

Instructions to install Eclipse Lyo

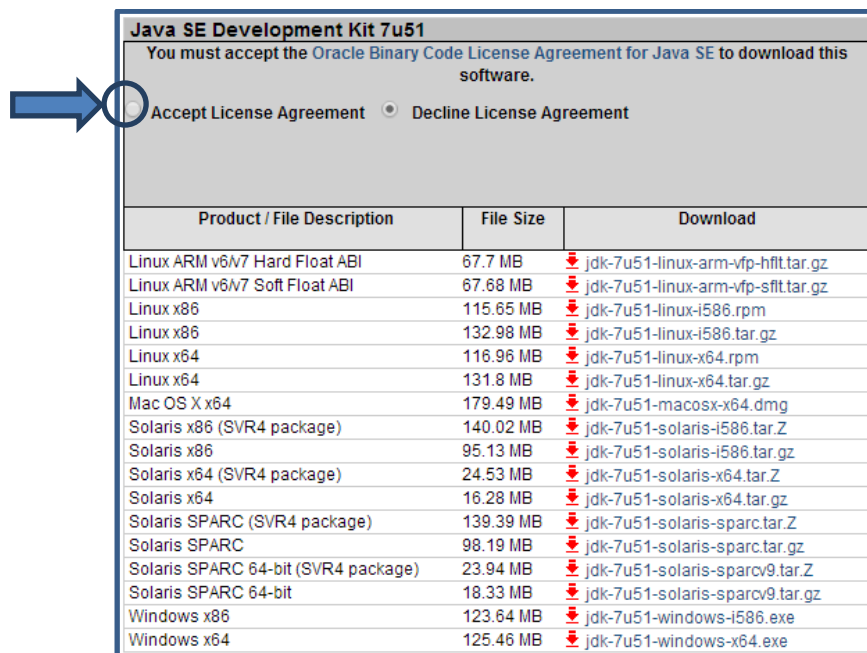
By Axel Reichwein, March 30 2014

1. Installing the Java SE Development Kit (JDK) 7

1. You can check if you have already installed the Java SE Development Kit 7 on your Windows machine by typing "java -version" in the command prompt window. Launch the command prompt by clicking on Start->All Programs-> Accessories-> Command Prompt. If Java SE Development Kit 7 is installed on your computer, the displayed java version will start with 1.7 as displayed in the figure below. You can also have other Java versions on your computer. No need to uninstall previous versions.

```
C:\Users\CParedis>java -version
java version "1.7.0_06"
Java(TM) SE Runtime Environment (build 1.7.0_06-b24)
Java HotSpot(TM) 64-Bit Server VM (build 23.2-b09, mixed mode)
```

2. If you do not the Java SE Development Kit 7 installed, go to the following web page to install Java SE JDK 7 for different platforms:
<http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html>
3. Accept the license agreement for Java SE Development Kit 7u51, as shown in the figure below



Java SE Development Kit 7u51
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

☒ Accept License Agreement ☐ Decline License Agreement

Product / File Description	File Size	Download
Linux ARM v6/v7 Hard Float ABI	67.7 MB	jdk-7u51-linux-arm-vfp-hflt.tar.gz
Linux ARM v6/v7 Soft Float ABI	67.68 MB	jdk-7u51-linux-arm-vfp-sflt.tar.gz
Linux x86	115.65 MB	jdk-7u51-linux-i586.rpm
Linux x86	132.98 MB	jdk-7u51-linux-i586.tar.gz
Linux x64	116.96 MB	jdk-7u51-linux-x64.rpm
Linux x64	131.8 MB	jdk-7u51-linux-x64.tar.gz
Mac OS X x64	179.49 MB	jdk-7u51-macosx-x64.dmg
Solaris x86 (SVR4 package)	140.02 MB	jdk-7u51-solaris-i586.tar.Z
Solaris x86	95.13 MB	jdk-7u51-solaris-i586.tar.gz
Solaris x64 (SVR4 package)	24.53 MB	jdk-7u51-solaris-x64.tar.Z
Solaris x64	16.28 MB	jdk-7u51-solaris-x64.tar.gz
Solaris SPARC (SVR4 package)	139.39 MB	jdk-7u51-solaris-sparc.tar.Z
Solaris SPARC	98.19 MB	jdk-7u51-solaris-sparc.tar.gz
Solaris SPARC 64-bit (SVR4 package)	23.94 MB	jdk-7u51-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	18.33 MB	jdk-7u51-solaris-sparcv9.tar.gz
Windows x86	123.64 MB	jdk-7u51-windows-i586.exe
Windows x64	125.46 MB	jdk-7u51-windows-x64.exe

4. Choose the JDK for your platform (e.g. Windows)

Note: The JDK includes the Java Virtual Machine.

The next steps are partly based on the OSLC tutorial instructions (<http://open-services.net/resources/tutorials/integrating-products-with-oslc/running-the-examples/>)

2. Downloading and installing Eclipse

1. Download the Eclipse IDE for Java EE Developers, version Kepler 4.3.2. If you have a windows 64 bit machine, then download Eclipse at following web page:
http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/kepler/SR2/eclipse-cpp-kepler-SR2-win32-x86_64.zip
2. And choose your mirror as in the figure below



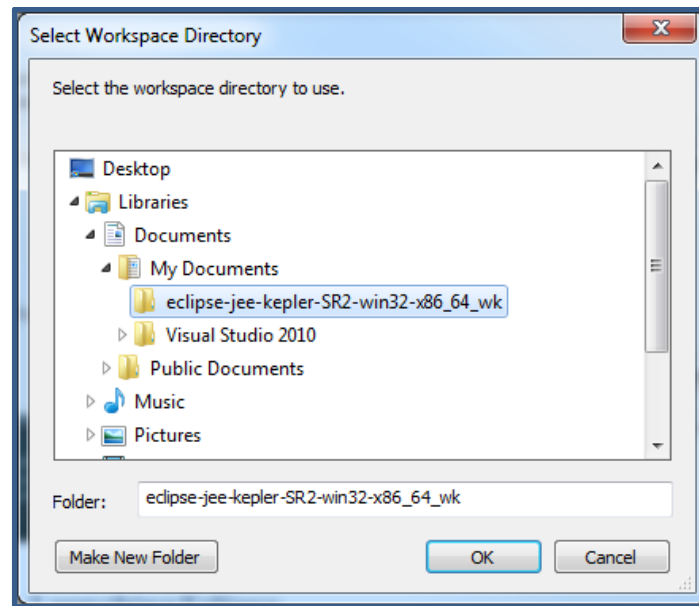
If you do not have a Windows 64bit machine, download Eclipse at this web page:

<http://www.eclipse.org/downloads/>

3. Unzip the downloaded Eclipse folder and place it at the location of your choice (e.g. under C:\Program Files)
4. Create a folder which will be the workspace for your Eclipse projects. Create for example a folder named "eclipse-jee-kepler-SR2-win32-x86_64_wk" in your Documents directory.

3. Launching Eclipse

1. Go to your Eclipse IDE installation directory, open the eclipse folder and execute the eclipse.exe application
2. Select the workspace folder for your Eclipse projects (the new folder you have created) as shown in the figure below

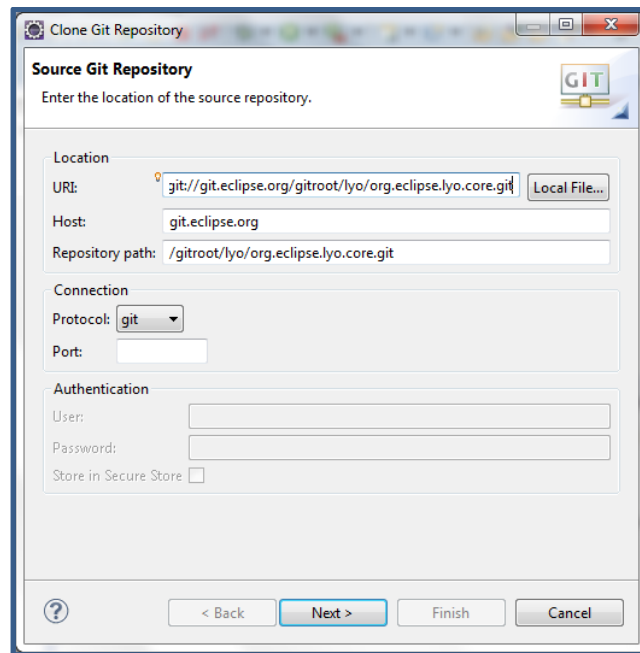


EGit and Maven are already installed within the Eclipse for Java EE package.

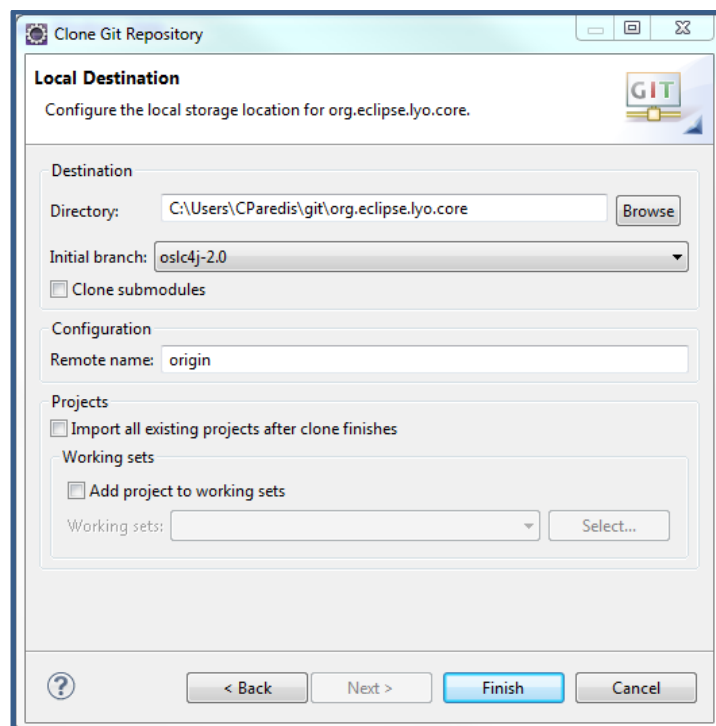
4. Installing the OSLC4J toolkit

1. In Eclipse, open the Git Repositories view. (Window → Show View → Other, search for Git repo by typing git in the search field, select the Git Repositories view among the search results, and click OK)
2. Click Clone a Git Repository.
3. In the Clone Git Repository window, in the URI field paste the following:
`git://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.core.git`

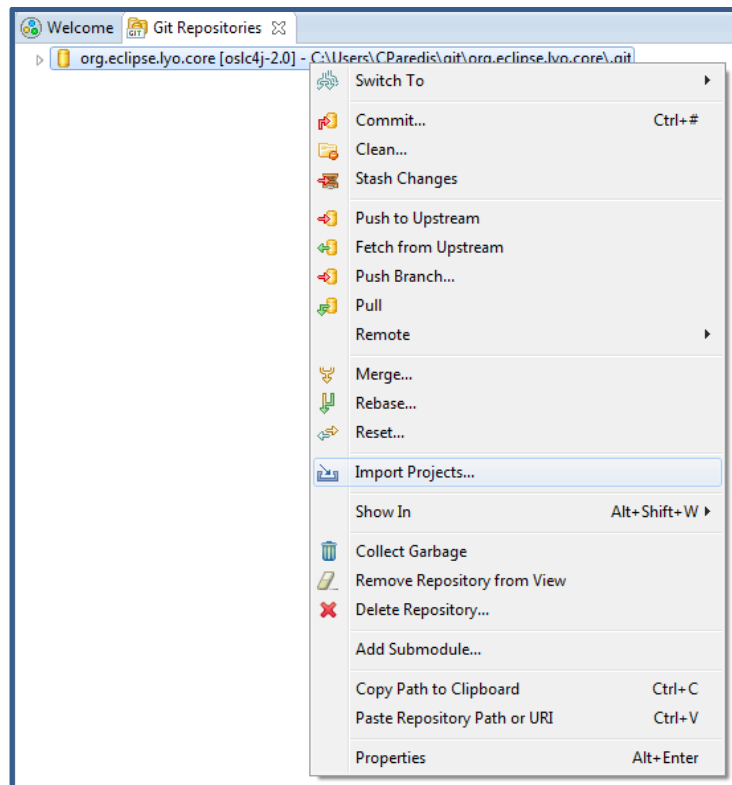
(**Warning:** if you are using a **proxy** and if you get a Transport Error, check the troubleshooting section below)



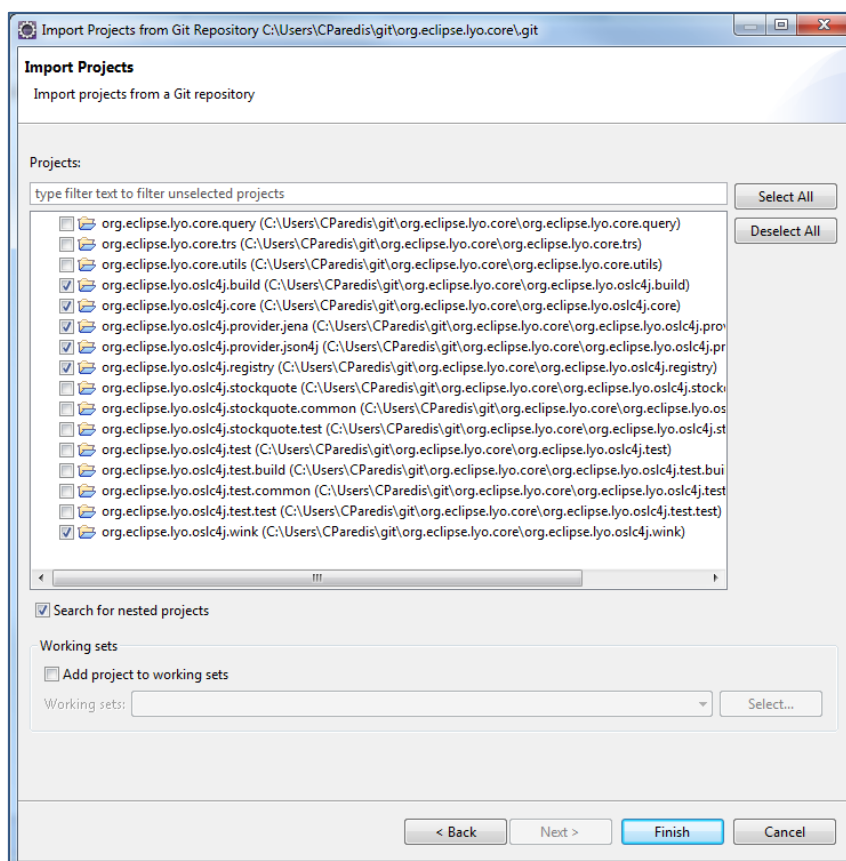
4. The Host and Repository fields will autofill. Leave the Username and Password fields empty.
5. Click Next.
6. On the Branch Selection page, only select oslc4j-2.0 and click Next.
7. For the Destination, select a folder of your choice or accept the default of your Eclipse workspace as shown below. However, do not choose your Eclipse workspace as Git working directory!



8. Click Finish. org.eclipse.lyo.core will appear in the Git Repositories view.
9. In the Git Repositories view, right-click org.eclipse.lyo.core and click Import Projects.



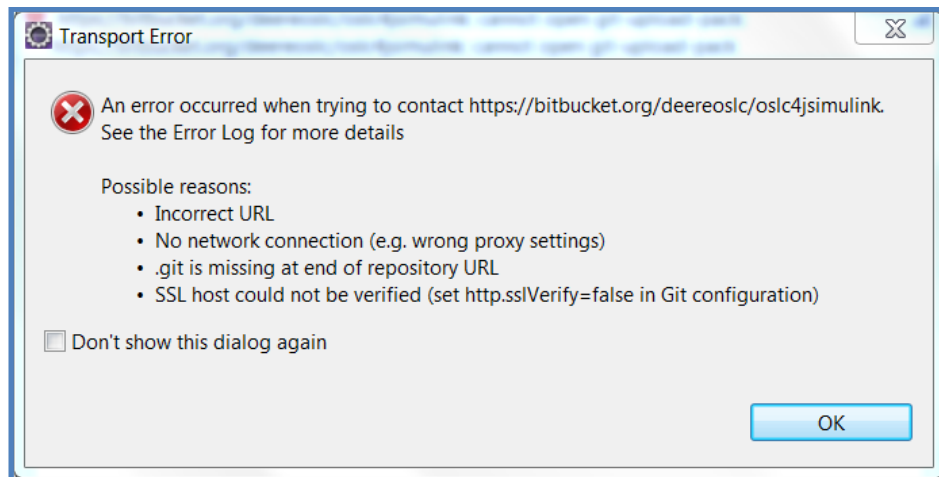
10. In the Import Projects from Git Repository wizard, select Import existing projects and click Next.
11. Only select all projects whose name start with “org.eclipse.lyo.oslc4j” except projects whose name end contain “stockquote” and “test”, as in the figure below, and click Finish.



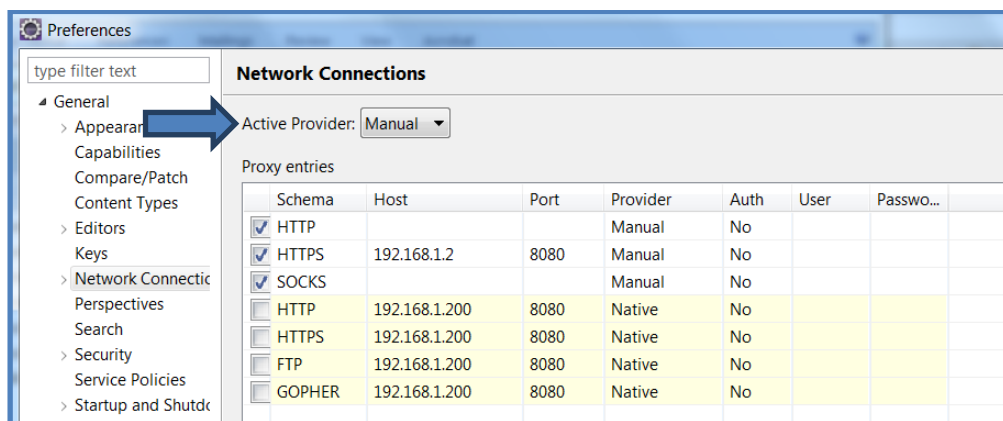
Troubleshooting

Proxy settings in Eclipse (this section is only necessary if you faced a Transport error at step 3 while cloning the git repository)

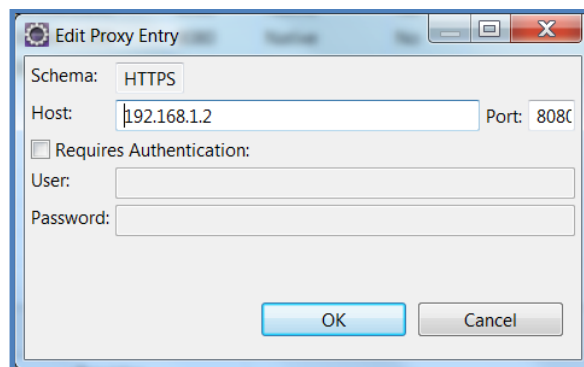
You may be using a proxy which Eclipse doesn't know about. If that is the case, you will receive an error message as displayed below.



Go to Window->Preferences->General->Network Connection. Set *Active Provider* to manual, as shown below.



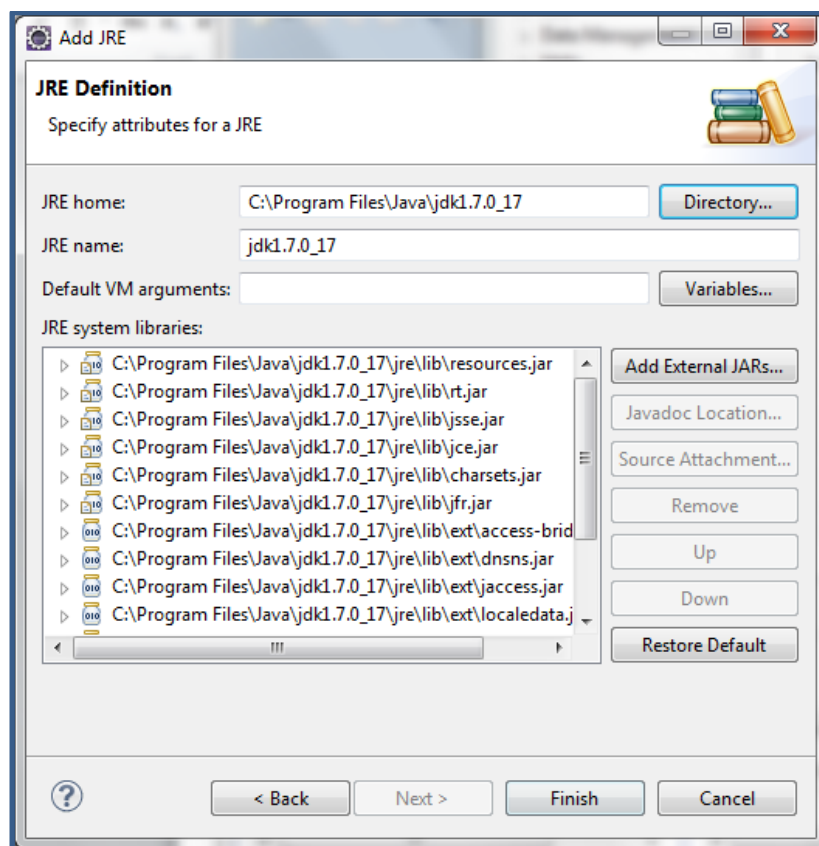
Select the *HTTPS* schema entry, and click edit, and enter your proxy information as for example shown below.



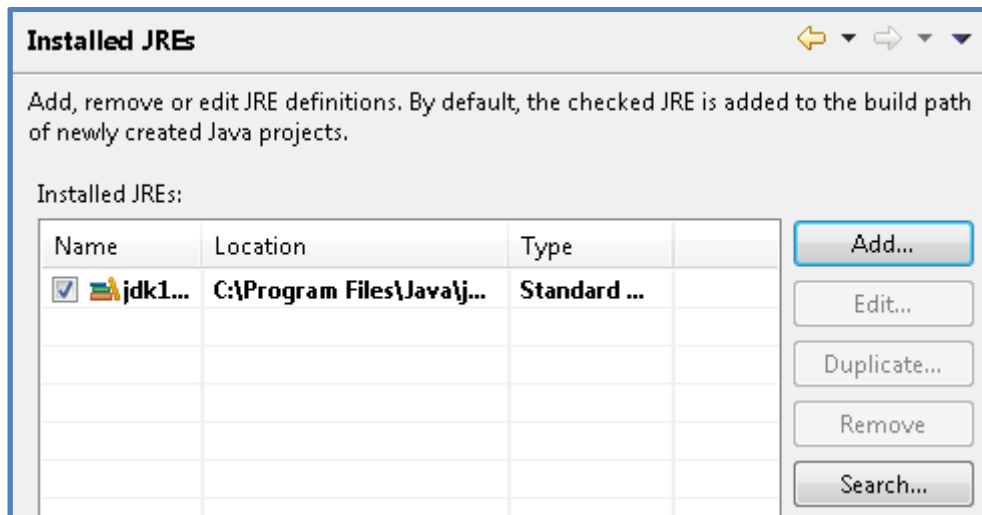
Redo Task #3 in this installation step with following repository URI if you are using the HTTPS scheme: <https://git.eclipse.org/gitroot/lyo/org.eclipse.lyo.core.git>, and continue with the remaining tasks

5. Setting the JRE of Eclipse

1. In Eclipse, open the Preference view (Window->Preferences).
2. Under Java, select the Installed JREs tab.
3. Click on Add, select Standard VM and click Next, and select the Java JDK 7 installation directory (not the JRE directory) as your JRE home as displayed in the images below.

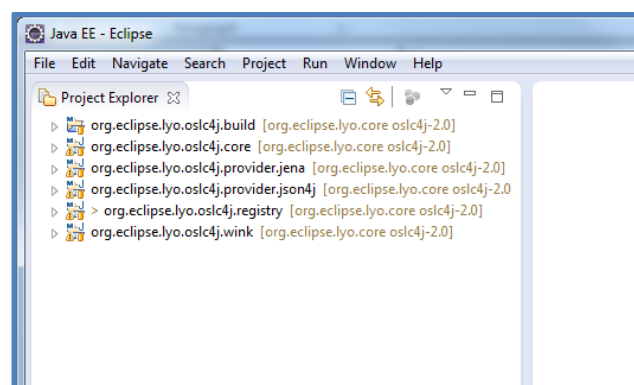


4. Mark jdk7 as your installed JRE, as in the figure below, and click OK.

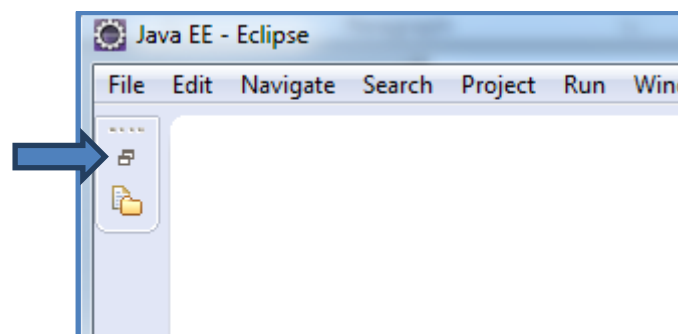


6. Building the OSLC4J projects

1. In Eclipse, open the Project Explorer view. (Window → Show View → Project Explorer). The view should be displayed in the top left corner in your Eclipse IDE, as shown below



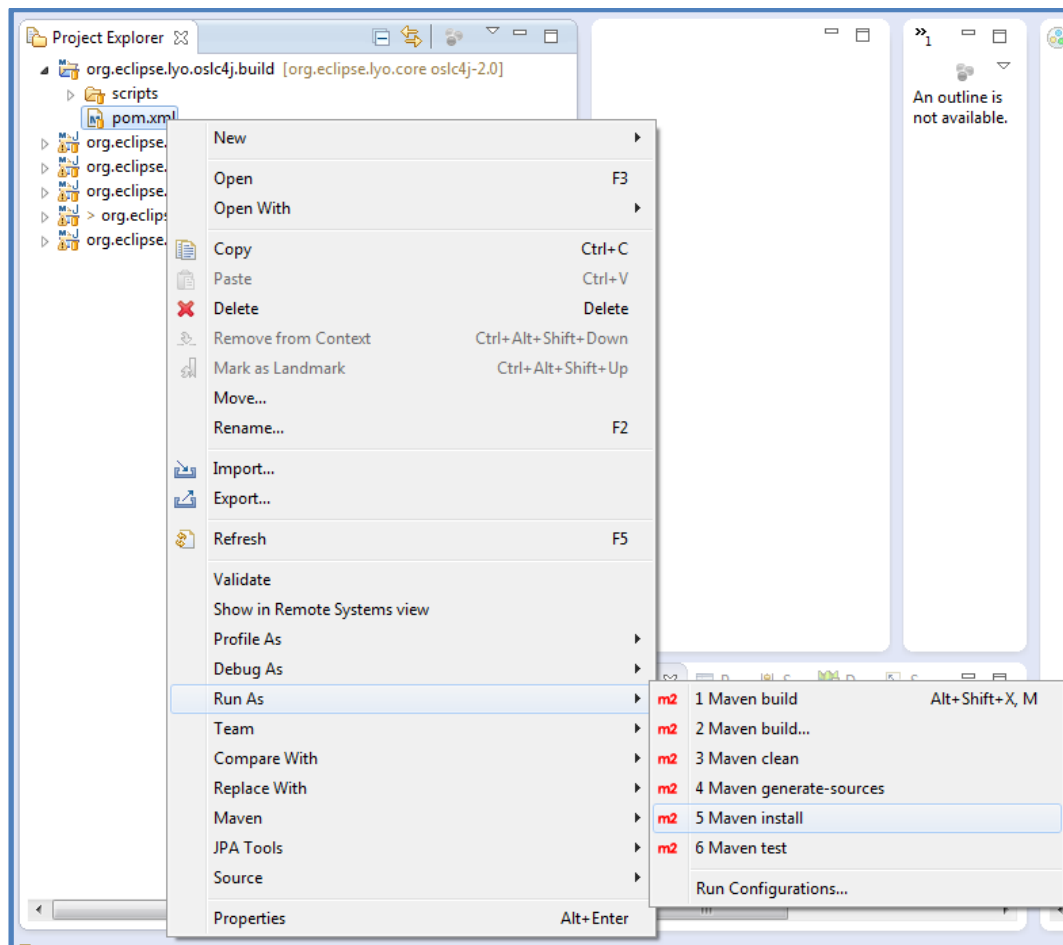
If you do not see it, the view is in “collapsed mode”, as shown below.



Click on the overlapped folder icon as highlighted in the above figure, and you will see the project explorer view

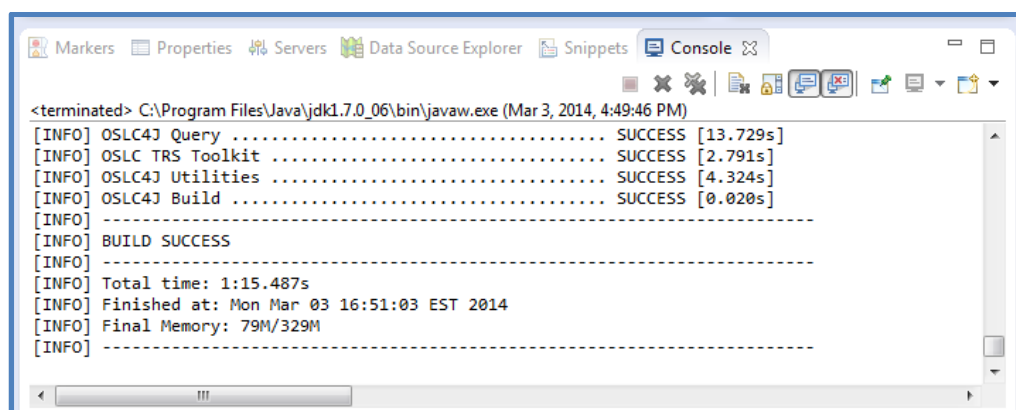
2. Expand the org.eclipse.lyo.oslc4j.build project by clicking on the arrow situated at the left of the project name

3. Delete the *ResolveURITest.java* class contained in the *org.eclipse.lyo.oslc4j.core.test* package of the *org.eclipse.lyo.oslc4j.core* project
4. Right click pom.xml -> Run As -> Maven clean, as shown below

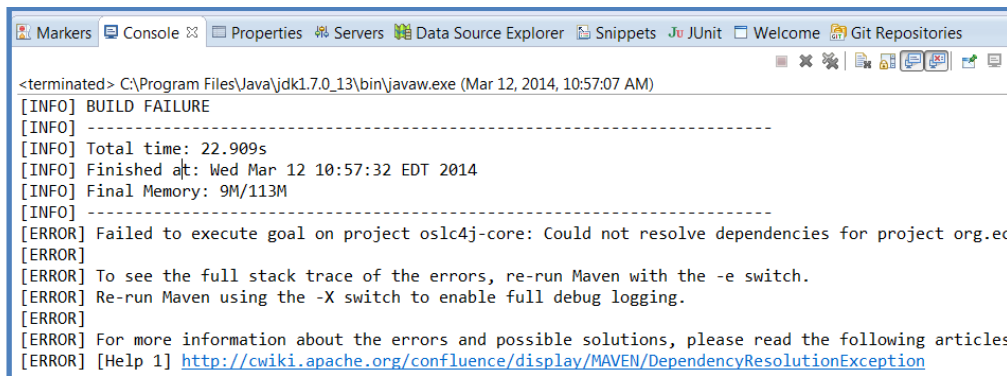


5. Right click pom.xml -> Run As -> Maven install. This procedure may take 1-2 minutes. Maven will download many dependent libraries and then compile the OSLC4J projects.

The console window in the middle of the Eclipse IDE should display a success message when Maven install is finished, as shown below.



You may be using a proxy which Maven doesn't know about. If that is the case, you will receive an error message in the console window as displayed below.



```
<terminated> C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Mar 12, 2014, 10:57:07 AM)
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 22.909s
[INFO] Finished at: Wed Mar 12 10:57:32 EDT 2014
[INFO] Final Memory: 9M/113M
[INFO] -----
[ERROR] Failed to execute goal on project oslc4j-core: Could not resolve dependencies for project org.ec
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/DependencyResolutionException
```

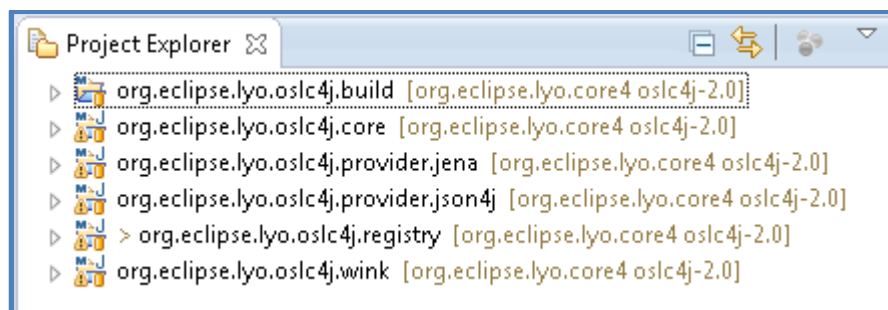
(Warning: if you are using a **proxy** and if you get a Transport Error, check the troubleshooting section below)

If you get a Maven build failure due to a test failure, you will see in the console window an error message referring to a test failure as shown below.

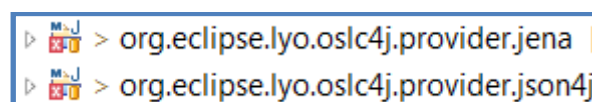
```
[INFO] -----
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 13.373s
[INFO] Finished at: Tue Mar 18 06:16:11 EDT 2014
[INFO] Final Memory: 19M/180M
[INFO] -----
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.10:test
(default-test) on project oslc4j-core: There are test failures.
```

(Warning: if you get a Maven build failure due to a **test failure**, check the troubleshooting section below)

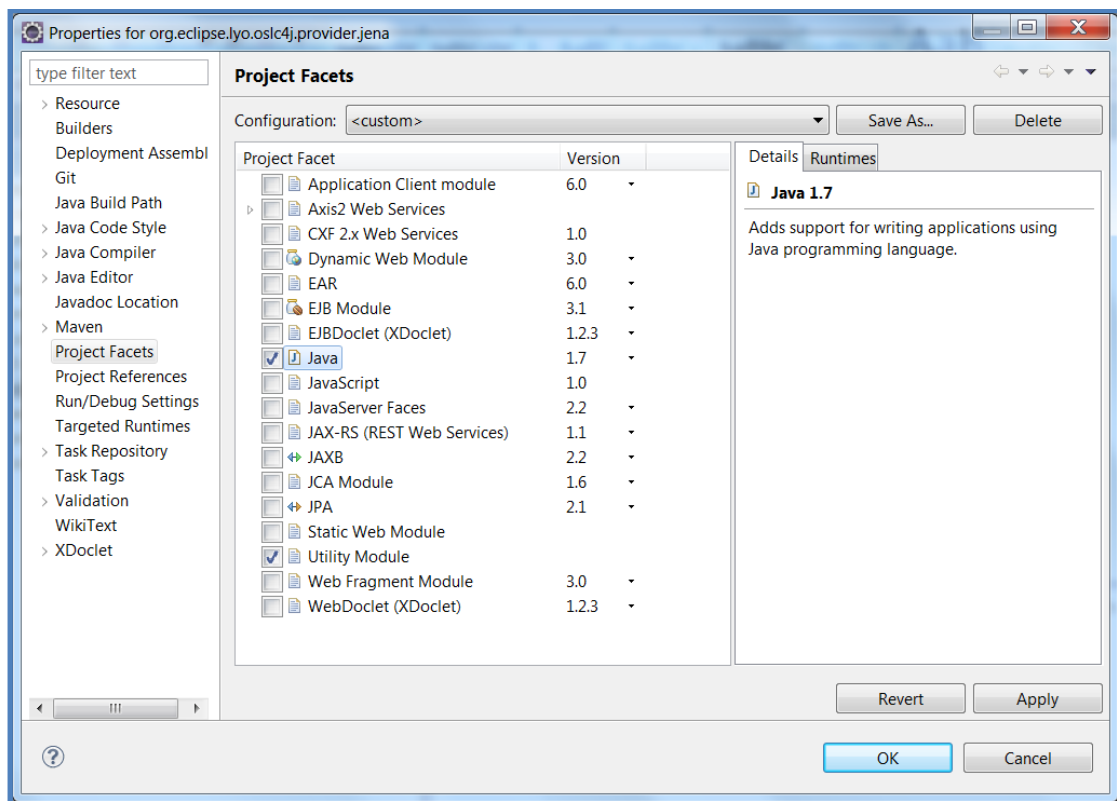
If there is no error mark next to any project, you can skip the next tasks. The installation of Eclipse Lyo was successful. The project explorer view should then look like this:



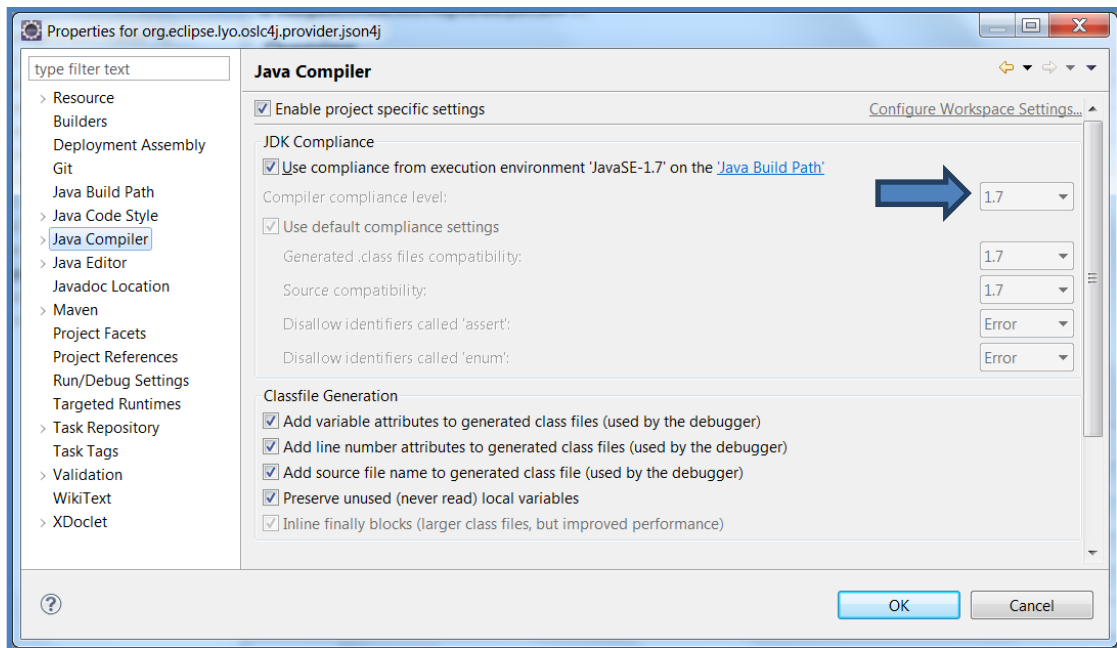
6. Make sure that the Eclipse projects displayed in the project explorer view do not contain any error icons displayed next to the project names as for example displayed below.



If a project has an error icon, then select the project and open its properties view (Project->right click->Properties). Under the Projects Facet tab, make sure that 1.7 is selected as Java version as shown below.



If a project still shows an error, then change its **JDK compliance** to 1.7. Select the project, right-click -> Properties. Select Java Compiler and select **1.7** in the drop down menu next to the JDK compliance setting as highlighted below.



Troubleshooting

Maven build failure due to test failures (this section is only necessary if you faced a compile error at step 4)

If you get a Maven build failure due to a test failure, you will see in the console window an error message referring to a test failure as shown below.

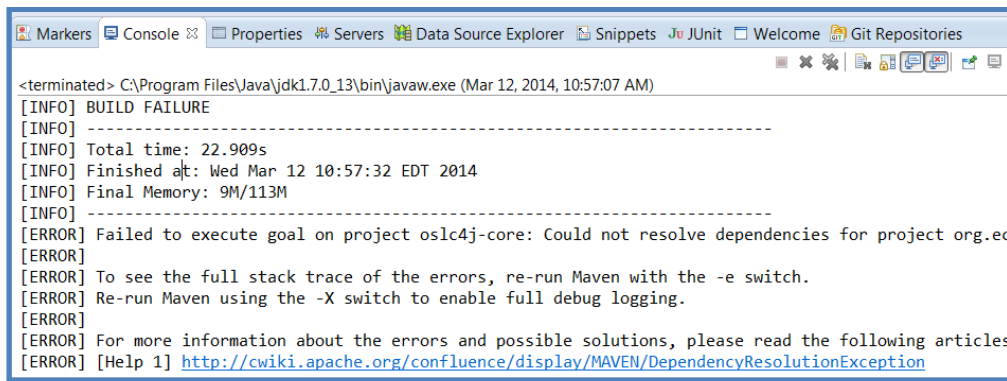
```
[INFO] -----  
[INFO] BUILD FAILURE  
[INFO] -----  
[INFO] Total time: 13.373s  
[INFO] Finished at: Tue Mar 18 06:16:11 EDT 2014  
[INFO] Final Memory: 19M/180M  
[INFO] -----  
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:2.10:test  
(default-test) on project oslc4j-core: There are test failures.
```

The test failure is due to a bug in Eclipse Lyo

(https://bugs.eclipse.org/bugs/show_bug.cgi?id=429946). The quickest solution is to delete the *ResolveURITest.java* class contained in the *org.eclipse.lyo.oslc4j.core.test* package of the *org.eclipse.lyo.oslc4j.core* project. The cleanest solution is to create two separate Java classes for the two test cases that were previously contained in the *ResolveURITest.java* class.

Proxy settings in Maven (this section is only necessary if you faced a compile error at step 4)

You may be using a proxy which Maven doesn't know about. If that is the case, you will receive an error message in the console window as displayed below.



```
<terminated> C:\Program Files\Java\jdk1.7.0_13\bin\javaw.exe (Mar 12, 2014, 10:57:07 AM)
[INFO] BUILD FAILURE
[INFO] -----
[INFO] Total time: 22.909s
[INFO] Finished at: Wed Mar 12 10:57:32 EDT 2014
[INFO] Final Memory: 9M/113M
[INFO] -----
[ERROR] Failed to execute goal on project oslc4j-core: Could not resolve dependencies for project org.ec
[ERROR]
[ERROR] To see the full stack trace of the errors, re-run Maven with the -e switch.
[ERROR] Re-run Maven using the -X switch to enable full debug logging.
[ERROR]
[ERROR] For more information about the errors and possible solutions, please read the following articles
[ERROR] [Help 1] http://cwiki.apache.org/confluence/display/MAVEN/DependencyResolutionException
```

The detailed error message will include following text:

[ERROR] Failed to execute goal on project ...: Could not resolve dependencies for project ...: Failed to read artifact descriptor for ...: Could not transfer artifact .. from/to central (<http://repo.maven.apache.org/maven2>): connection timed out to ... -> [Help 1]

Check first with your system administrator to see if there is a Maven **settings.xml** file for your specific organization. If not, create a text file named settings.xml and copy following contents into the file:

```
<settings>
<proxies>
<proxy>
<active>true</active>
<protocol>http</protocol>
<host>proxy.somewhere.com</host>
<port>8080</port>
<username>proxyuser</username>
<password>somepassword</password>
</proxy>
</proxies>
</settings>
```

Enter your proxy information in the placeholders and save the file. Place the *settings.xml* file in your Maven directory which is typically located at <user home>/.m2/. Your *settings.xml* file will for example be placed at "C:\Users\Axel\.m2\settings.xml".

In Eclipse, Window->Preferences->Maven->User Settings, make sure that the path to the settings.xml file is correctly defined. Click on *Update Settings*.