



OpenInfobutton Development Environment Setup Guide

Table of Contents

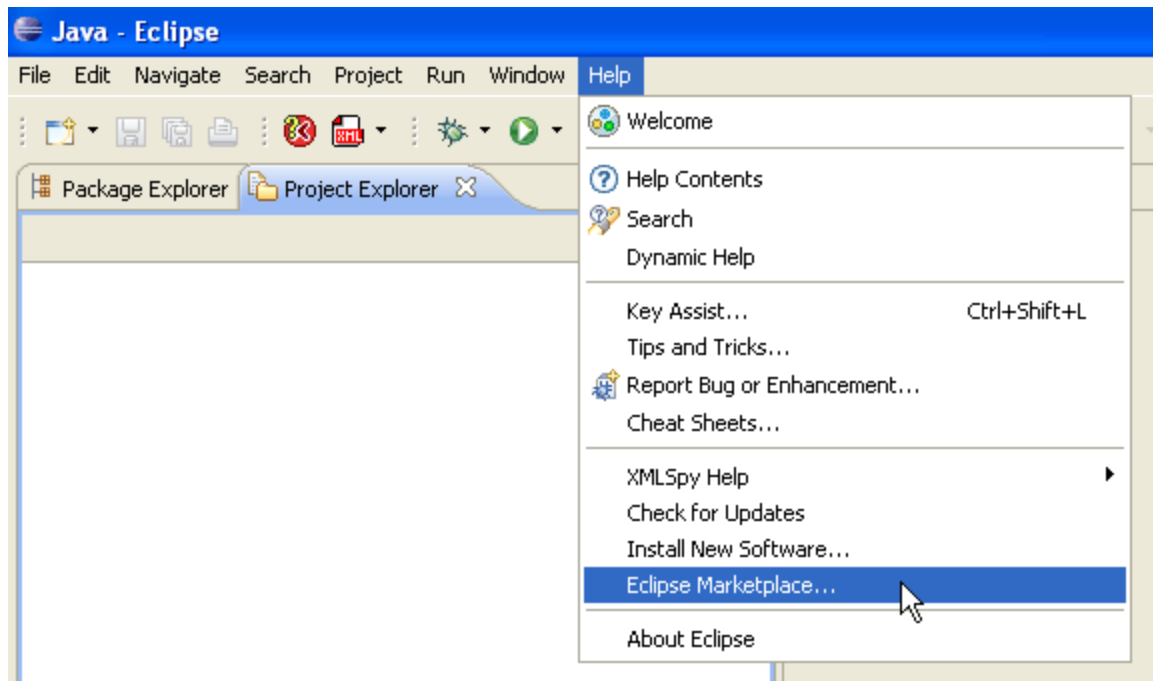
1	Software Requirements	1
2	Eclipse Plugins	1
2.1	Subclipse	2
2.2	M2Eclipse	7
3	Retrieval from SVN	16
4	Building	17
5	Running	20

1 Software Requirements

- Eclipse 3.6 Helios (or better) for Java EE
- Java 6 JDK
- Tomcat 6
- Maven 2.2 (has to be version 2.2)
- JAVA_HOME, CATALINA_HOME, and M2_HOME (Maven) environment variables all set
- PATH environment variable has the JAVA bin and the Tomcat bin directories

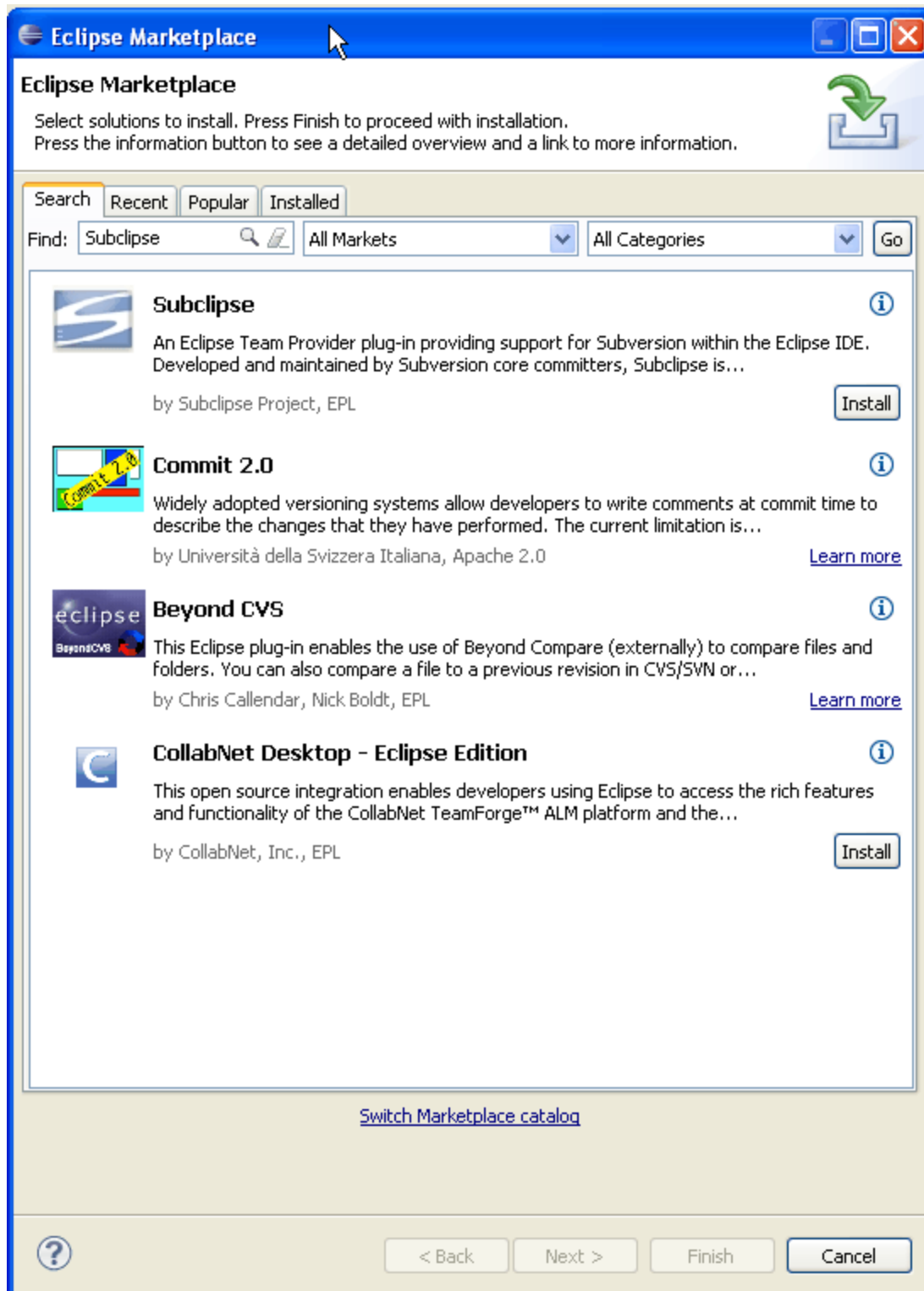
2 Eclipse Plugins

Before retrieving the code from source control, a couple of Eclipse plugins must be installed and configured. The newest version of Eclipse includes a built in plugin search tool called Eclipse Marketplace. This guide assumes usage of that tool, so make sure you're on Eclipse 3.6 or better.

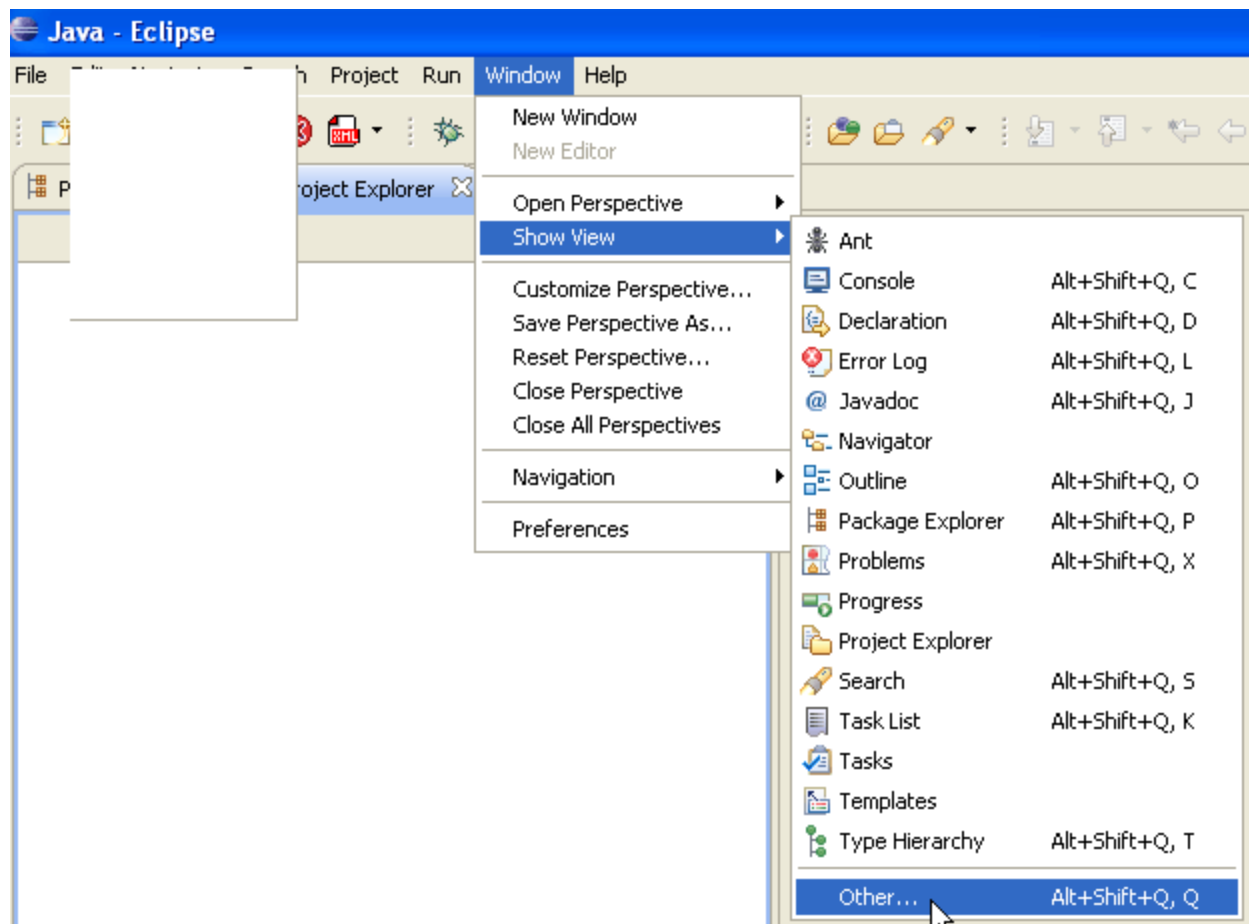


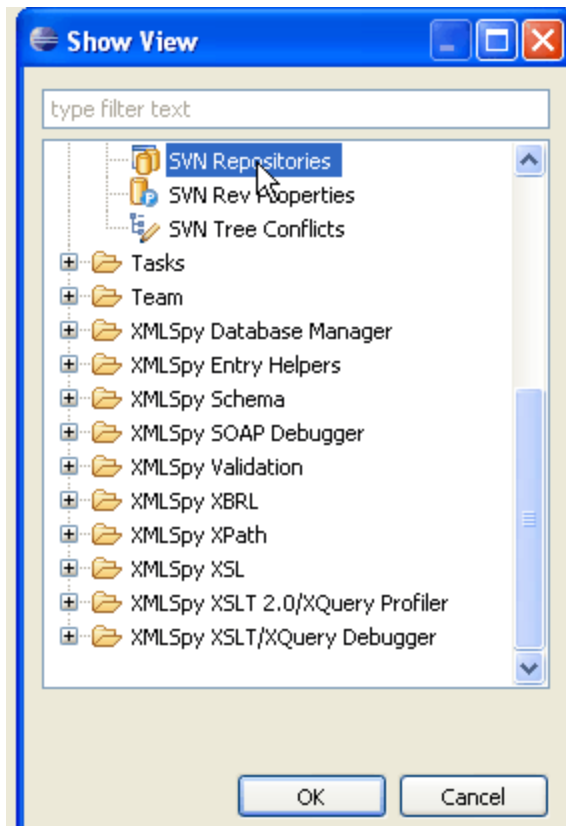
2.1 Subclipse

The first plugin is for connecting to and retrieving source code from the SVN repository. Open up the Eclipse Marketplace (found under Help) and search for “subclipse”. Click on the install button and continue clicking through the prompts until it’s installed.



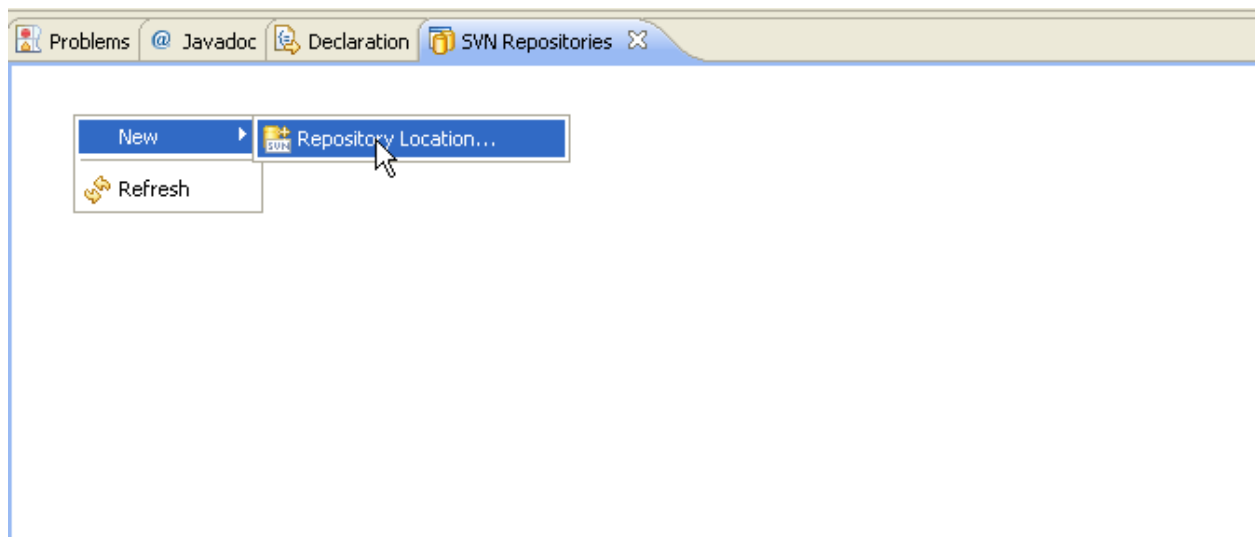
Once installed, it's time to add the source repository. Under Window, go to Show View and then click on Other. In the Show View menu, expand the SVN subcategory, select SVN Repositories, and click OK. This will add an SVN Repositories tab to the pane at the bottom of the main Eclipse window.

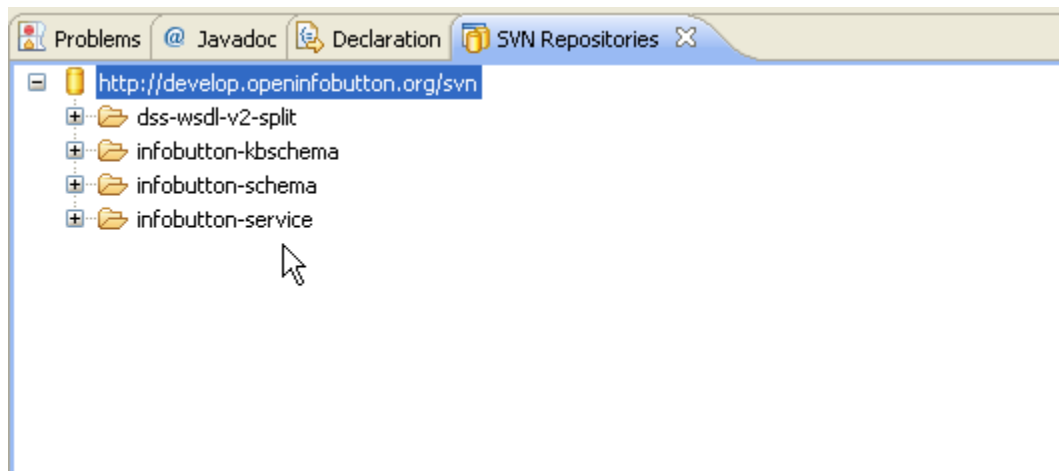
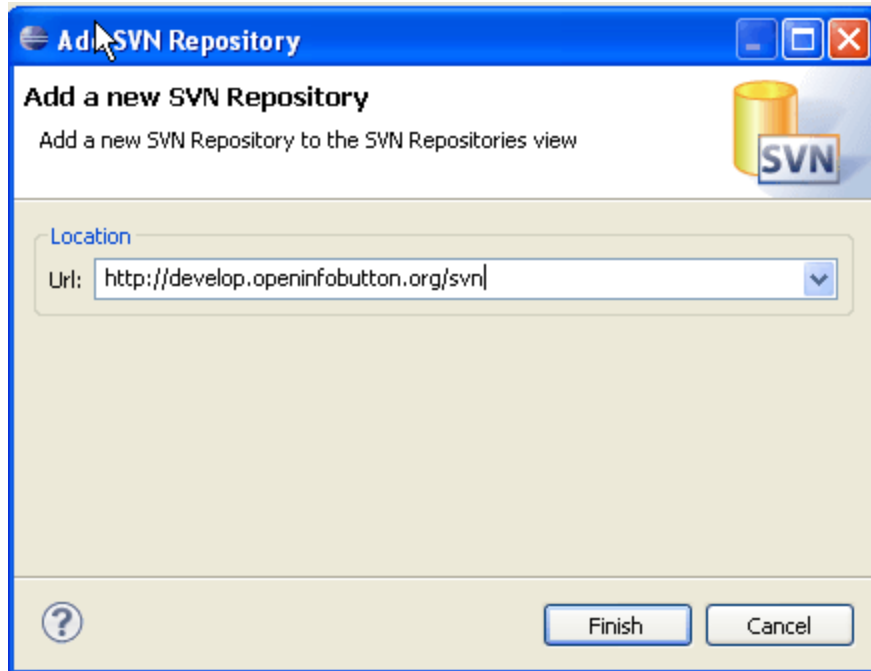




Select the SVN Repositories tab, right click on the empty space inside the pane, select New, and click on Repository Location. Enter <http://develop.openinfobutton.org/svn> and click Finish.

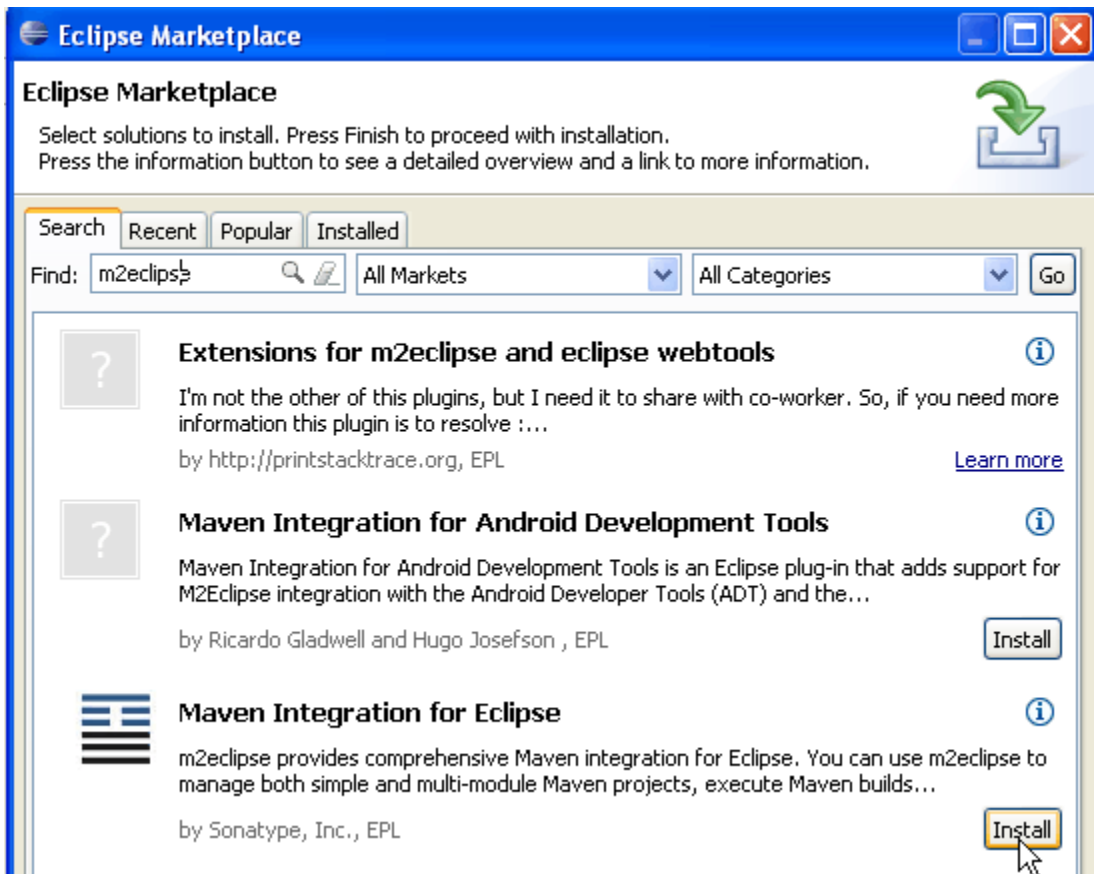
NOTE: you will be prompted for a user name and a password to access the OpenInfobutton repository. If you don't have an OpenInfobutton SVN repository account, you will need to request the OpenInfobutton team to create one for you.



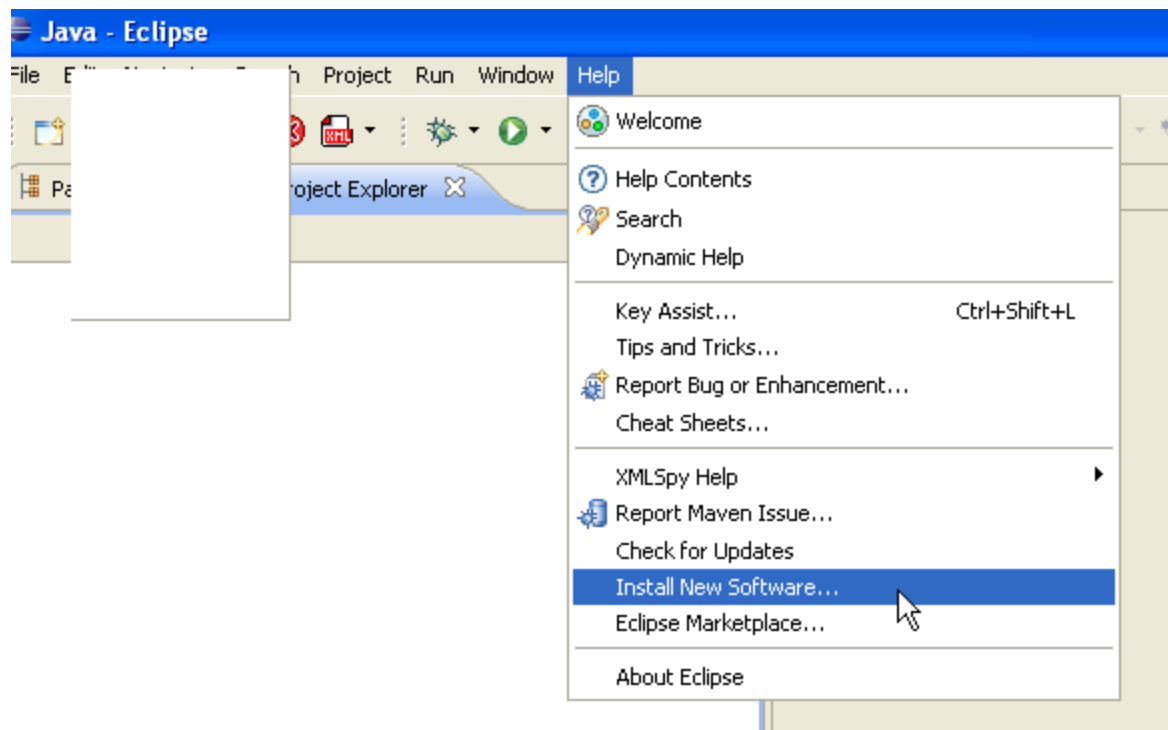


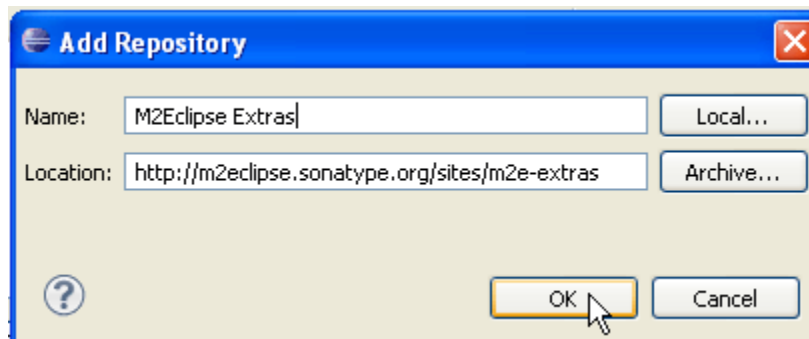
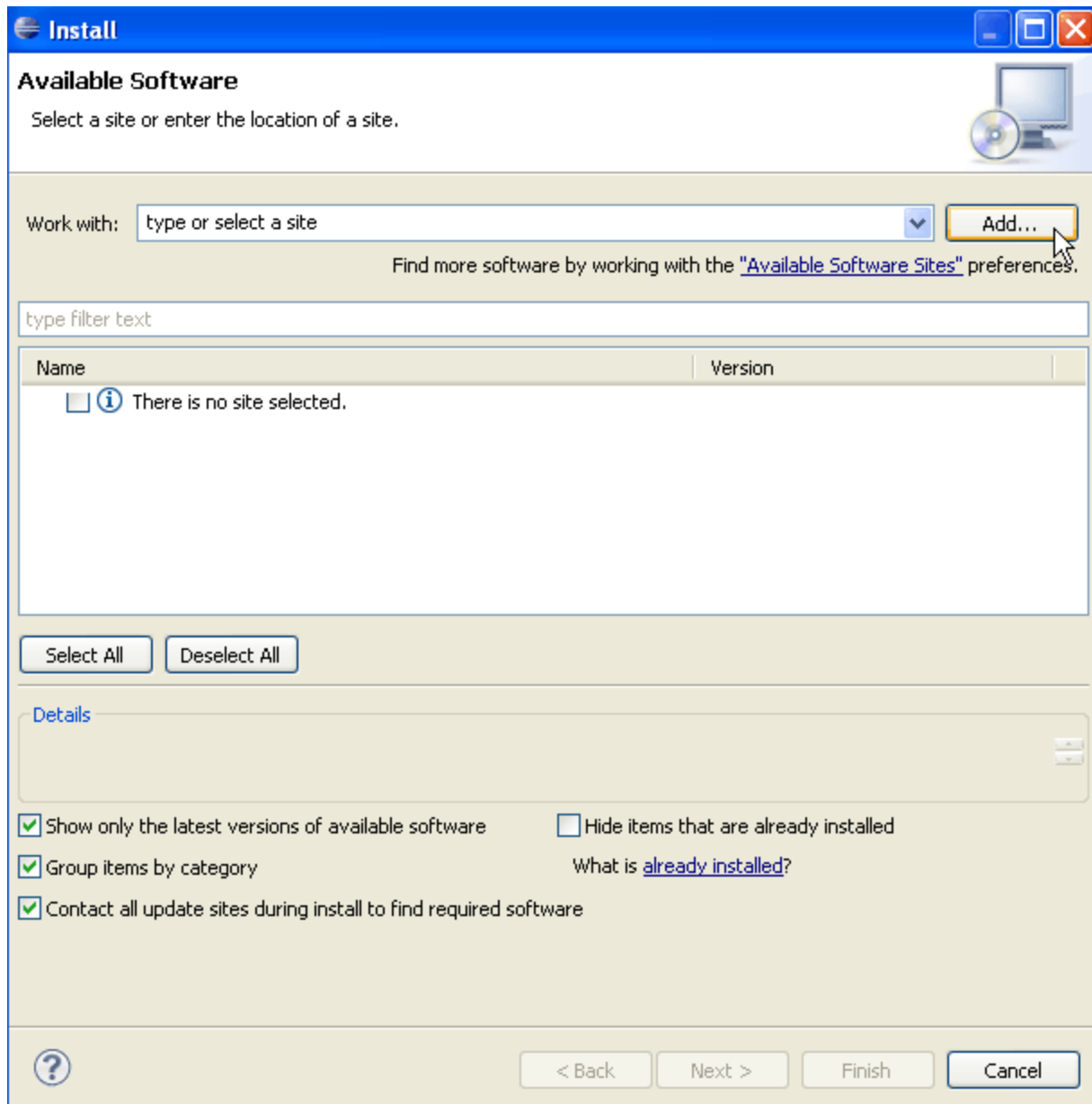
2.2 M2Eclipse

The next plugin is for dependency management and building with Maven. Open the Eclipse Marketplace again and search for “m2eclipse”. Install it just like Subclipse.



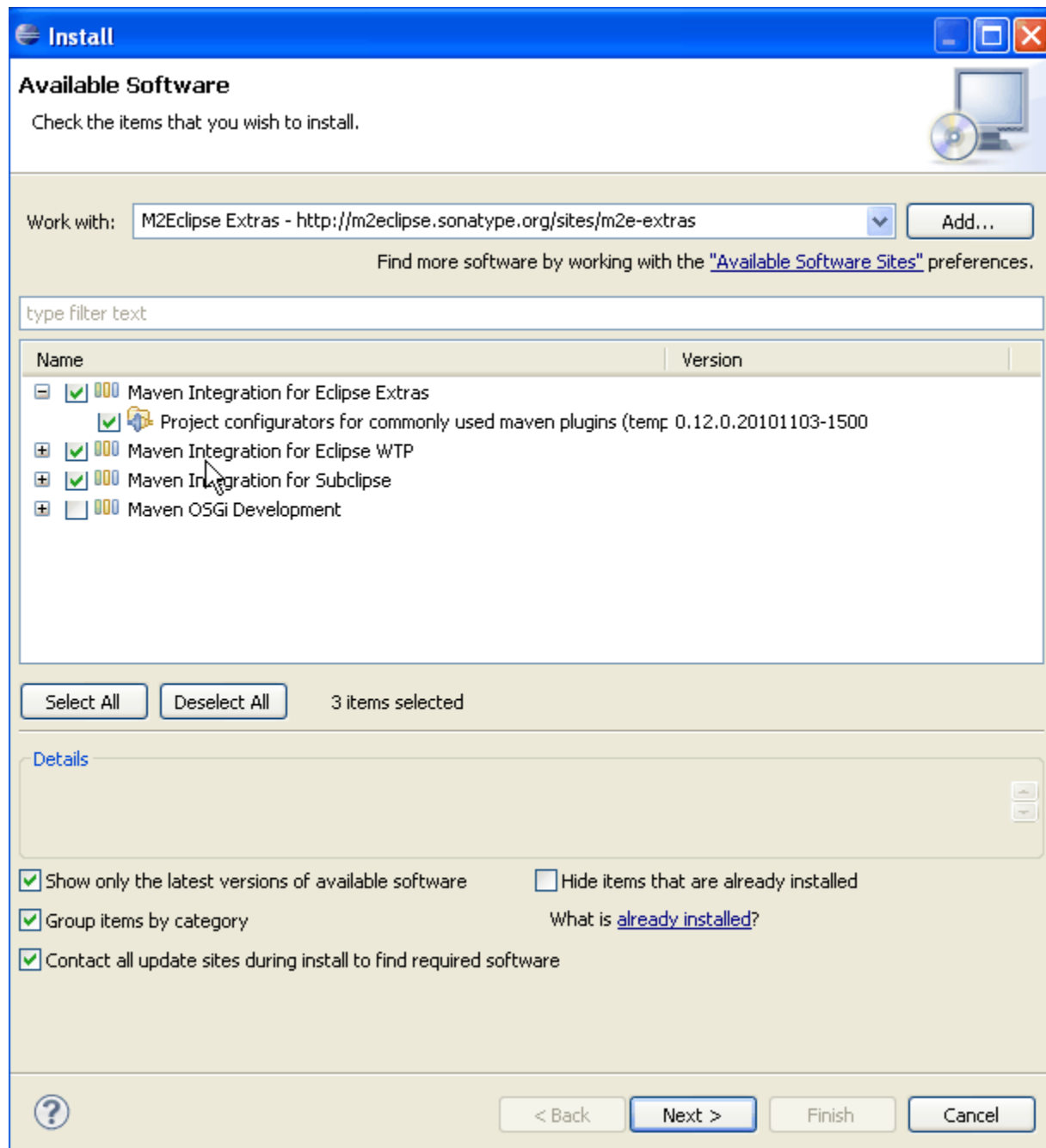
There are also a set of extras for M2Eclipse that need to be installed separately. These are not available via the Eclipse Marketplace, so they need to be installed the old fashioned way. Go to the Help menu, and click on Install New Software. Click Add, and then enter the following URL in the Location section of the Add Repository window, <http://m2eclipse.sonatype.org/sites/m2e-extras>. You can enter anything for name, then click OK.





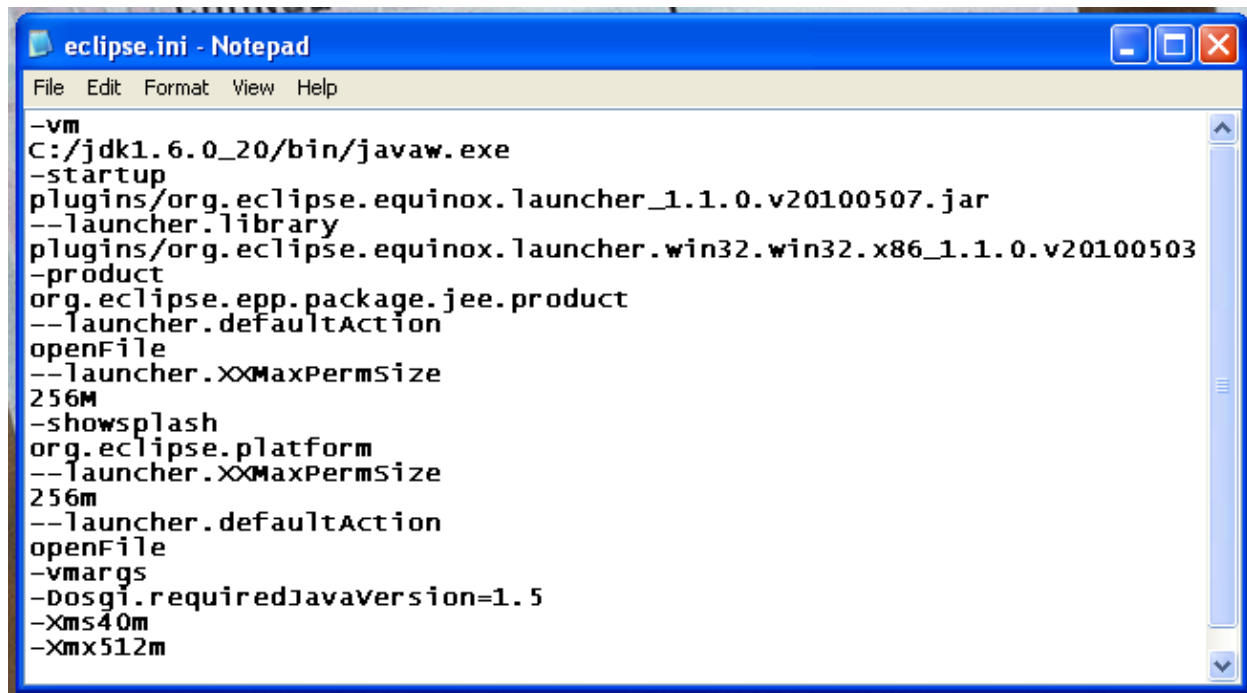
Now you should see a list of software with checkboxes next to them. Make sure everything but “Maven

OSGi Development” is checked and click Next to continue with installation.



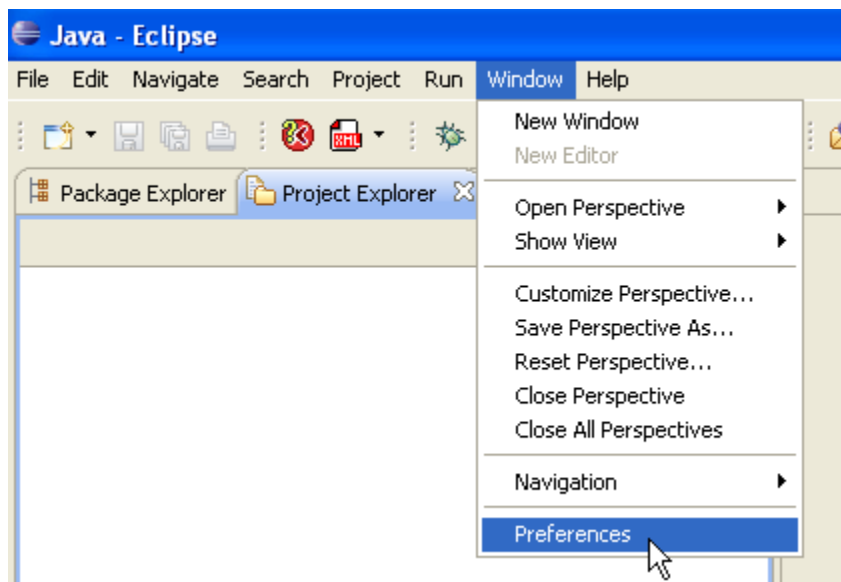
For Maven to work, we need to make sure that your Eclipse is running in a JDK. Go to your file system and navigate to your Eclipse directory. Should be something like C:\Program Files\eclipse . Open the file “eclipse.ini” in your preferred text editor, and add the follow lines to the top of the file,

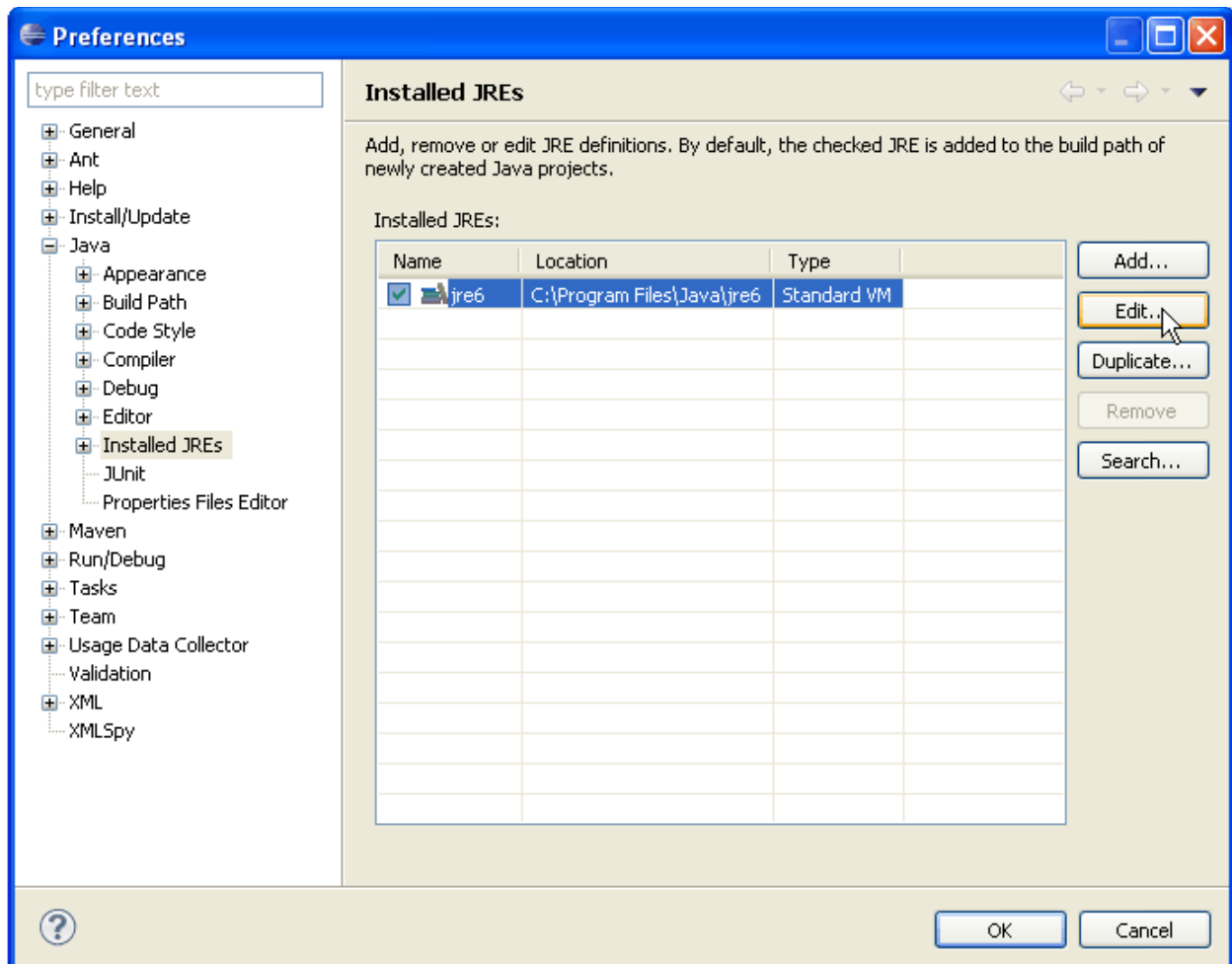
```
-vm
<path to your Java 6 JDK>\bin\javaw.exe
```

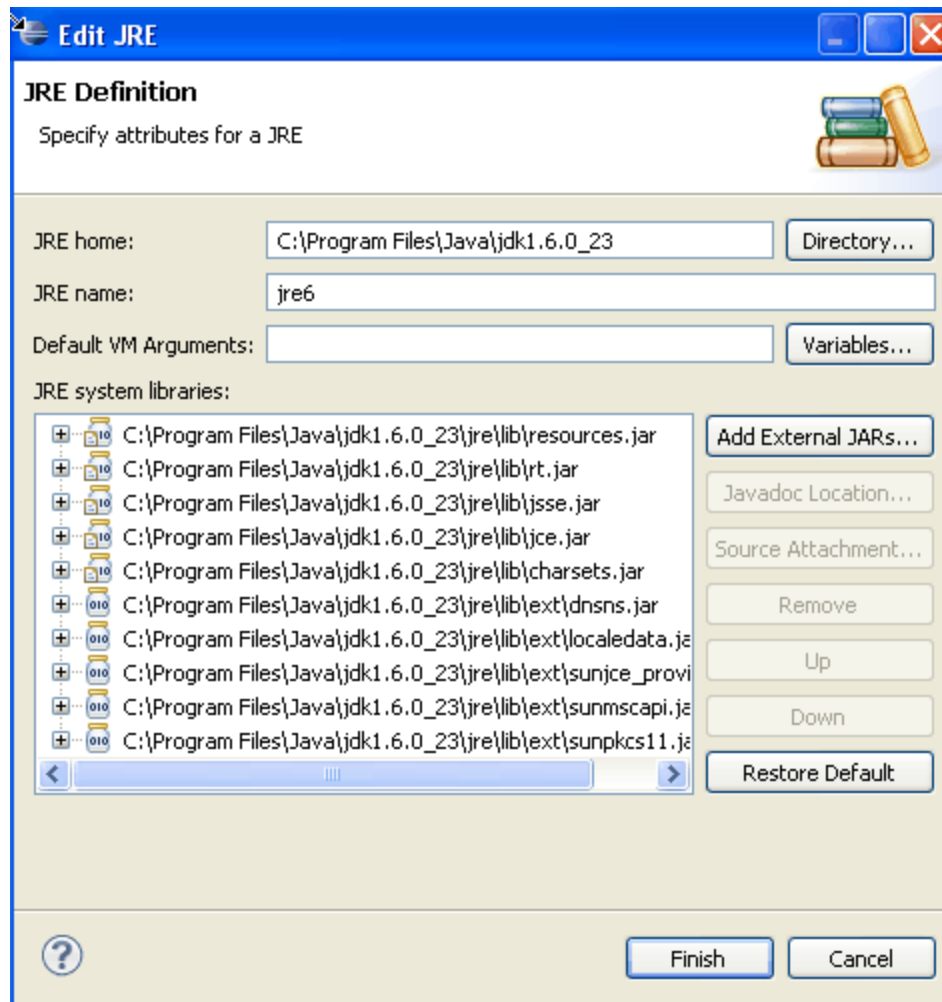
A screenshot of a Notepad window titled 'eclipse.ini - Notepad'. The window contains the following configuration lines for the Eclipse IDE:

```
-vm
C:/jdk1.6.0_20/bin/javaw.exe
-startup
plugins/org.eclipse.equinox.launcher_1.1.0.v20100507.jar
--launcher.library
plugins/org.eclipse.equinox.launcher.win32.win32.x86_1.1.0.v20100503
-product
org.eclipse.epp.package.jee.product
--launcher.defaultAction
openFile
--launcher.XXMaxPermSize
256M
-showsplash
org.eclipse.platform
--launcher.XXMaxPermSize
256m
--launcher.defaultAction
openFile
-vmargs
-Dosgi.requiredJavaVersion=1.5
-Xms40m
-Xmx512m
```

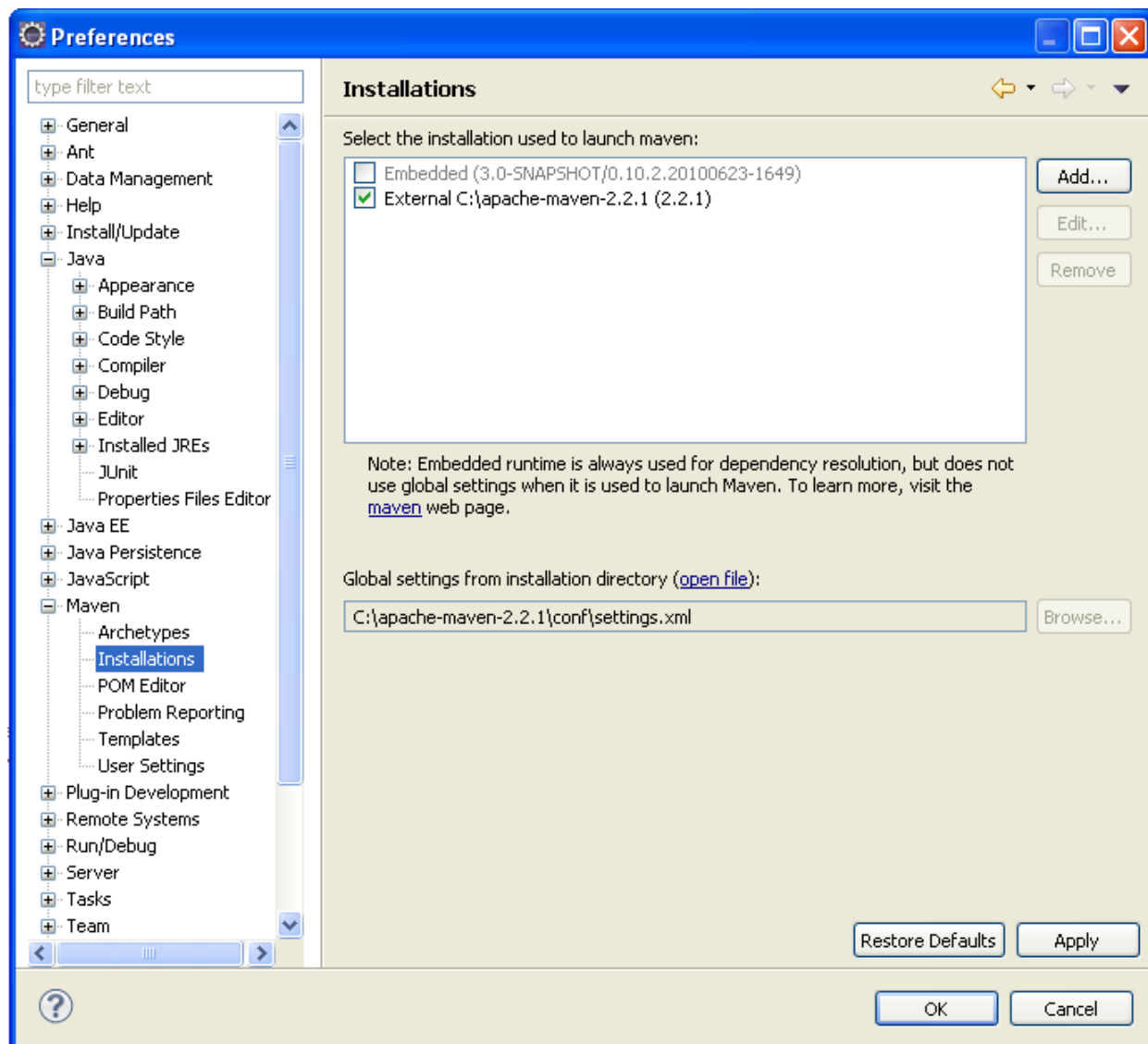
Make sure to restart Eclipse after doing this. You should also go to Window, Preferences, Java, and click on Installed JREs to check that default JRE is pointing to your Java 6 JDK rather than Eclipse's internal JRE. If not, select it, click Edit, and make change the JRE home path to point to your JDK.







Once installed, go to Window, Preferences, and expand the Maven category. Then select Installations and click on Add. Navigate to your Maven install directory and click OK.



The final step for configuring Maven is creating settings.xml file. This file is for any global settings that apply to all your Maven projects. First, navigate to your Maven repository in your file system. If you're using Windows XP, it should be located at,

C:\Documents and Settings\<username>\.m2 \

Create a file in this directory called settings.xml and copy the following into it.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
    http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository>C:\Documents and
Settings\ai28\.m2\repository</localRepository>
  <interactiveMode/>
  <usePluginRegistry/>
  <offline/>
  <pluginGroups/>
  <servers/>
</settings>
```

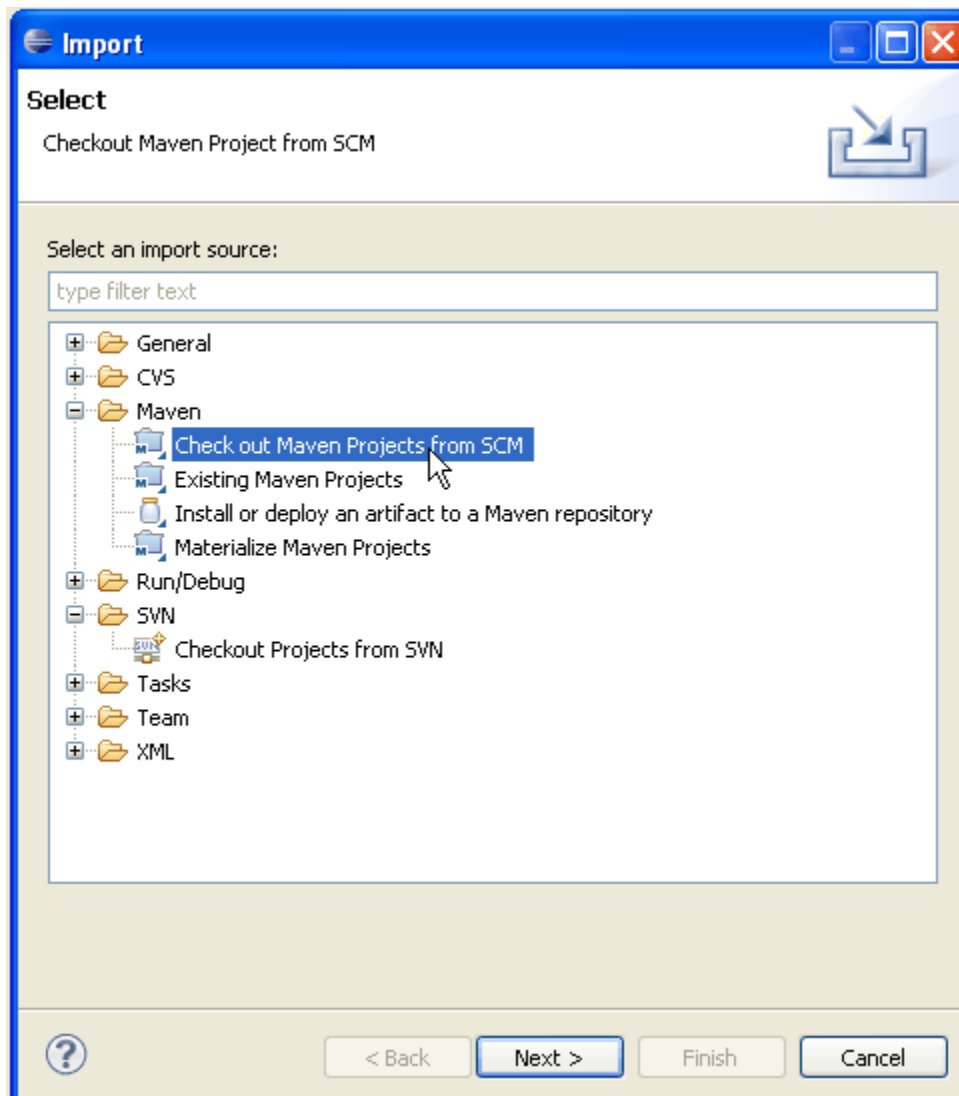
Make sure to edit the path in the <localRepository> element to reflect your personal file system. In most cases, all you should need to change is the user name. If you connect to the internet via a proxy, you also need to add the following element,

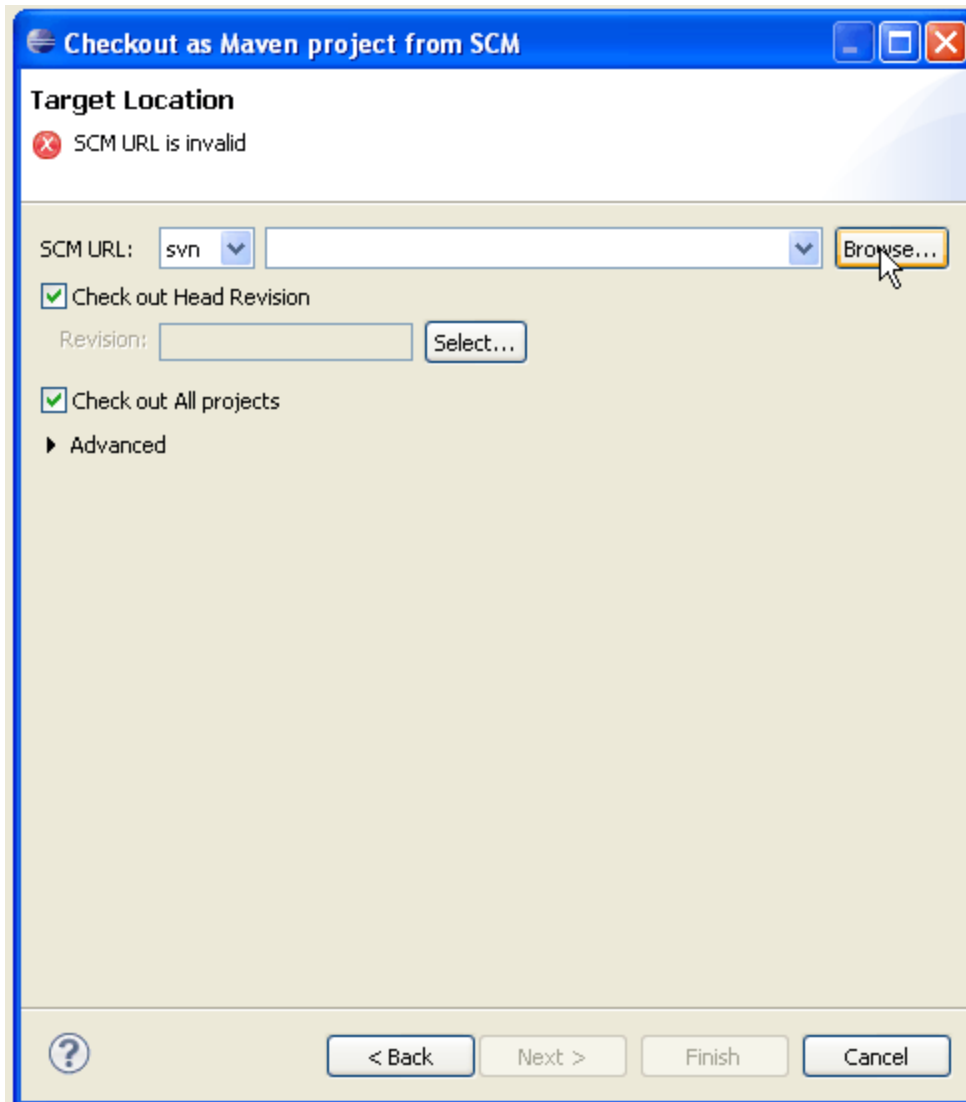
```
..
<proxies>
  <proxy>
    <active>true</active>
    <protocol>http</protocol>
    <host>proxy.somewhere.com</host>
    <port>8080</port>
    <username>proxyuser</username>
    <password>somepassword</password>
    <nonProxyHosts>www.google.com|*.somewhere.com</nonProxyHosts>
  </proxy>
</proxies>
..
```

and fill in the correct information for your proxy server. You may need to restart Eclipse before any changes you make to the settings.xml file are reflected.

3 Retrieval from SVN

Since this project uses Maven, projects need to be imported in a very specific manner. Go the File and click Import. Under Maven, select “Check out Maven Projects from SCM” and click Next. In the dropdown menu next to SCM URL, type or select svn then click Browse. You should then see the source repository we added earlier. Expand it and select dss-wsdl-v2-split , then click OK. Finally, click Finish and wait for Eclipse to add the project. Once added, go back and repeat this procedure for infobutton-kbschema, infobutton-schema, infobutton-service, and XMLTransformationService2Client. Now all 5 projects should be visible in the Project Explorer on the left hand side of the Eclipse window.



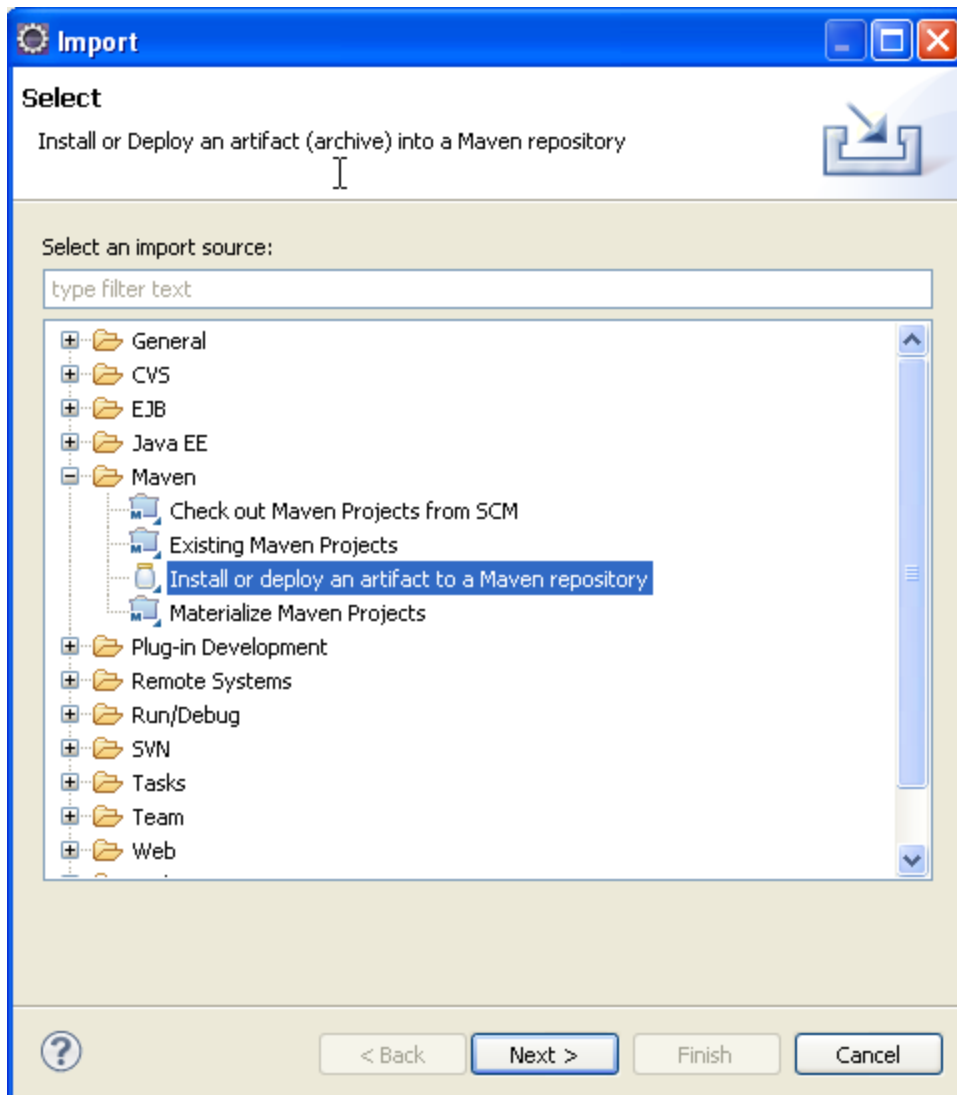


4 Building

Before building, there is one dependency that must be manually added to your local Maven repository because it isn't available in any public repository. This is the Microsoft SQL JDBC Driver used by the Apelon DTS terminology server. The following instructions assume you installed Apelon DTS on a Microsoft SQL Server 2008. First, download the driver from the following link if you don't already have it.

<http://www.microsoft.com/downloads/en/details.aspx?FamilyID=99b21b65-e98f-4a61-b811-19912601fdc9&displaylang=en> .

Run the executable to extract the sqljdbc.jar file to a location on your file system. Next, click File-Import on Eclipse and choose "Maven/Install or deploy an artifact to a Maven repository" as shown below.



Fill out the fields as follows and click Finish. The screen should look like the screen shot below.

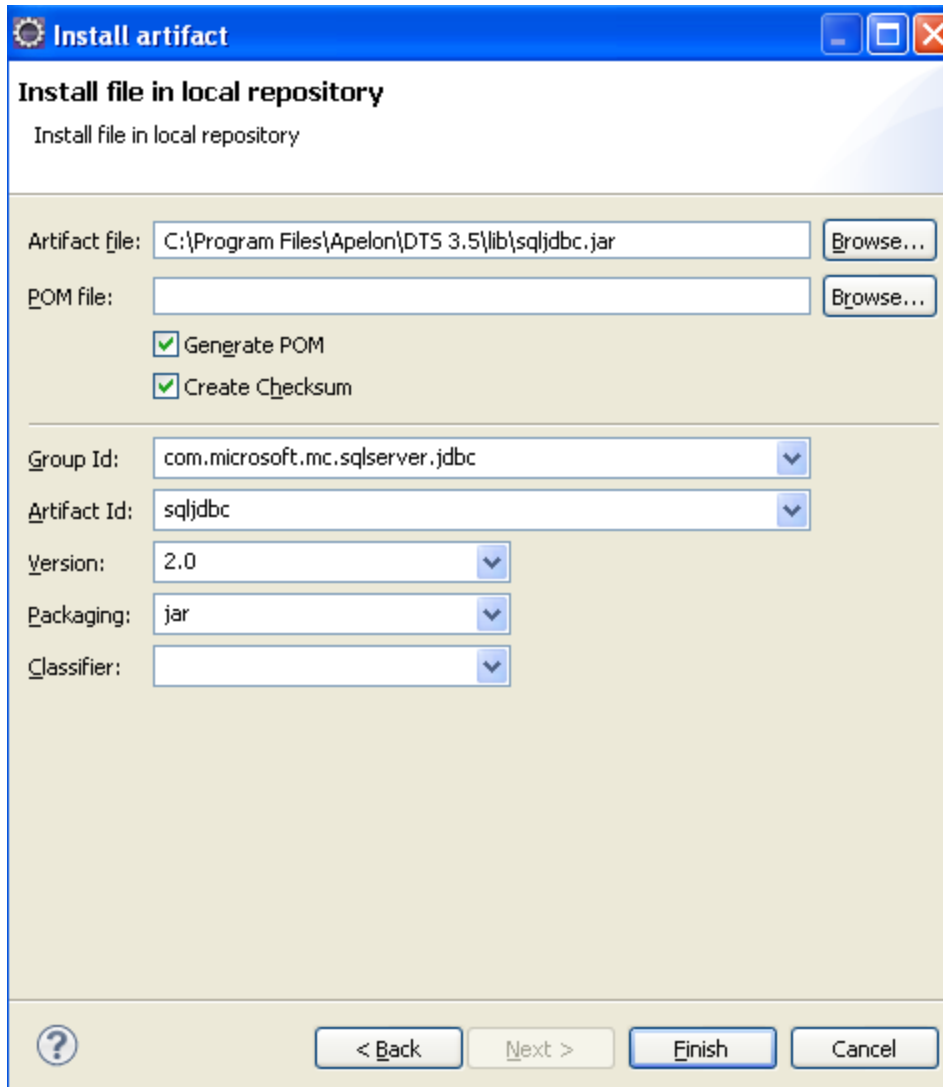
Artifact File: choose the location of the sqljdbc.jar file

GroupId: com.microsoft.mc.sqlserver.jdbc

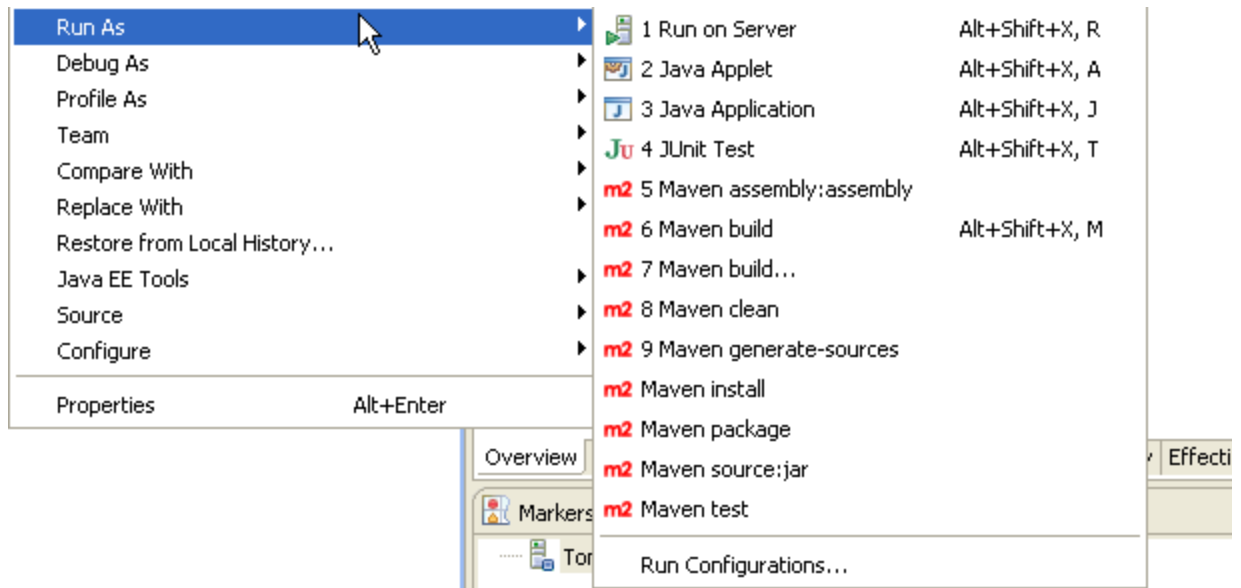
ArtifactId: sqljdbc

Version: 2.0

Packaging: jar

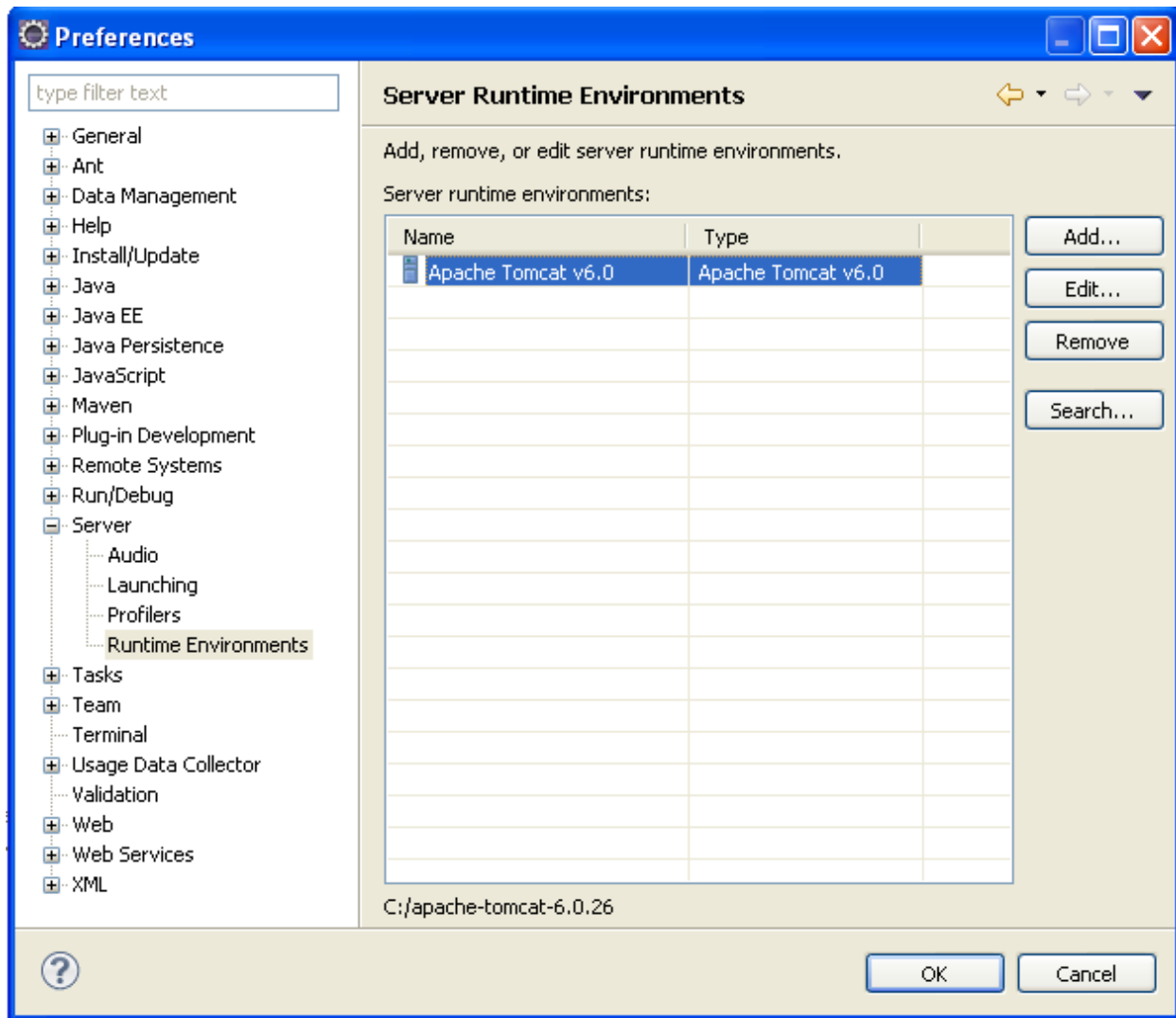


For building, infobutton-service includes the other four projects as dependencies. Therefore, they must be built first before attempting to build infobutton-service. The order doesn't matter, as long as all four are built before infobutton-service. Right click on one of them and select Run As, then click on "Maven clean". Wait for this to finish, then select Run As, Maven install. Repeat this for all the projects, again, saving infobutton-service for last.



5 Running

So now that the application is built and ready to run, it's time to configure the Tomcat server. Go to Window and click on Preferences. Expand the Server category and click on Runtime Environments. Click add, select Apache 6, and then click Next. Make sure the installation directory is pointing to your local Tomcat installation and then click Finish.



The final step is to setup your knowledge resource profiles and terminology inference profiles directory. Knowledge resource profiles consist of the OpenInfobutton knowledge base. These profiles contain instructions that configure the different resources that OpenInfobutton will access to. These profiles need to be copied to a directory called profiles on your c: drive.

First go the Project Explorer in Eclipse and expand the infobutton-kbschema project. Open src/main/resources and then ResourceProfiles. Select and copy all the XML profile instances in that directory. Now go out to your local filesystem and create directory called C:\profiles. Paste all those copied profiles into that directory.

Also in the infobutton-kbschema project, src/main/resources, select TerminologyInferences. Select and copy all the XML profile instances in that directory. Now go out to your local filesystem and create directory called C:\terminologyInferences. Paste all those copied terminology inference files into that directory.

Now the application can run. Go back to Eclipse, right click on infobutton-service in the Project Explorer and select Run As, Run on Server. Make sure to select the Tomcat server we configured earlier. Once

running, try a test query.

