In this tutorial, we describe steps for setting up a Maven project that uses `libSBOLj` in Eclipse. Another tutorial follows this one which explains how we use SBOL 2 to represent the function of a state-of-the-art design, namely a CRISPR-based repression module Kiani *et al.* [1] using the Java library.

Set up Maven plugins in Eclipse

In this section, we describe steps for installing Maven plugins in Eclipse. The Eclipse version used is Luna Service Release 2 (4.4.2). One thing to mention before we start the tutorial is users do need to take note while attempting this tutorial that if any plugins are already installed, then Eclipse won't allow re-installation of that plug-in and any relating steps described within the tutorial can be skipped. Here are the steps:

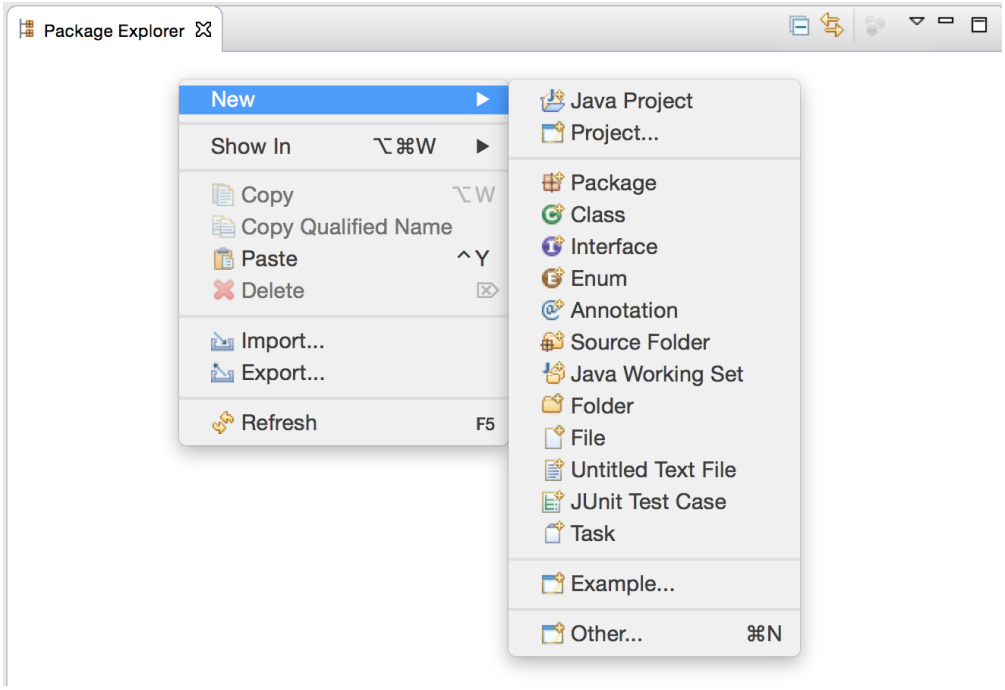
1. Versions of Eclipse later than Luna has Maven installed already, therefore users can skip steps to the install Maven plugin and continue to Creating a new project section.
2. In Eclipse, go to Help and select Install New Software,
3. Add a new software site: Name = slf4j, URL = `http://www.fuin.org/p2-repository/`,
4. Check the site specified in previous step to work with, expand Maven osgi-bundles, and select slf4j.api,
5. Click Next and follow the installation process.

   Note: Users may be prompted to restart Eclipse. While it isn't required to restart Eclipse, it is **recommended** to restart to ensure that the new software was installed.
6. Add a new software site: Name = Maven Plugins, URL = `http://download.eclipse.org/technology/m2e/releases`,
7. Check the site specified in previous step to work with, expand Maven Integration for Eclipse, and select m2e - Maven Integration for Eclipse, and
8. Click Next and follow the installation process.

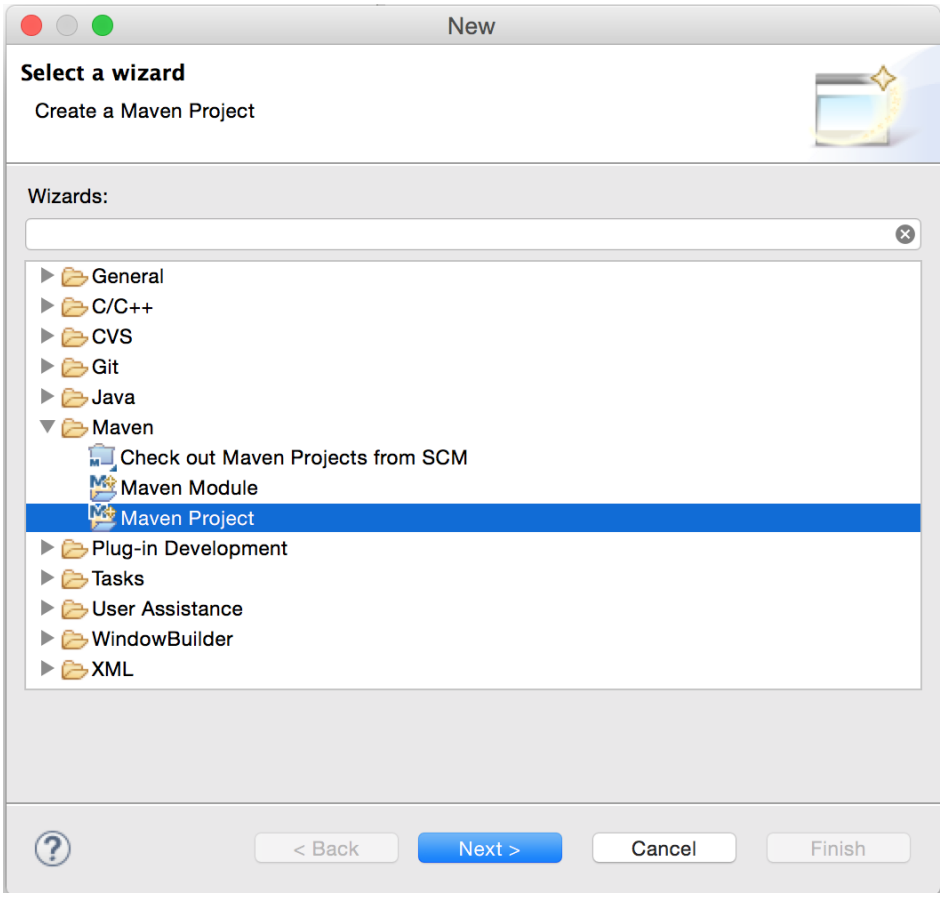
Creating a new project

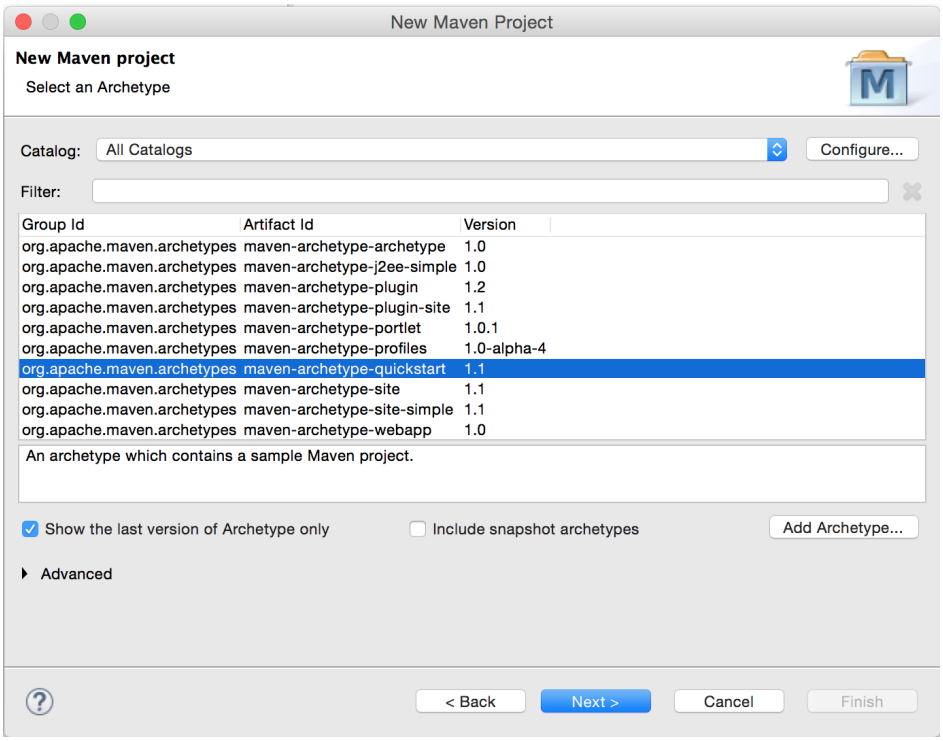
After the Maven plugin is installed, we are now ready to create a new Maven project in Eclipse. The following text describe the necessary steps.

In the package explorer window, right click and select **New** → **Other**.

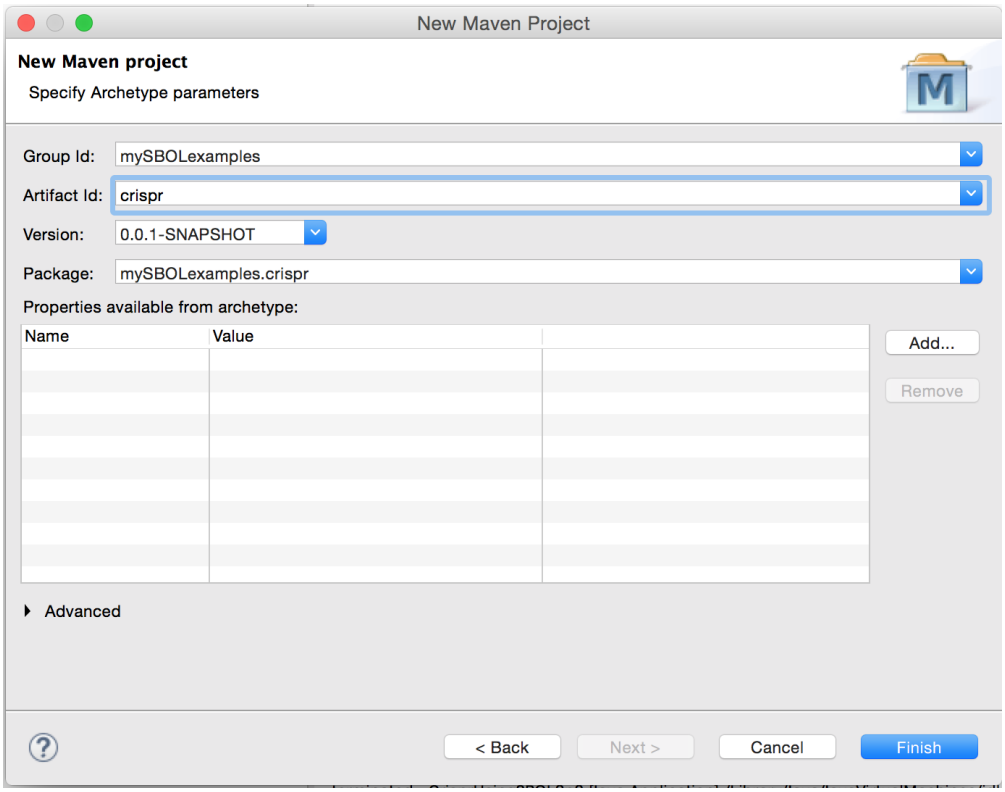


Under the **Maven** folder, choose **Maven Project**, and then follow the default options for the project setup.



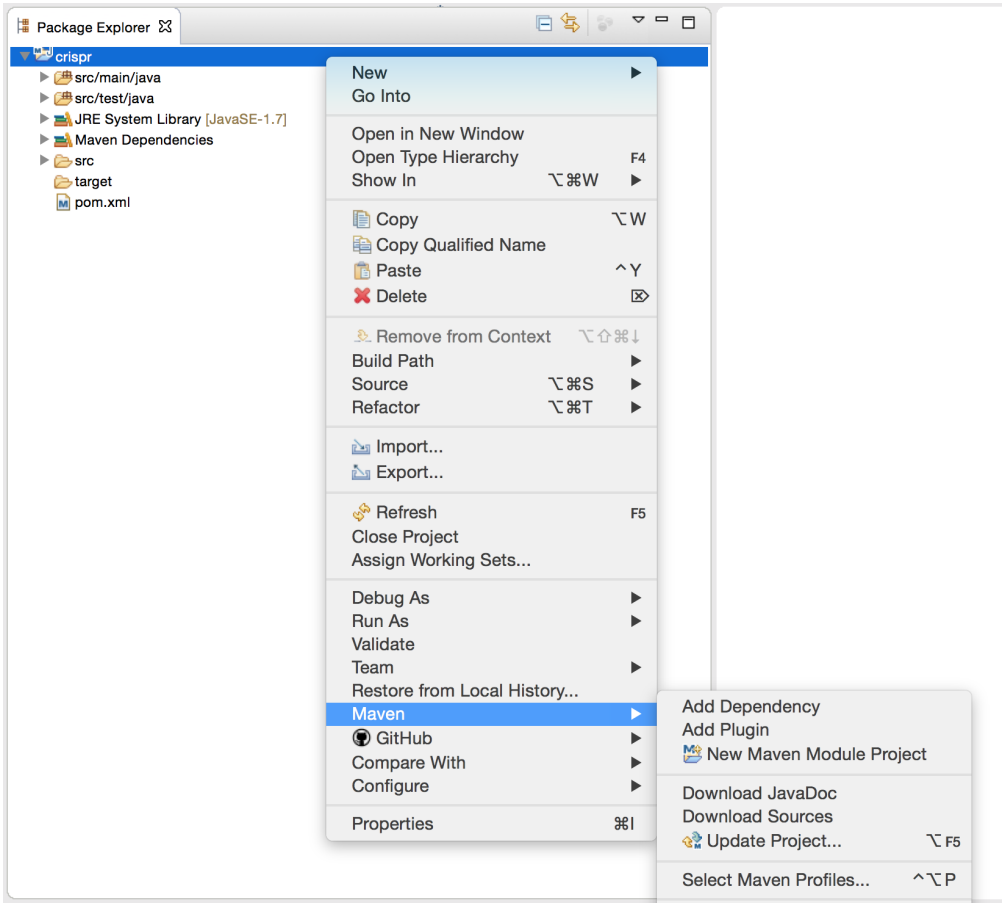


In the window for specifying archetype parameters, you may type your group ID and artifact ID. In this tutorial, they are specified as shown below.



Once the project setup is finished, you should be able to see two Java source folders, a **JRE System Library** and a **Maven Dependencies** library, and a **pom.xml** file.

It is possible that the JRE System Library created by Maven is not compatible with the installed JREs. In the screenshot shown below, it is set to **J2SE-1.5**, but the Maven compiler is compatible with **JavaSE-1.7** instead. To change it, right-click on the "crispr" project and then select **Maven** → **Update Project**, the JRE will reset itself back to J2SE-1.5.



A permanent fix would be to manually add the plugin management information below to the pom.xml file. To access the pom.xml file, the user needs to double-click on the pom.xml file in the project menu. Then once a window pops up, select the rightmost tab on the bottom of the screen titled **pom.xml**.

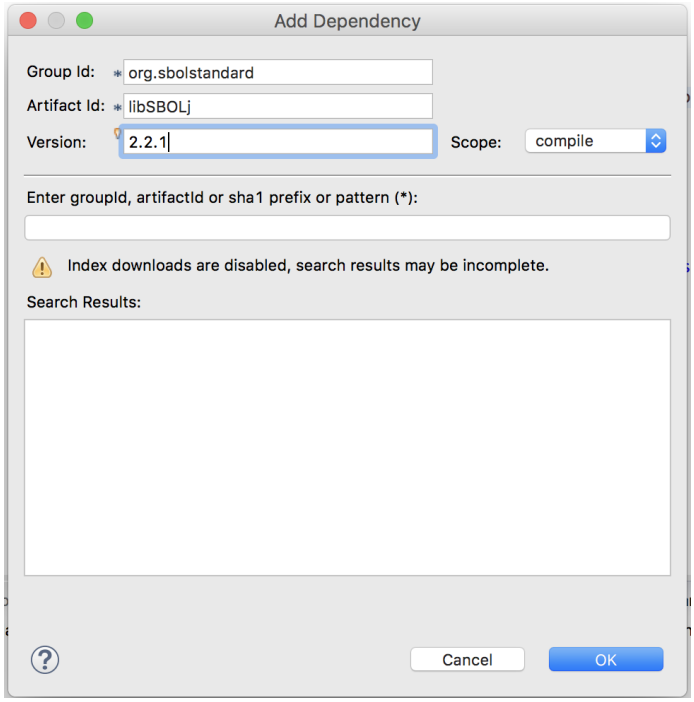
```
[language=xml,basicstyle=\footnotesize\ttfamily]
<build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
```

The pom.xml should look like the one shown below after this modification. Remember to save the file and then do **Maven → Update Project**.

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>mySBOLexamples</groupId>
  <artifactId>crispr</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>crispr</name>
  <url>http://maven.apache.org</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.1</version>
                <configuration>
                    <source>1.7</source>
                    <target>1.7</target>
                </configuration>
            </plugin>
        </plugins>
    </pluginManagement>
</build>
</project>
```

Adding `libSBOLj` as dependency

We are now ready to add libSBOLj as a Maven dependency. This can be easily done by right-clicking on the "crispr" project and then select **Maven** → **Add Dependency**. In the popup window shown below, fill in the information for the `libSBOLj` library. The group ID is **org.sbolstandard**, and the artifact ID is **libSBOLj** and the version is **2.2.1**.



After this dependency is added, Maven automatically brings in the `libSBOLj-2.2.1.jar` and its dependencies from the Maven Central Repository, and places them under the **Maven Dependencies** directory. The user will need to right click on the project directory and select **Maven** → **Update Project**. Within this window, the user needs to ensure that **Force Update of Snapshots/Releases** is checked.

References

[1] S. Kiani, J. Beal, M. Ebrahimkhani, J. Huh, R. Hall, Z. Xie, Y. Li, and R. Weiss, "Crispr transcriptional repression devices and layered circuits in mammalian cells," *Nature Methods*, vol. 11, no. 7, pp. 723–726, 2014.