Custom Java Development

While typically complete applications can be created using VDAB's core nodes or extended node packages, custom handling can be written and incorporated in your flows when the available nodes do not provide the features or efficiency needed.

There are two approaches to adding these custom features including writing a complete custom node or writing a single Java method.

This document details how to set up an environment for developing your own custom Java Functions and VDAB Nodes.

An additional section details how to set up code that is currently found in a VDAB github project.

For additional details covering creating Functions see the *Creating Custom Functions* guide.

For additional details covering creating Nodes see the *Creating Custom Nodes* guide

Before This	• You should have reviewed the VDAB introductory tutorial and documentation to understand how to create basic flows from nodes. • You should have at least limited Java programming experience. • You should be familiar with either the Eclipse or other IDE.

Contents

Related Documentation

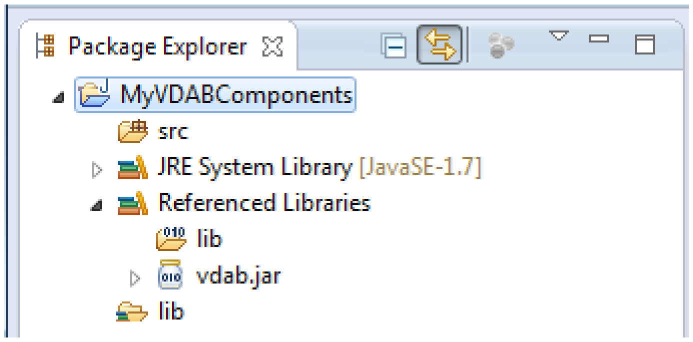
The following document and tutorials either are a) available or b) being developed to further support this subject. Those available are highlighted in green while those under development are currently grey.

Related Guides	Details
VDAB Standard Directory reference	This reference document defines the standard directory structure used by VDAB.
Creating Custom Nodes	This guide documents the approach for writing complete custom nodes in Java.
Creating Custom Functions	This guide documents the approach for writing individual Java functions that can be invoked in the JavaFunction node.
Creating Custom Conditionals	This guide documents the approach for writing individual Java conditionals that can be invoked in the JavaConditional node.

Related Tutorial	Details

Development Setup Overview

Custom development requires that a Java project be set up which includes the *vdab.jar* and the source for the components that are being created or modified.

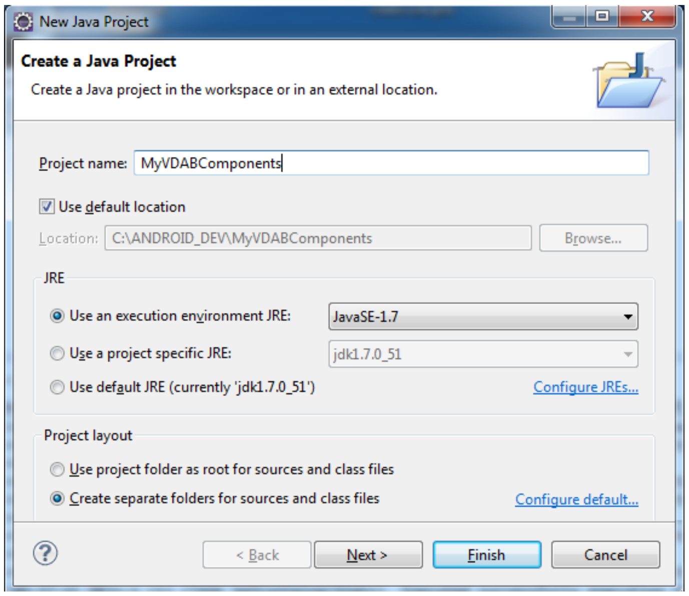


While this can be done in a variety of different ways with different IDE’s, the following is required.

- A Java Project for Java 1.7 or later.
- A reference to the *vdab.jar* library.
- Program execution using the *vdab.jetty.AFJetty* class main method.

Detailed Eclipse Setup

1. Create a standard java project by select *New Java Project* from the *File* tab.



New Java Project

Create a Java Project

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

Project name: MyVDABComponents

[✓] Use default location

Location: C:\ANDROID_DEV\MyVDABComponents

JRE

● Use an execution environment JRE: JavaSE-1.7

○ Use a project specific JRE: jdk1.7.0_51

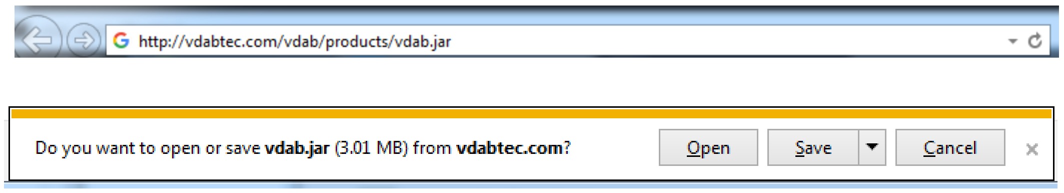
○ Use default JRE (currently 'jdk1.7.0_51')

Project layout

○ Use project folder as root for sources and class files

● Create separate folders for sources and class files

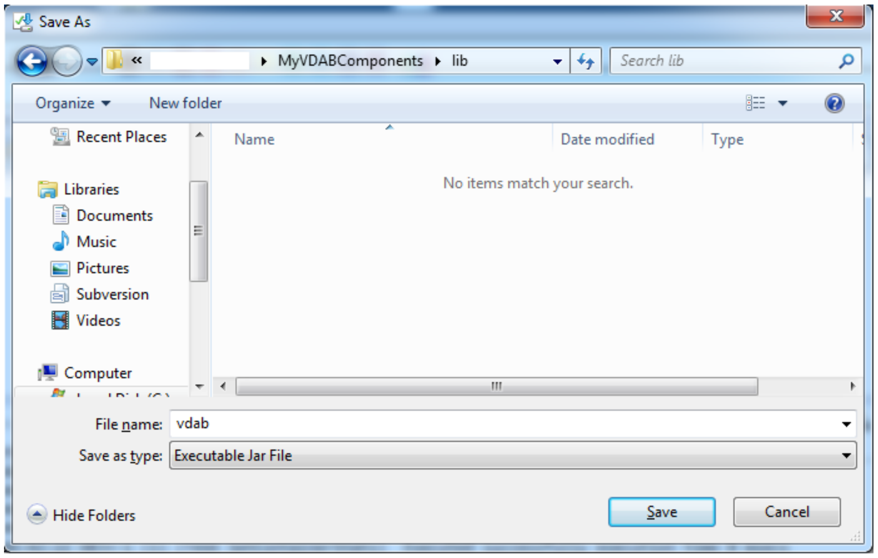
2. Download the most recent VDAB.jar from http://vdabtec.com/vdab/products/vdab.jar



http://vdabtec.com/vdab/products/vdab.jar

Do you want to open or save **vdab.jar** (3.01 MB) from **vdabtec.com**?

3. Select Save As and save the *vdab.jar* in a location within your project. (In this example a *lib* subdirectory was created to hold the jar).

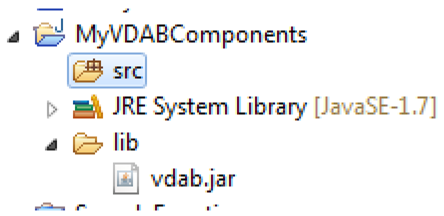


No items match your search.

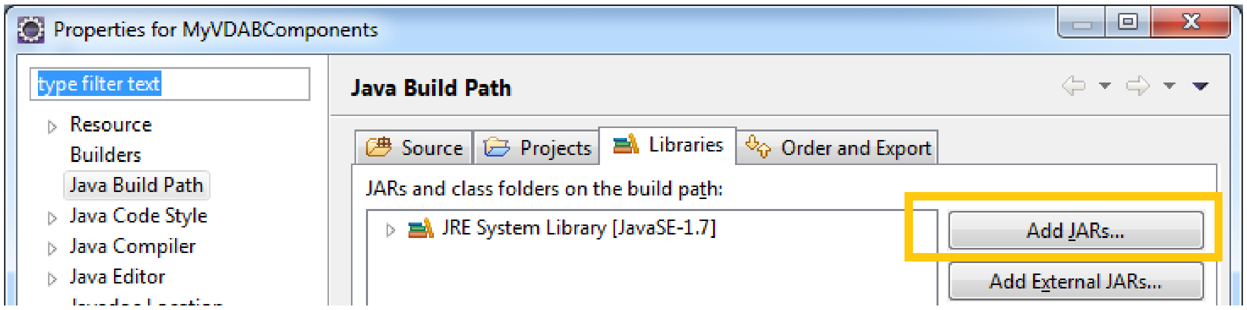
File name: vdab

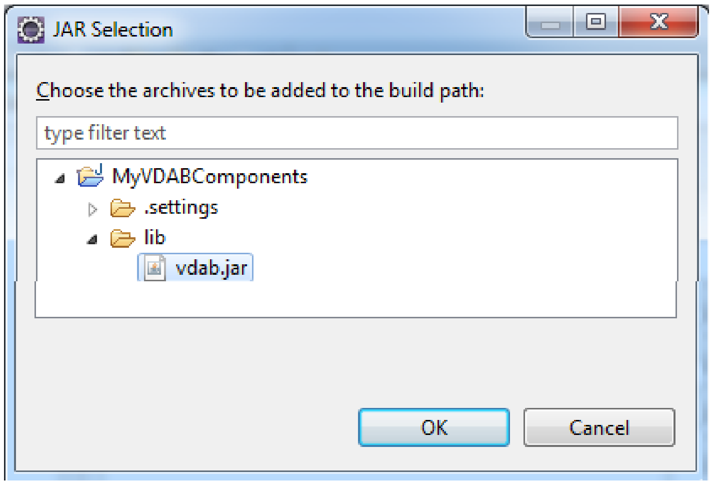
Save as type: Executable Jar File

4. Select the project you just created in Eclipse and Refresh (F5) the project. The *vdab.jar* and *src* directory should be visible.

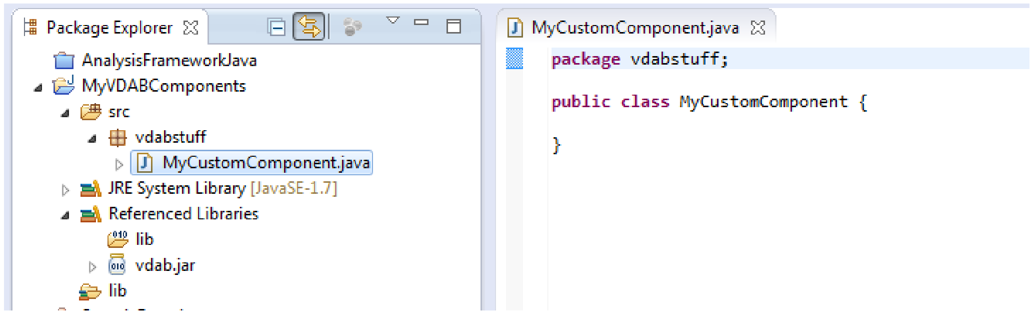


5. Add the *vdab.jar* to your build path by selecting the *Propertie*s option from the *Project* tab.



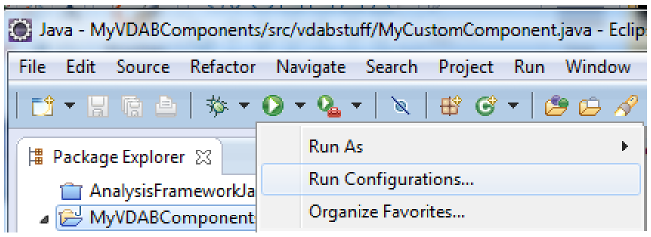


6. Create your custom java code under the *src* folder. (Refer to the guides on *Creating Custom Nodes* and *Creating Custom Functions* regarding the requirements for the Java components.)

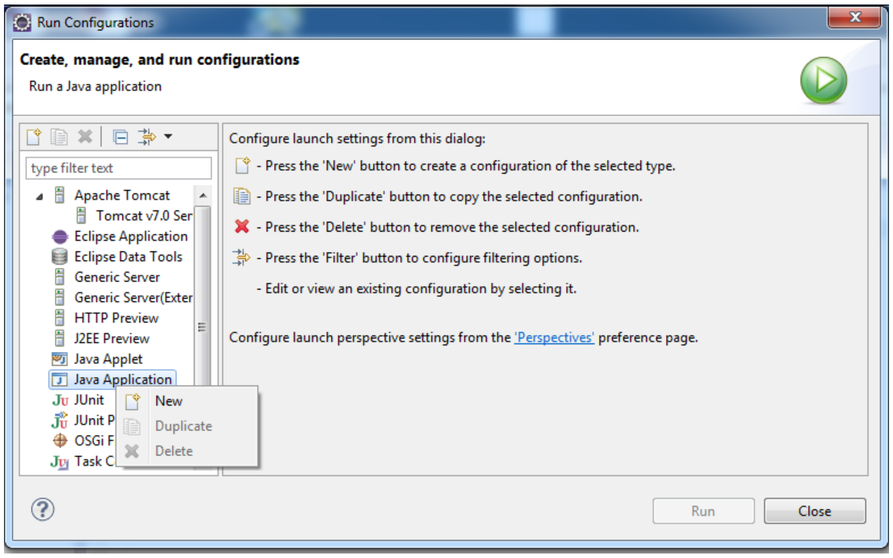


```
package vdabstuff;

public class MyCustomComponent {

}
```

7. Run and test the code by configuring option to call the *AFJetty* main program.


Java - MyVDABComponents/src/vdabstuff/MyCustomComponent.java - Eclip
File Edit Source Refactor Navigate Search Project Run Window
Package Explorer
AnalysisFrameworkJa
MyVDABComponent
Run As
Run Configurations...
Organize Favorites...

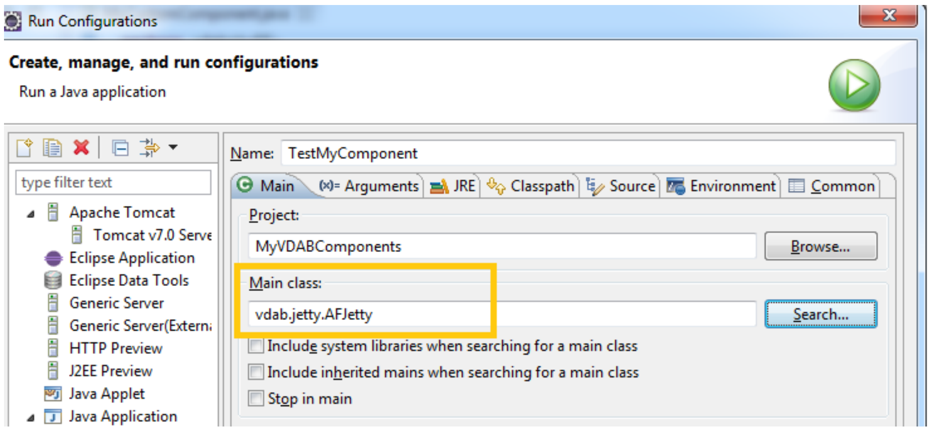
8. Select Run Configuration for a new Java Application



Configure launch settings from this dialog:
- Press the 'New' button to create a configuration of the selected type.
- Press the 'Duplicate' button to copy the selected configuration.
- Press the 'Delete' button to remove the selected configuration.
- Press the 'Filter' button to configure filtering options.
- Edit or view an existing configuration by selecting it.

Configure launch perspective settings from the 'Perspectives' preference page.

9. Select the main class as *vdab.jetty.AFJetty* and save the configuration.



Name: TestMyComponent

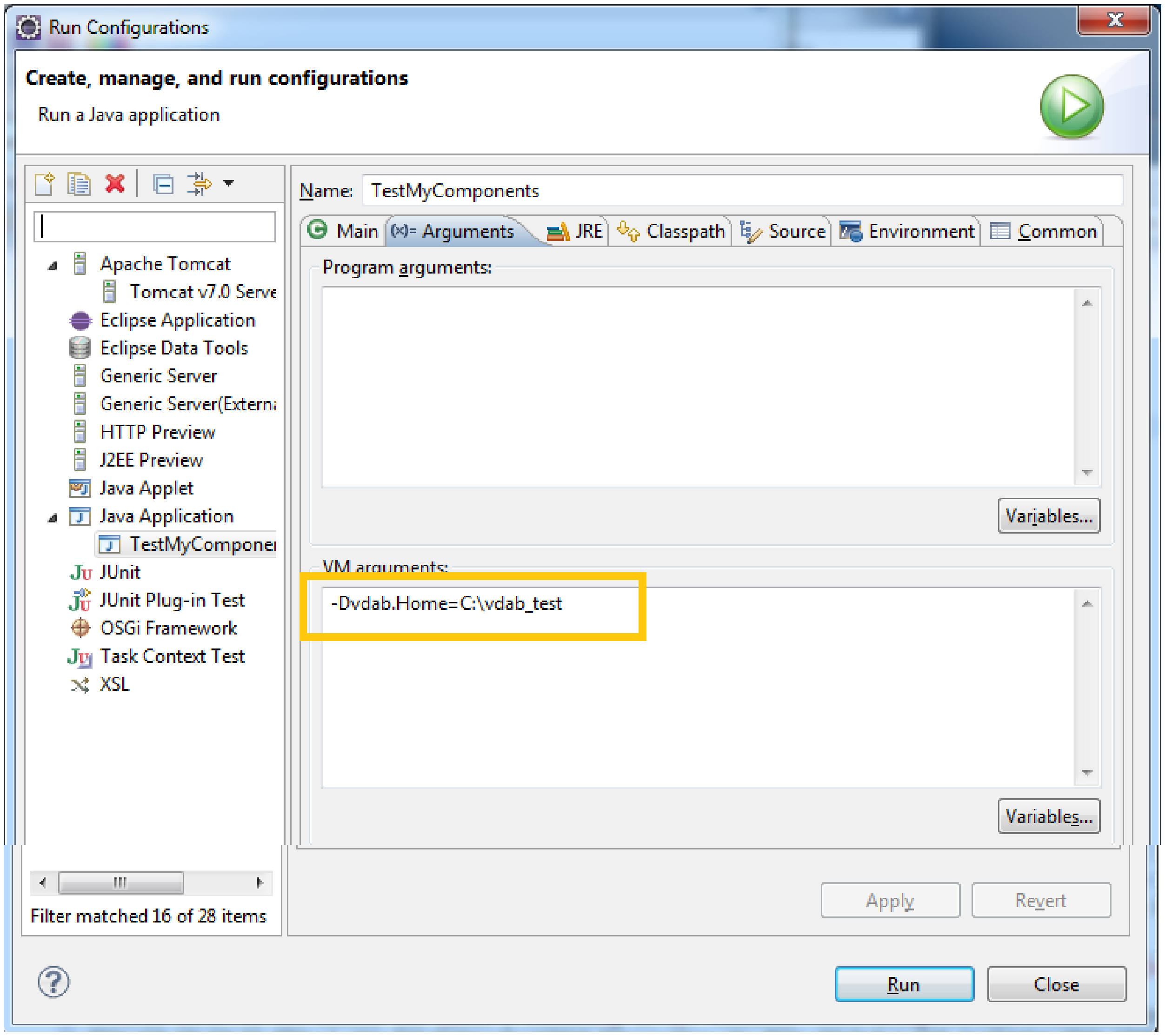
Project: MyVDABComponents

Main class: vdab.jetty.AFJetty

☐ Include system libraries when searching for a main class
☐ Include inherited mains when searching for a main class
☐ Stop in main

10. Run this named configuration and the VDAB server should start. Your own source will be accessible and when running in the debug mode, can be debugged.

11. If you wish to run VDAB from a different home directory, simply define the home directory using the *vdab.Home* VM parameter within the run configuration.



Name: TestMyComponents

Program arguments:

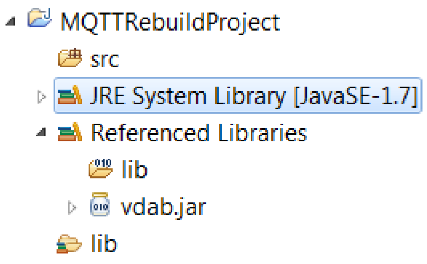
VM arguments:

```
-Dvdab.Home=C:\vdab_test
```

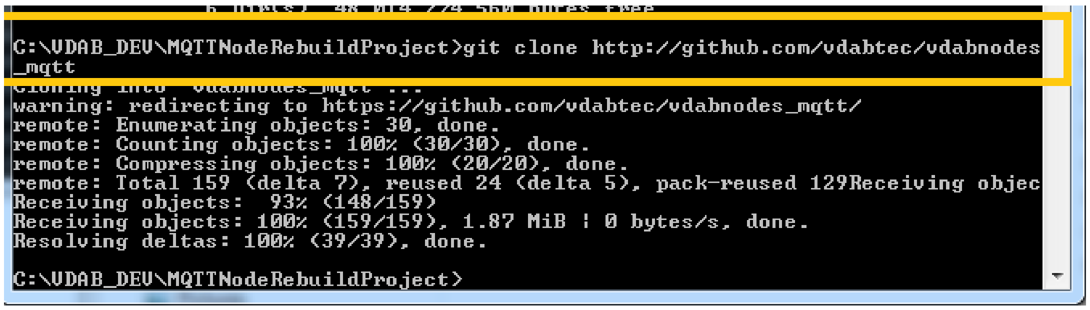
VDAB Github Projects

The previous directions assumed you were creating an entirely new custom component. If you are modifying an existing VDAB Github project, it is recommended that you clone the existing Github into the existing Eclipse project. The setup will follow the previous directions but requires that you reference the proper source folder for the Github project source and that you reference an jars included in the Github project:

1. Setup the Eclipse project following the procedures in the previous section.

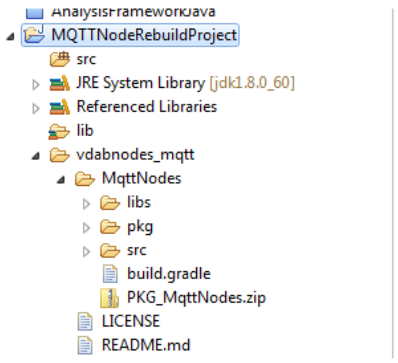


2. Ensure that you can run VDAB from this project. (See previous section.)
3. Clone the Git project underneath the project. (In this example we are cloning the *vdabnodes_mqtt* project.)

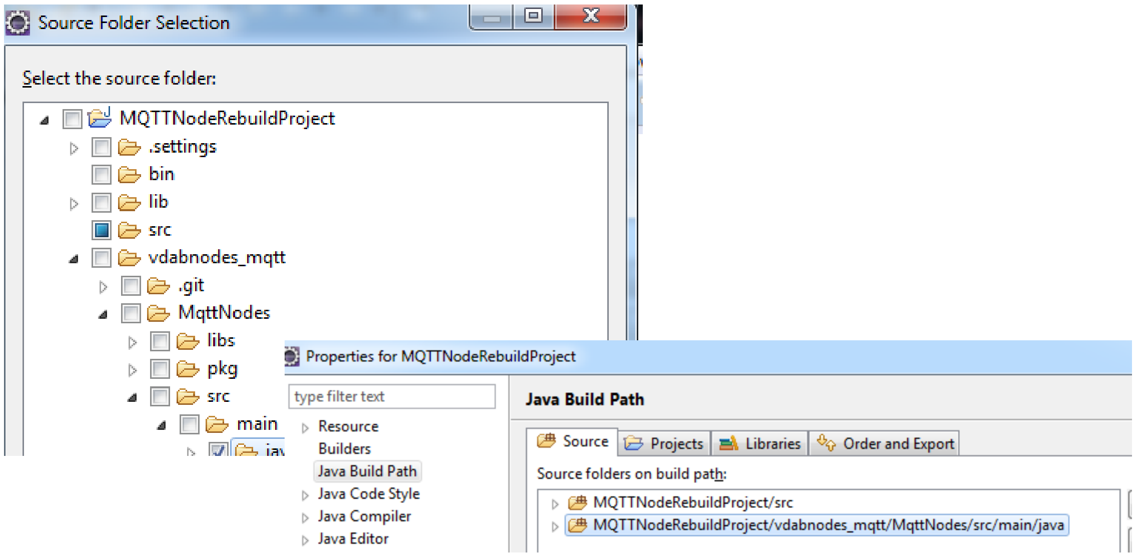


```
C:\VDAB_DEV\MQTTNodeRebuildProject>git clone http://github.com/vdabtec/vdabnodes
_mqtt
Cloning into 'vdabnodes_mqtt'...
warning: redirecting to https://github.com/vdabtec/vdabnodes_mqtt/
remote: Enumerating objects: 30, done.
remote: Counting objects: 100% (30/30), done.
remote: Compressing objects: 100% (20/20), done.
remote: Total 159 (delta 7), reused 24 (delta 5), pack-reused 129Receiving objec
Receiving objects:  93% (148/159)
Receiving objects: 100% (159/159), 1.87 MiB | 0 bytes/s, done.
Resolving deltas: 100% (39/39), done.

C:\VDAB_DEV\MQTTNodeRebuildProject>
```

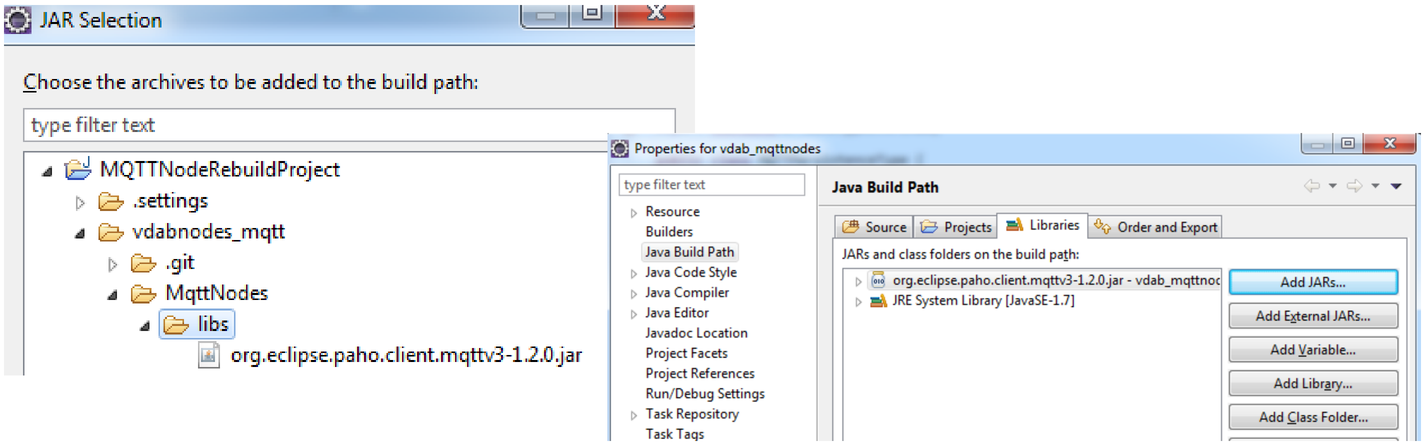
4. Refresh the project so that the Github node project is visible under your Eclipse project.



5. In Eclipse tell the project to use the source folder where the Github java code is located:

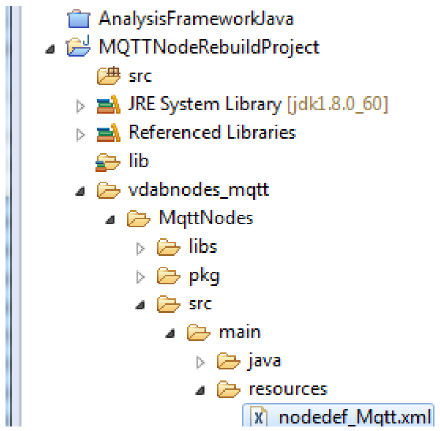


6. Reference any jars that are needed for support of the java node code. In this case the mqttv3 jar, which is found in the lib directory of the Github project, needs to be added to the libraries on the build path.

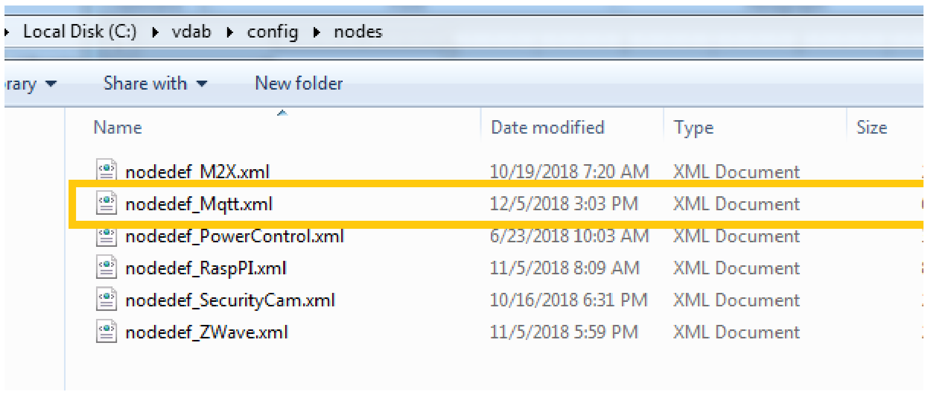


7. The GitHub source can now be edited and rebuilt.

8. When running the edited node from Eclipse, the nodedef file must be copied to the `../config/nodes` directory to ensure that the nodes are defined while testing.



```
AnalysisFrameworkJava
MQTTNodeRebuildProject
  src
  JRE System Library [jdk1.8.0_60]
  Referenced Libraries
  lib
  vdabnodes_mqtt
    MqttNodes
      libs
      pkg
      src
        main
          java
          resources
            nodedef_Mqtt.xml
```



Local Disk (C:) ▸ vdab ▸ config ▸ nodes

rary ▾ Share with ▾ New folder

Name	Date modified	Type	Size
nodedef_M2X.xml	10/19/2018 7:20 AM	XML Document	
nodedef_Mqtt.xml	12/5/2018 3:03 PM	XML Document	
nodedef_PowerControl.xml	6/23/2018 10:03 AM	XML Document	
nodedef_RaspPI.xml	11/5/2018 8:09 AM	XML Document	
nodedef_SecurityCam.xml	10/16/2018 6:31 PM	XML Document	
nodedef_ZWave.xml	11/5/2018 5:59 PM	XML Document	

.