# Developing OpenDJ with Eclipse IDE

## Table of Contents



## trunk/master (from OpenDJ 3.0.0 onward)

### Importing OpenDJ within Eclipse using m2eclipse

The quickest and easiest way to use the Eclipse IDE for OpenDJ development is to:

1. Install the maven to eclipse (m2e) plugin
2. Check out the source code from Subversion into a new directory inside your Eclipse workspace:

   ```
   $ mkdir workspace # if necessary
   $ cd workspace
   $ svn co https://svn.forgerock.org/opendj/trunk/opendj opendj-trunk
   $ cd opendj-trunk
   $ mvn clean install
   ```

   The final step builds some tools and generates some files that are awkward to build inside Eclipse.
3. Start Eclipse, opening the above workspace.
4. Import OpenDJ projetct, in Eclipse click on **File > Import > Maven > Existing Maven Projects**
5. Click **Next** and select your workspace path to retrieve OpenDJ maven modules
6. Click **Finish**.
7. You may have some errors related to the maven to eclipse plugin. Just click on **Resolve All Later** button and then click on **Finish**.
8. OpenDJ maven modules should be ready for your work!
9. You may want to consider importing these Eclipse code formatting rules as well if you want to do any significant development in **opendj-server-legacy** module.
   Or these for the SDK : ForgeRock code formatting rules.
   Note that formatter rules can be selected on a per-project basis: right click on your project, then **Properties** > **Java Code Style** > **Formatter** > Import...

### Importing OpenDJ within Eclipse using mvn eclipse:eclipse

1. Check out the source code from Subversion into a new directory inside your Eclipse workspace:

   ```
   $ mkdir workspace # if necessary
   $ cd workspace
   $ svn co https://svn.forgerock.org/opendj/trunk/opendj opendj-trunk
   $ cd opendj-trunk
   $ mvn clean install
   $ mvn eclipse:clean eclipse:eclipse
   ```

   The step before final builds some tools and generates some files that are awkward to build inside Eclipse.
2. Edit `.classpath` file and replace the following lines:

   ```
   <classpathentry kind="src" path="src/test/java" including="**/*.properties" excluding="**/*.java"/>
   <classpathentry kind="output" path="target/classes"/>
   ```

   with the following lines (line order does not affect output):

   ```
   <classpathentry kind="src" path="src/test/java" excluding="org/opends/server/snmp/"/>
   <classpathentry kind="src" path="target/classes" excluding="org/opends/guitools/controlpanel/|org/opends
   /guitools/uninstaller/|org/opends/quicksetup/"/>
   <classpathentry kind="output" path="target/bin"/>
   ```

   Or run this sed command:

```
sed -i 's@<classpathentry kind="src" path="src/test/java" including="\*\*/\*.properties" excluding="\*\*/\*.
java"/>@<classpathentry kind="src" path="src/test/java" excluding="org/opends/server/snmp/"/>@' .classpath
sed -i 's@<classpathentry kind="output" path="target/classes"/>@<classpathentry kind="src" path="target
/classes" excluding="org/opends/guitools/controlpanel/|org/opends/guitools/uninstaller/|org/opends
/quicksetup/"/><classpathentry kind="output" path="target/bin"/>@' .classpath
```

### Running TestNG unit tests within Eclipse

Unfortunately it is not possible to run the unit tests within Eclipse using the TestNG plugin without some initial preparation:

1. Go to **Window > TestNG > Run/Debug**. Add the following VM Argument in the **JVM args** field:
   `-Dorg.opends.server.BuildRoot=${project_loc}`
2. Now refresh (F5) your workspace in Eclipse and select a unit test and run.

## Previous OpenDJ versions (before 3.0.0)

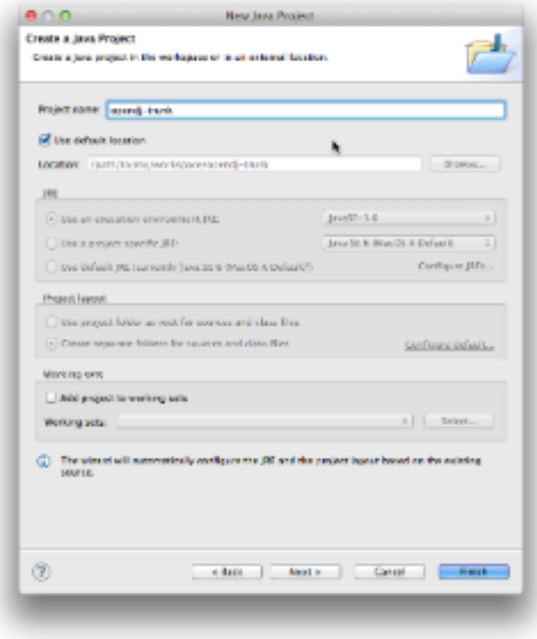
### Importing OpenDJ within Eclipse

The quickest and easiest way to use the Eclipse IDE for OpenDJ development is to:

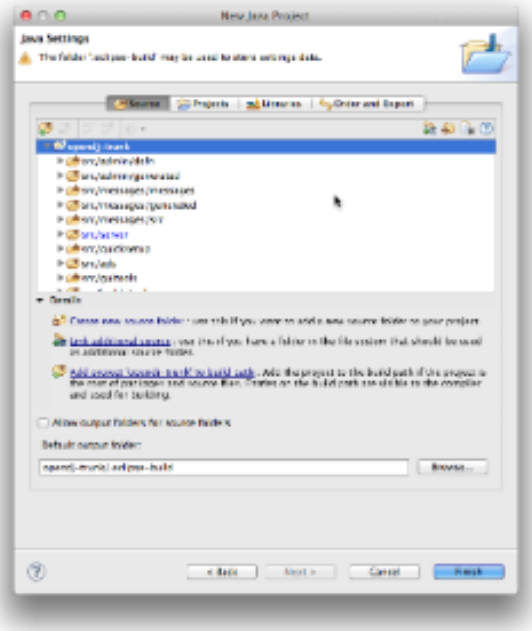
1. Check out the source code from Subversion into a new directory inside your Eclipse workspace:

   ```
   $ mkdir workspace # if necessary
   $ cd workspace
   $ svn co https://svn.forgerock.org/opendj/trunk/opends opendj-trunk
   $ cd opendj-trunk
   $ ./build.sh
   ```

   The final step builds some tools and generates some files that are awkward to build inside Eclipse.
2. Unzip the file EclipseClasspath.zip into the top of the checked out directory. This helps Eclipse better understand the layout of the OpenDJ code.
3. Start Eclipse, opening the above workspace.
4. Create a **File > New > Project > Java Project** (Eclipse Indigo) or **File** > **New** > **Java Project** (Eclipse Juno)
5. Enter the name of the directory you checked out the source into (e.g. `opendj-trunk`):



6. Click **Next** to see the project settings; note you don't need to edit anything else:



7. Click **Finish**. The new project (e.g. `opendj-trunk`) should be ready for your work!
8. You may want to consider importing these Eclipse code formatting rules as well if you want to do any significant development. Or these for the SDK : Forgerock code formatting rules.
   Note that formatter rules can be selected on a per-project basis: right click on your project, then **Properties** > **Java Code Style** > **Formatter** > Import...

## Running TestNG unit tests within Eclipse

Unfortunately it is not possible to run the unit tests within Eclipse using the TestNG plugin without some initial preparation:

1. First of all you need to set the following VM argument for the installed JRE that you are using within Eclipse. Go to **Window > Preferences > Java > Installed JREs**. Select your default JRE and then click on **Edit** and add the following **Default VM Argument**:

   ```
   -Dorg.opends.server.BuildRoot=${project_loc}
   ```

2. The unit tests will look for the resource bundles during initialization so these need to be located in the Eclipse TestNG plugin's class-path. If Eclipse is configured to build in the directory ".eclipse-build" then copy the resources using the following command:

   ```
   $ cp -r build/classes/admin build/classes/messages .eclipse-build    # Before 9 Oct 2013
   $ ant eclipse                                                        #  After 9 Oct 2013 - trunk r9692
   ```

3. Now refresh (F5) your workspace in Eclipse and select a unit test and run.