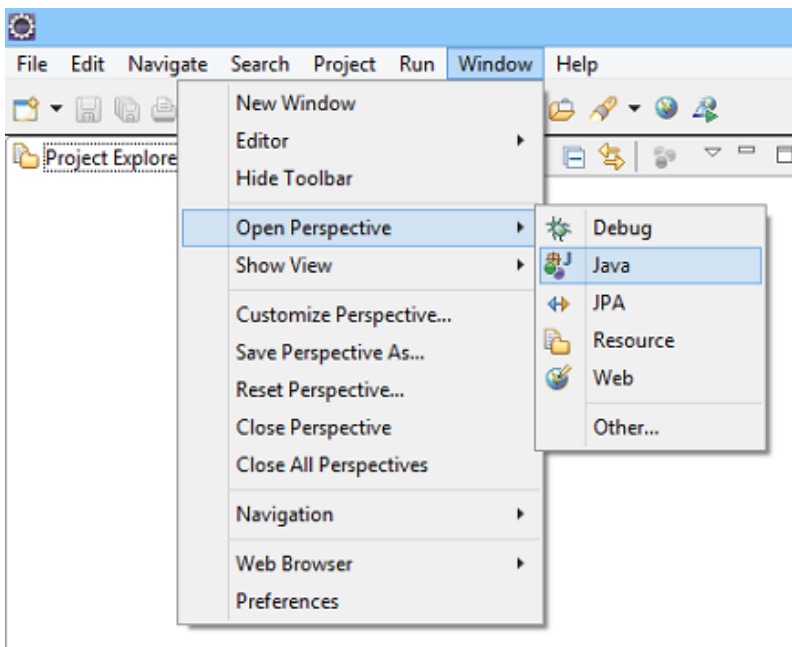
Click on the X on the **Welcome** tab or click on the toilet paper roll to get to the workbench.



It's not really a toilet roll but it sure looks like one to me. The version of Eclipse that we use at Dawson is configured for Java EE. As we are going to create a desktop application we should change the perspective. A perspective is the arrangements and choices of windows on the workbench. Here is the Java EE perspective: ^

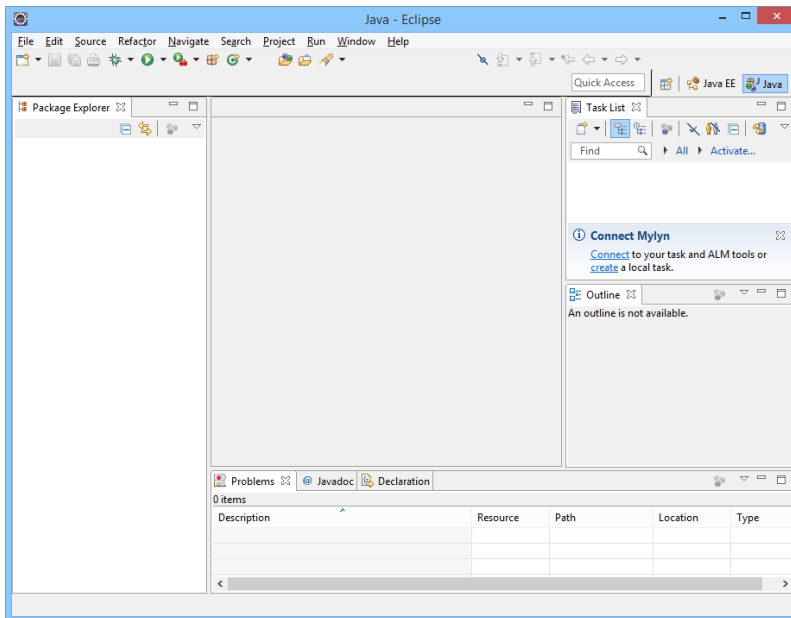


Go to **Window -> Open Perspective** and chose **Java**.

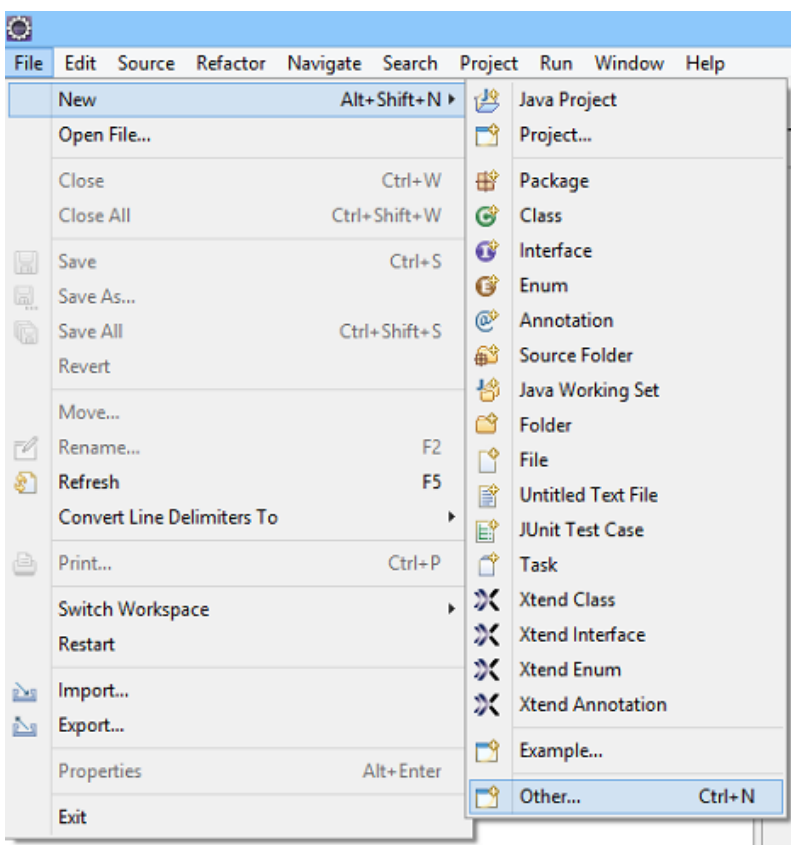


The workbench will now look like:



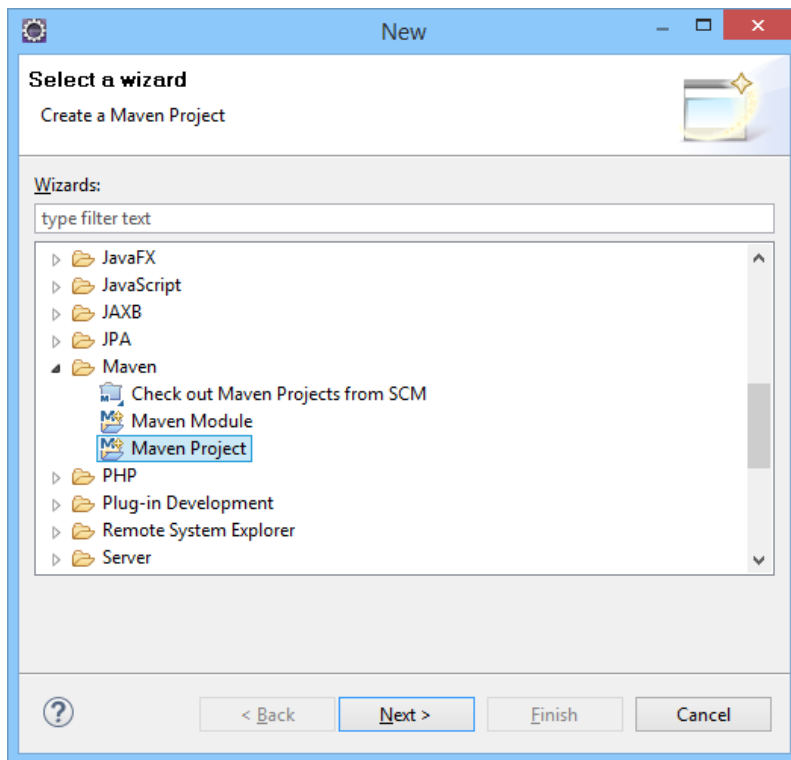


We are now ready to create our first project. We will only create Maven managed projects that use my Super Pom File (see <http://netbeans.dzone.com/nb-class-maven-4-kf>). From the **File** menu select **New -> Other**

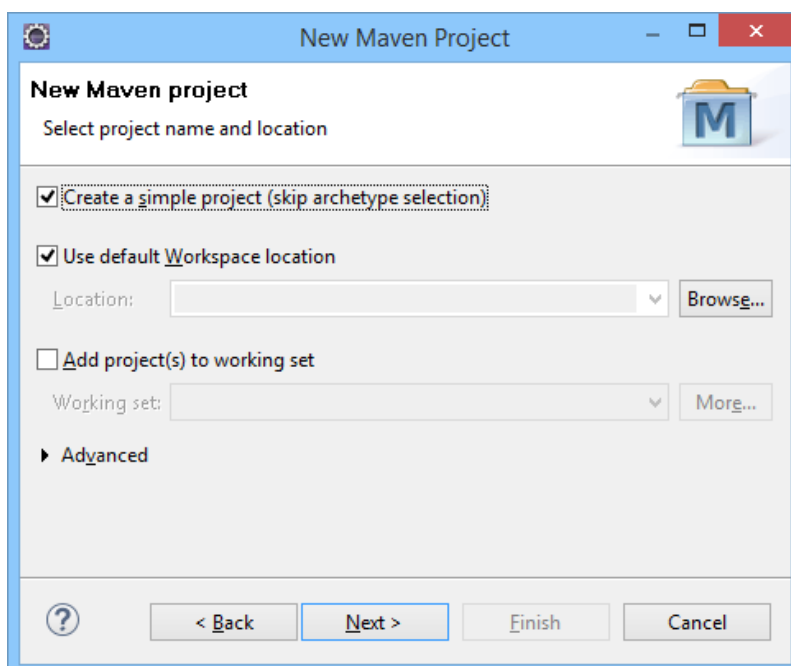


The **New** dialog will appear for you to select a wizard. Scroll down to **Maven** and select **Maven Project** and click on **Next**.



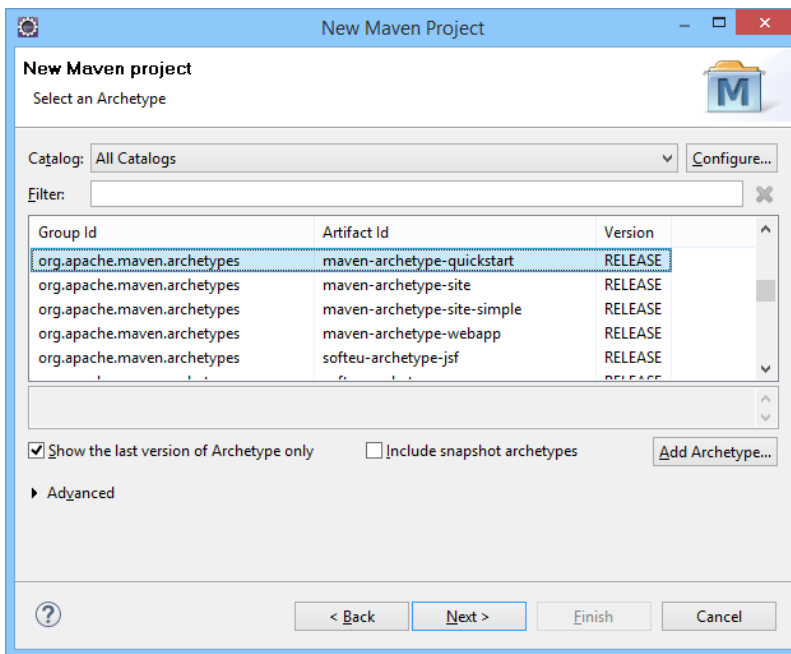


On the **New Maven Project** you must check **Create a simple project (skip archetype selection)**.

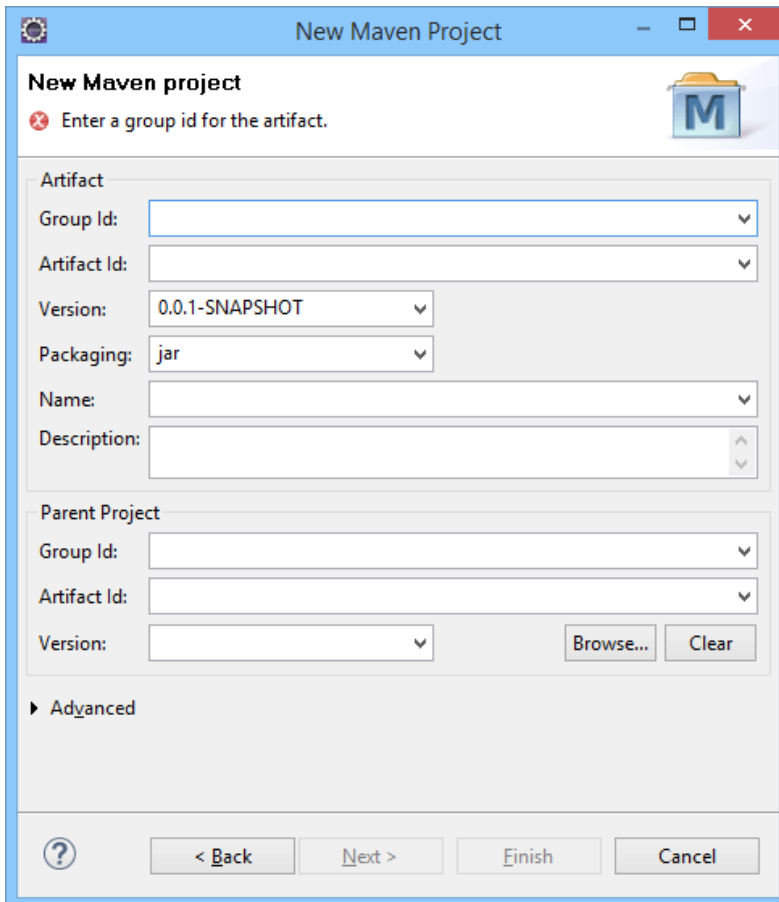


If you forget to do this you will see:

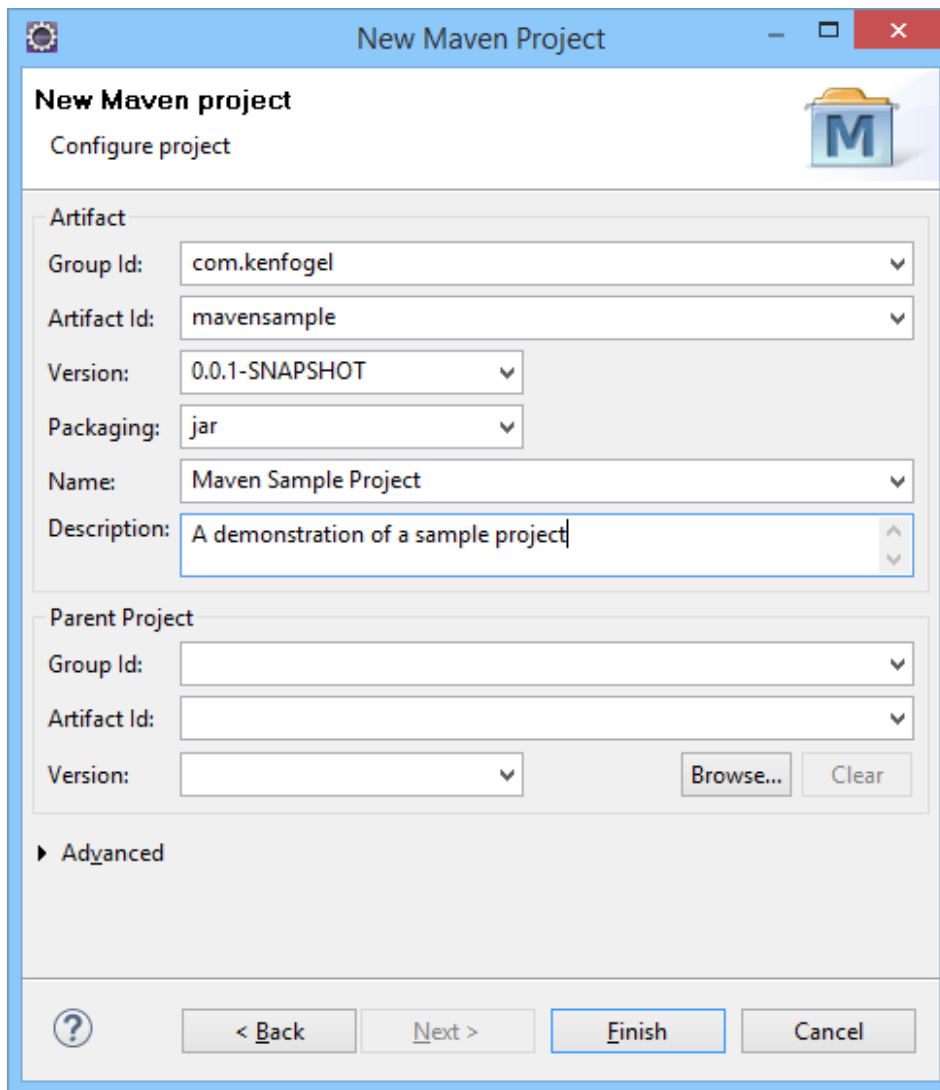




Click on the **Back** button, check **Create a simple project** and when all is well you should see:



Fill in the form as follows but replace my name with yours and chose appropriate names and description for your project.



New Maven Project

Configure project

Artifact

Group Id: com.kenfogel

Artifact Id: mavensample

Version: 0.0.1-SNAPSHOT

Packaging: jar

Name: Maven Sample Project

Description: A demonstration of a sample project

Parent Project

Group Id:

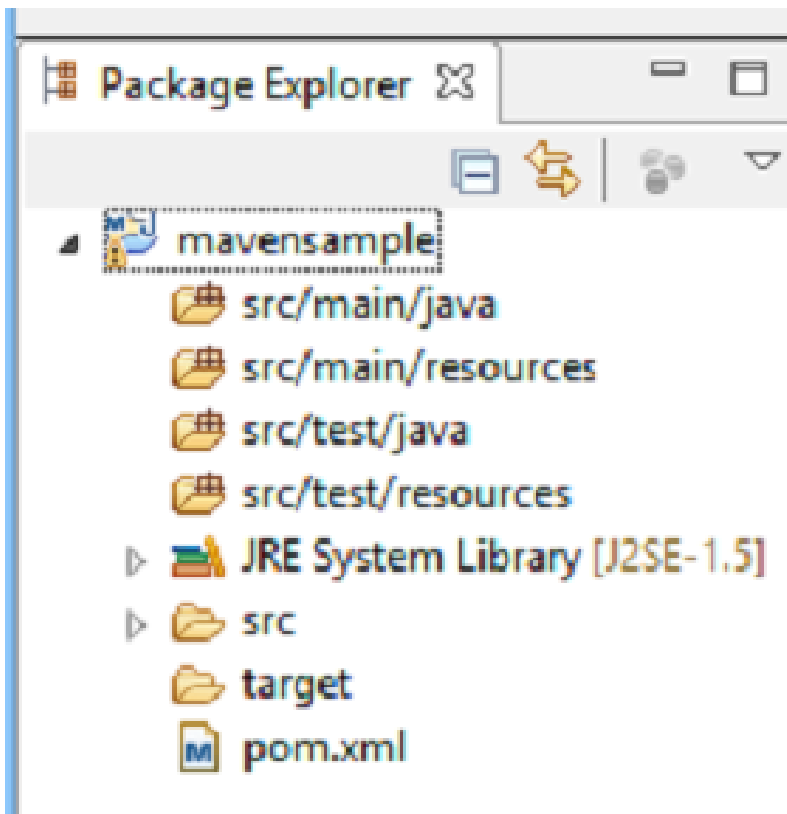
Artifact Id:

Version: Browse... Clear

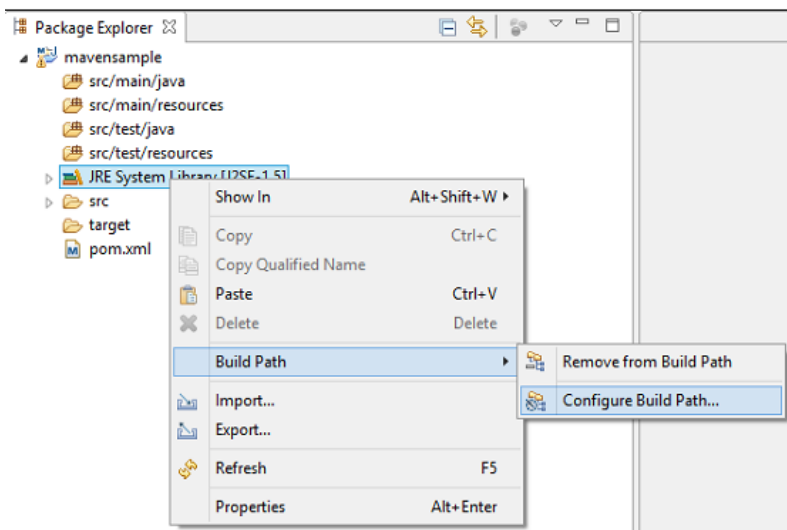
► Advanced

? < Back Next > Finish Cancel

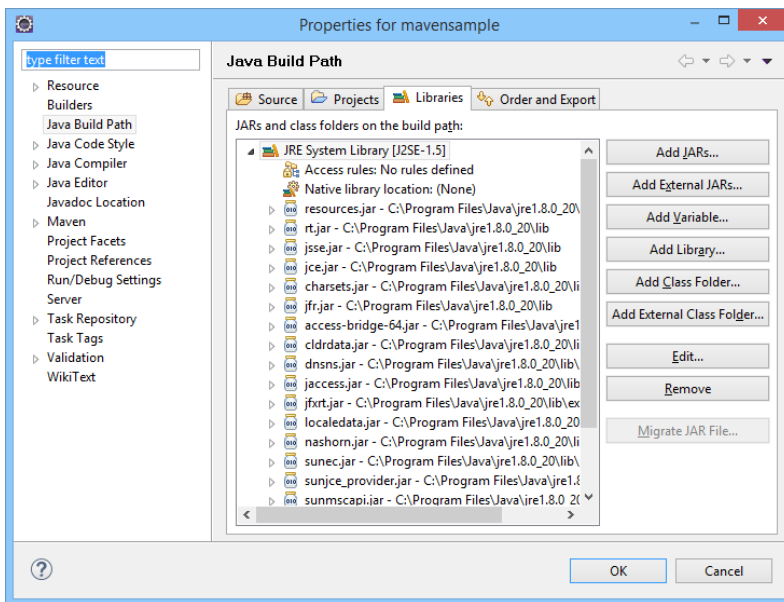
Click on **Finish** and the project should appear in the **Package Explorer**. Open up the project tree by clicking on the small triangle and you should see:



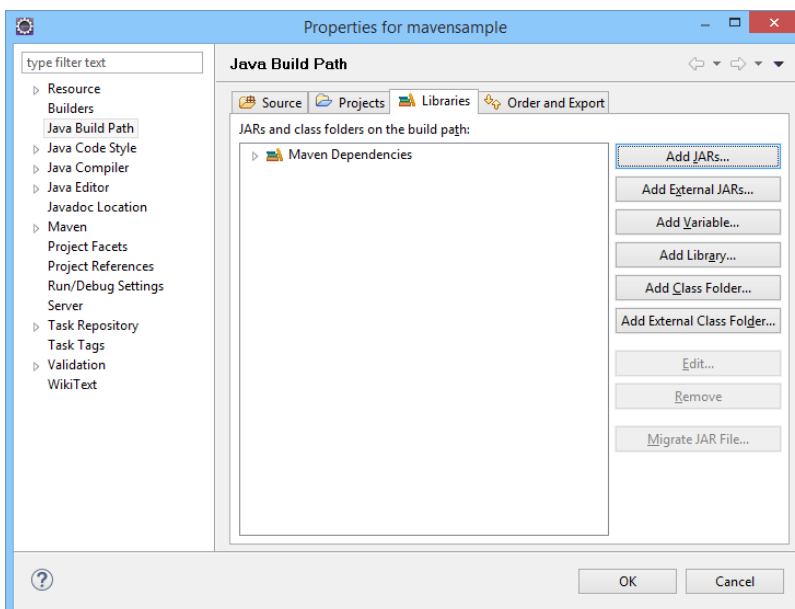
There are two last tasks we must complete before we can start coding. The first thing is to change the JRE System Library. We will be using Java 1.8 and the pom.xml file will indicate Java 1.8. The Maven plugin in Eclipse is configured to default to Java 1.5. Right mouse click on **JRE System Library (J2SE-1.5)** and select **Build Path -> Configure Build Path**.



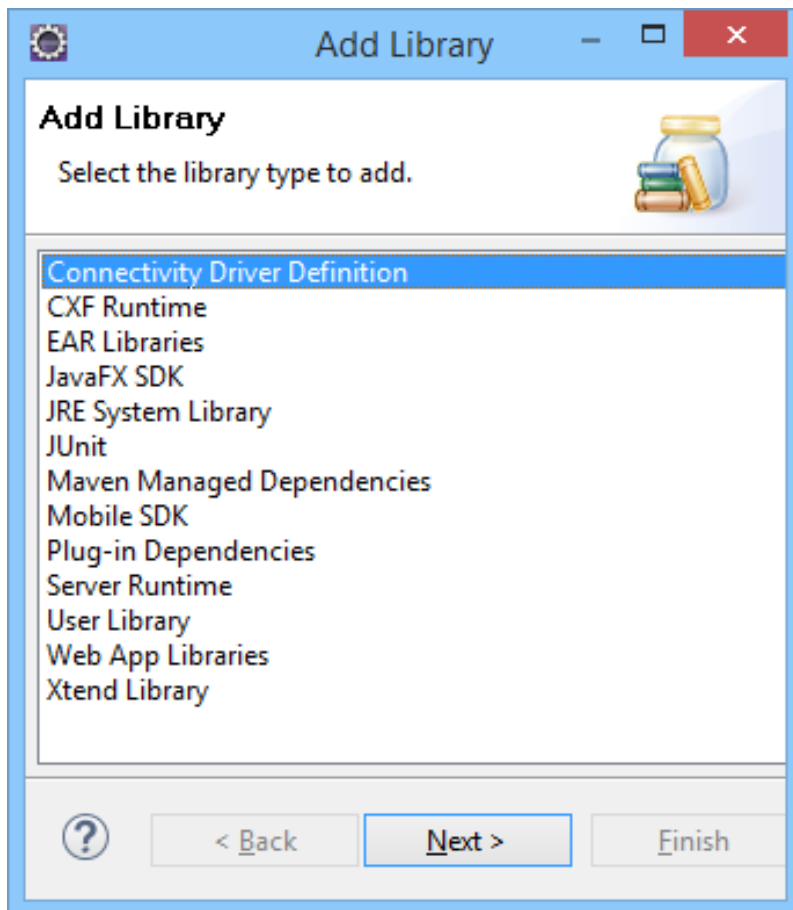
You should see:



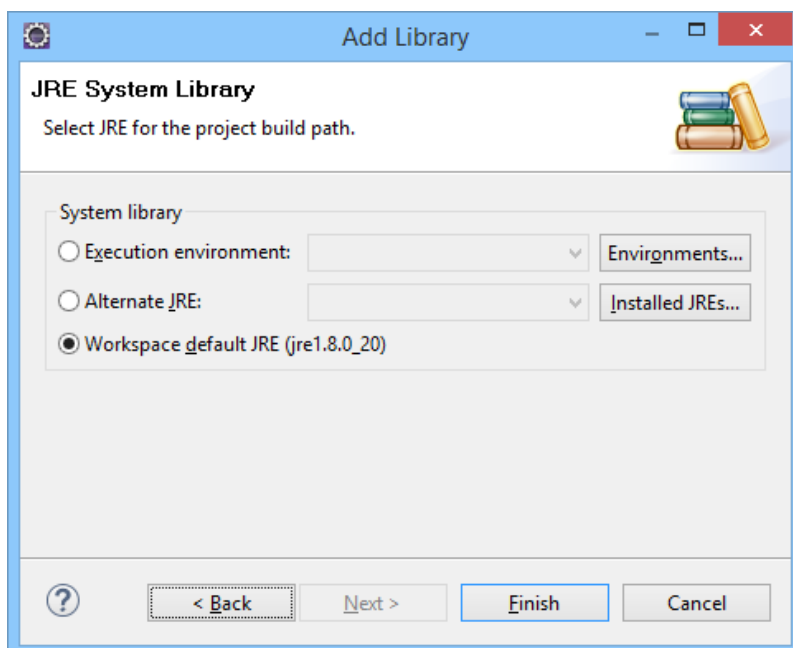
Click on **Remove** and Java will disappear. We need to put in Java 1.8.



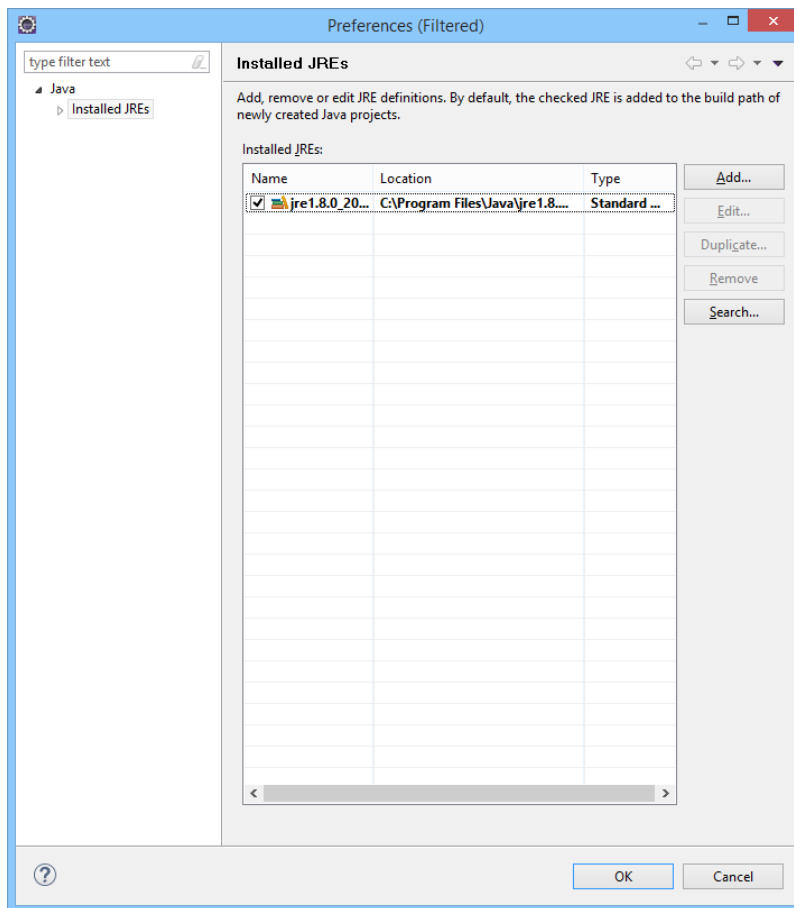
Select **Add Library** and you will see:



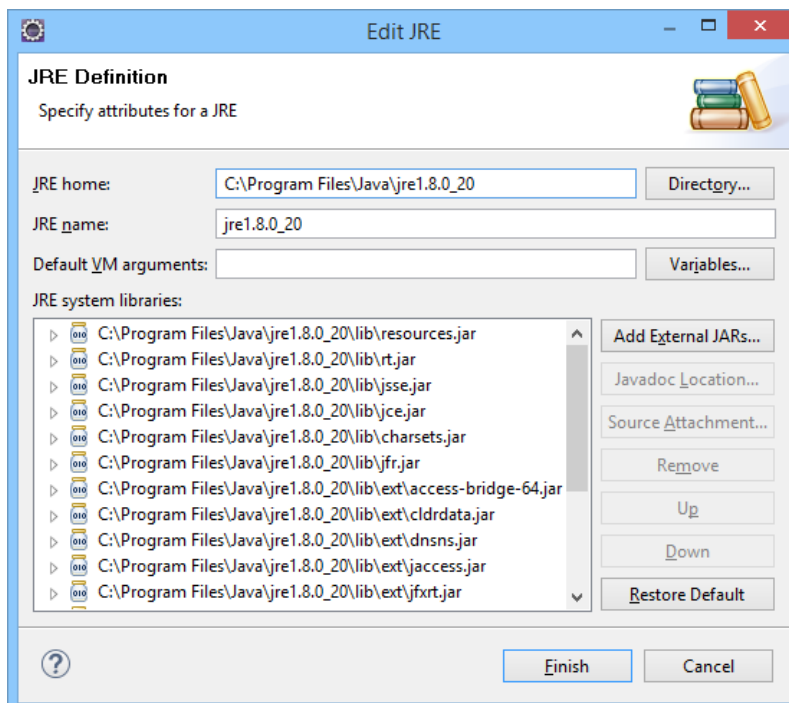
Select **JRE System Library** and click on **Next**. You should now see:



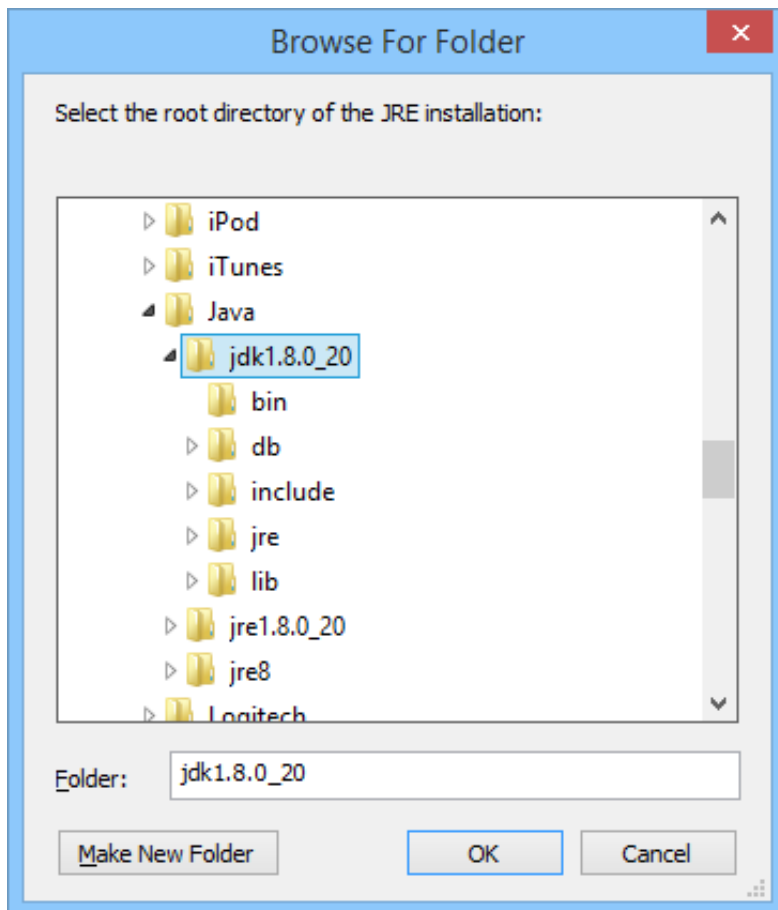
This is the dialog for a computer that only has Java 1.8 on it. It declares that the default JRE is a jre (jre1.8.0_20) which is wrong. It must be a JDK. Click on Installed JREs and you will get:



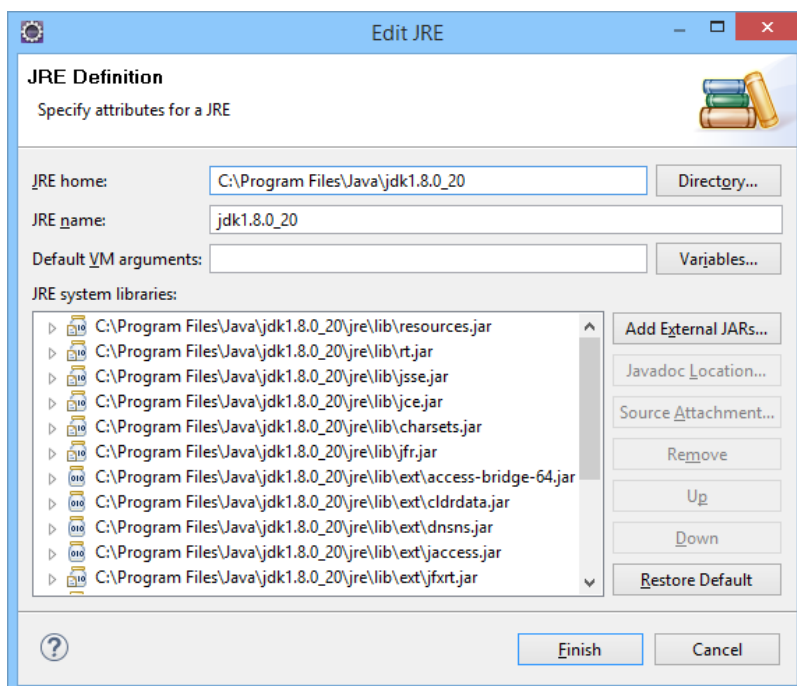
Click on the row and select **Edit**. This will let you point at the JDK instead of the JRE.



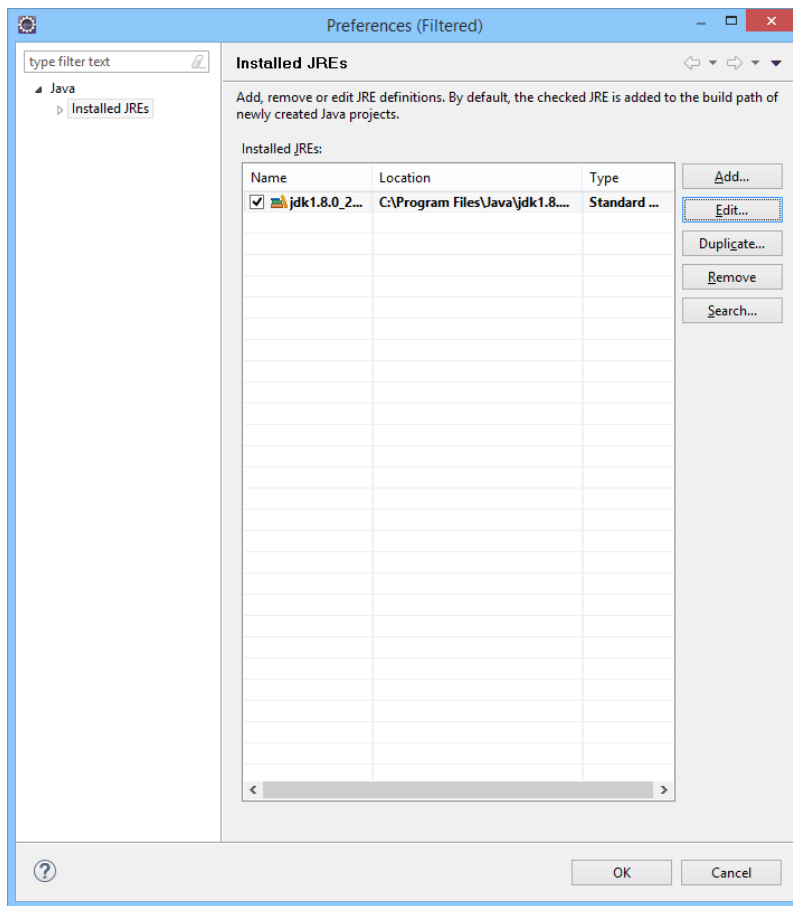
Now click on **Directory** and browse to the JDK. Here is what it looks like on my system:



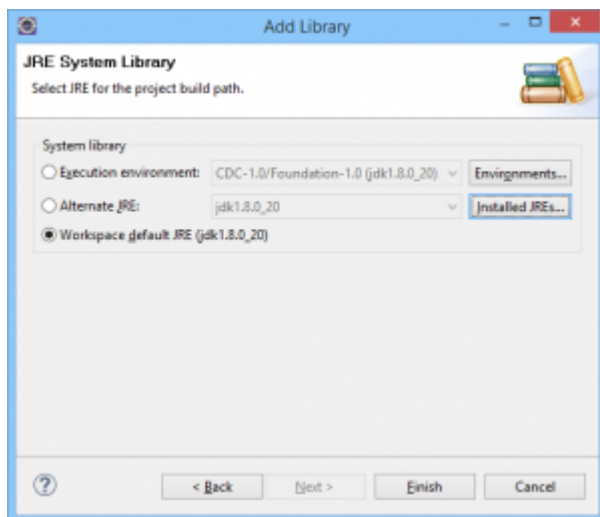
Click **OK** and back on the **Edit JRE** dialog change the **JRE name** to **jdk**.



Once this is resolved click on **Finish** and the **Preferences** should look like:

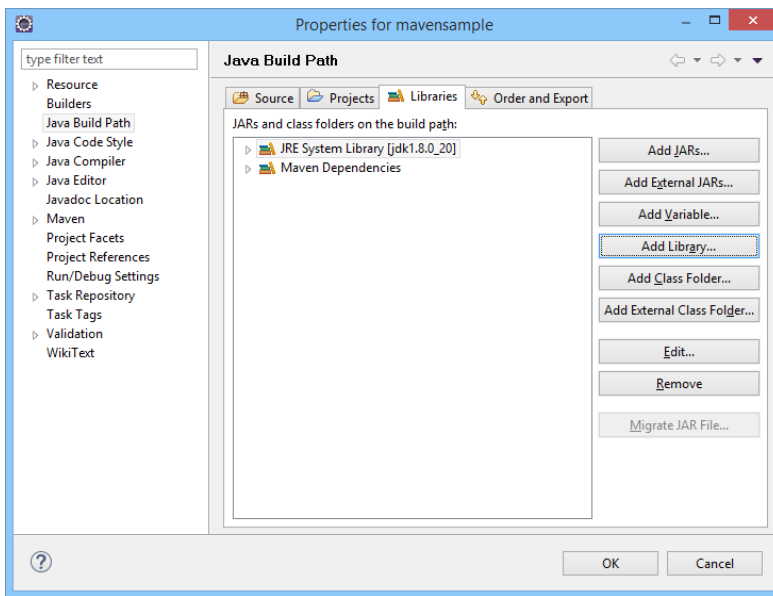


Click on **Ok** and the **Add Library** should look like:

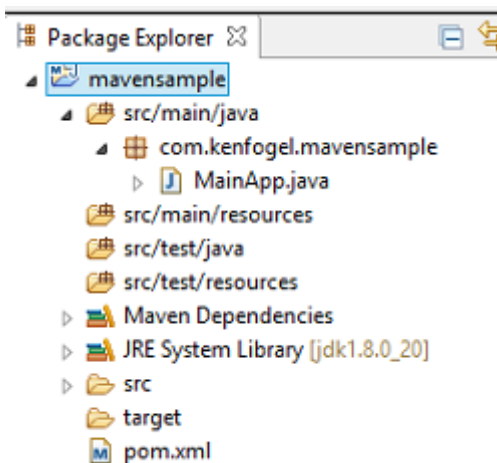


The **Java Build Path** will now be:



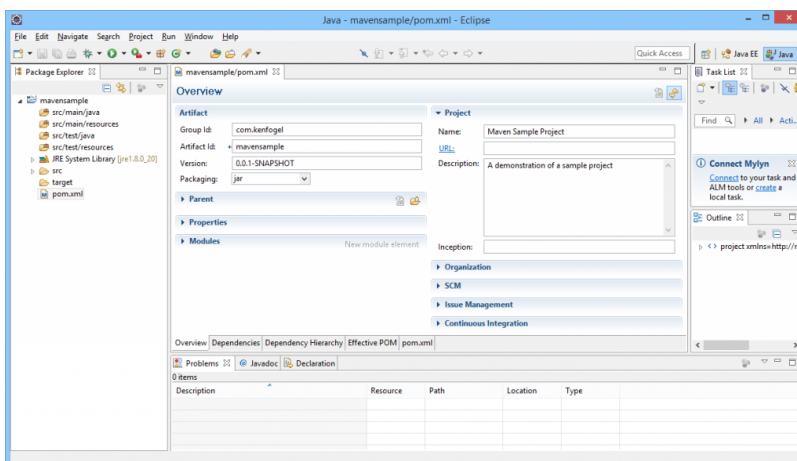


Click on **OK** and the **Package Explorer** should look like:

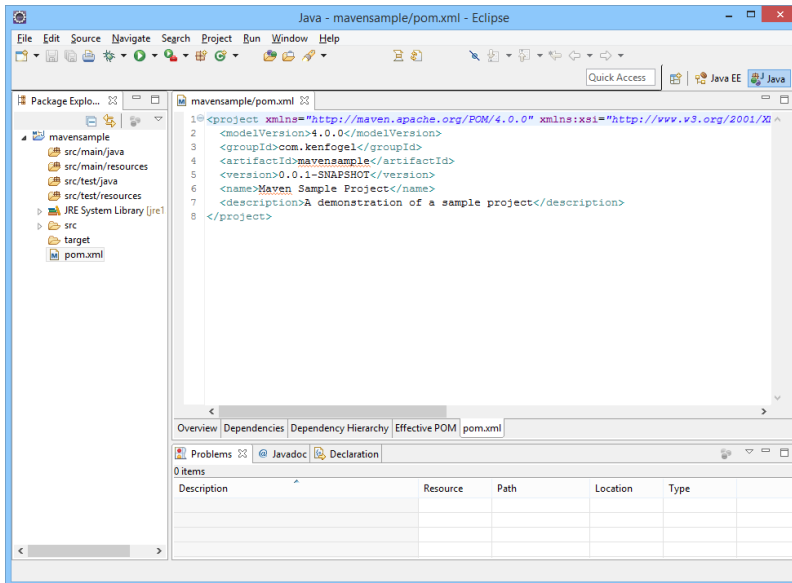


*** Note: An error in the first posting is now corrected. In the subsequent images the JRE System Library will show jre1.8.0_20 but if you are following these instructions it will be the correct jdk1.8.0_20.*

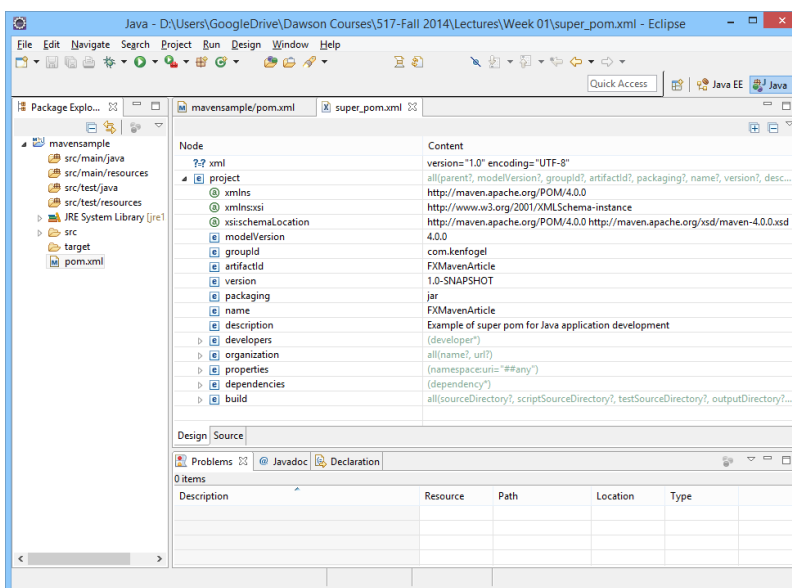
The last step is to update the **pom.xml** file. Double click on it and you will the **Overview** tab of the Maven pom file editor.



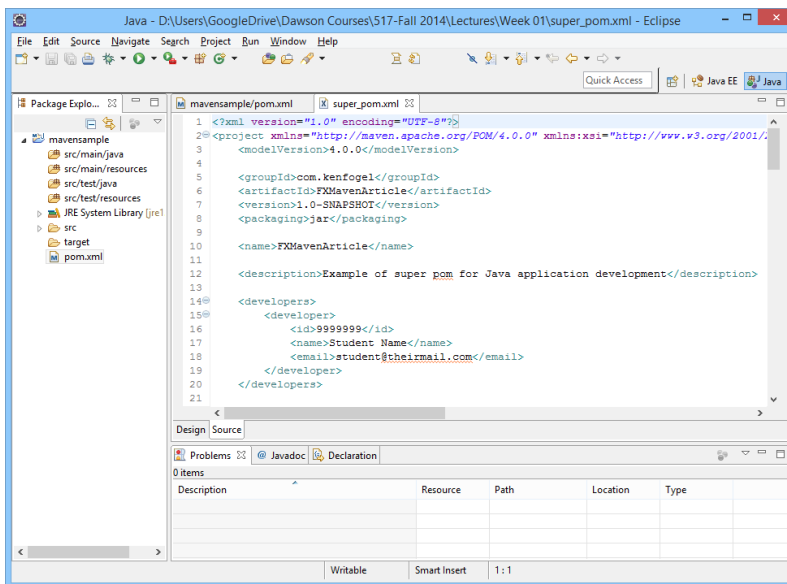
Switch to the **pom.xml** tab and you will see the raw file. If you close the **Task List** and **Outline** tabs on the right side of the workbench you will have a larger editor window.



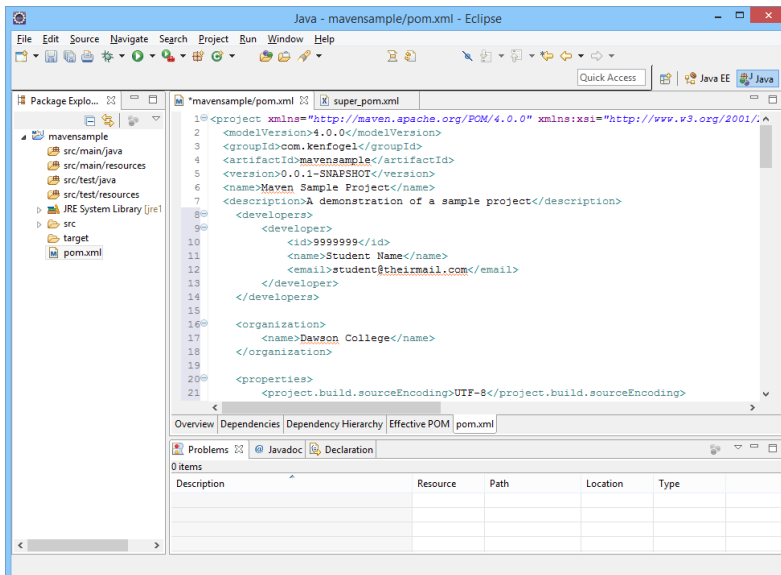
You will now need the Super Pom xml file. You can retrieve it at the end of the article at <http://netbeans.dzone.com/nb-class-maven-4-kf>. Copy it from the article and paste it in a file that you can use each time you create a new project. I keep mine in a file called **super_pom.xml**. In Eclipse open this file from **File -> Open File**. It will appear in the Eclipse XML Editor looking like:



Switch to the **Source** tab and you will see:

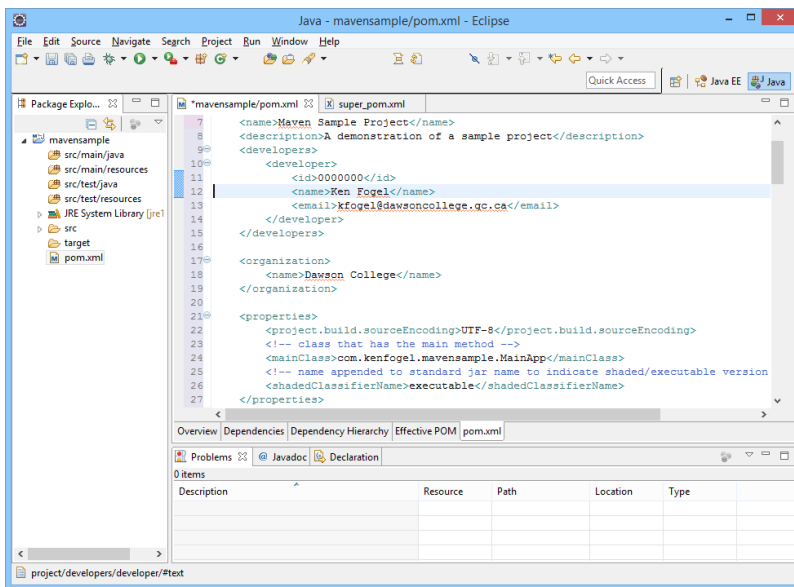


Copy the contents of the file starting at the **<developers>** tag till the end of the file. Go to the **pom.xml** file of the project and paste what you copied over the **</project>** tag. It should look like:




Go to the **Source** menu and select **Format** to clean up the indenting. If you have any errors in the **pom.xml** they are likely due to having a second **</project>** tag or that you pasted the entire **super_pom.xml** file into your own **pom.xml**. These can be easily fixed by deleting the duplicated information. Correct the values in the **<developer>** section. Go to the **<properties>** section and correct the **<mainClass>** tag to show the name of your class that will contain the main method including the full path.





You are now ready to write code.



 Eclipse, Java, Java 8, Maven

About Ken

I am the Program Coordinator and Chairperson of the Computer Science Technology program at Dawson College in Montreal, Canada. I am also a Program Consultant to and part-time instructor in the Computer Institute of Concordia University's School of Extended Learning.

[View all posts by Ken →](#)

