Using Maven for HJlib projects

Maven is a build automation tool used primarily for Java projects. Projects using Maven or other build systems are easiest to use as they simplify compiling, building, and testing the project. In addition, major IDEs like IntelliJ and Eclipse have excellent support via maven plugins which simplifies the development process. Dependency management is one of the areas where Maven excels, for our purposes this means that we will save a lot of effort in configuring the projects to set up the dependency on the HJlib jars. Hence, for the labs and assignments in COMP322 we will be distributing maven project templates to be used by students to complete their work.

Maven incorporates the concept of convention over configuration by providing sensible default behavior for projects. Without customization, source code is assumed to be in *${basedir}/src/main/java* and resources are assumed to be in *${basedir}/src/main/resources*. Tests are assumed to be in *${basedir}/src/test*, and a project is assumed to produce a JAR file. The pom.xml file is the core of a project's configuration in Maven. It is a single configuration file that contains the majority of information required to build a project in just the way you want. The interested reader can learn more about maven by reading various articles and tutorials.

Maven Installation:

Maven is a Java tool, so you must have Java installed in order to proceed. Next, follow the instructions on the maven site to install maven. Once installed, open a new command prompt and run `mvn --version` to verify that it is correctly installed.

Your first HJlib Maven project:

Once you have maven installed, it is now time to run your first HJlib project using maven.
Download the <project-name>.zip file provided to you for use as the lab or assignment.
Unzip the contents of the zip file, for example `lab1.zip` unzips the contents into a directory named `lab1`.
After unzipping this is how the directory structure should look like:

```
.
README
pom.xml
src
    main
       java
           edu
               rice
                   comp322
                       HelloWorldError.java
                       ReciprocalArraySum.java
    test
        java
            edu
                rice
                    comp322
                        Lab1Test.java
```

In all the projects, the README file will provide additional instructions on how to build and run the projects.
Note that there are separate directories for the source code and unit tests.
We will rely on writing JUnit tests for our labs and assignments.

Building your project

To build your project, simply run **`mvn clean compile`** from the command line. Running the command should produce an output like below:

```
[12:30:06 ~/projects/comp322-s2015-projects/lab1]
shamsimam@MacbookPro $ mvn clean compile
[INFO] Scanning for projects...
[INFO] ------------------------------------------------------------------------
[INFO] Building lab1 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
...
Downloading: http://www.cs.rice.edu/~vs3/hjlib/code/maven-repo/edu/rice/hjlib-cooperative/0.1.4-SNAPSHOT
/hjlib-cooperative-0.1.4-SNAPSHOT.jar
...
Downloaded: http://www.cs.rice.edu/~vs3/hjlib/code/maven-repo/edu/rice/hjlib-cooperative/0.1.4-SNAPSHOT
/hjlib-cooperative-0.1.4-SNAPSHOT.jar (447 KB at 208.5 KB/sec)
[INFO]
[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ lab1 ---
[INFO]
[INFO] --- maven-dependency-plugin:2.9:properties (default) @ lab1 ---
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ lab1 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /Users/shamsimam/projects/comp322-s2015-projects/lab1/src/main
/resources
[INFO]
[INFO] --- maven-compiler-plugin:2.3.2:compile (default-compile) @ lab1 ---
[INFO] Compiling 2 source files to /Users/shamsimam/projects/comp322-s2015-projects/lab1/target/classes
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.352s
[INFO] Finished at: Sun Jan 04 12:33:31 CST 2015
[INFO] Final Memory: 17M/102M
[INFO] ------------------------------------------------------------------------
```

Note that maven automatically downloaded the hjlib-cooperative-0.1.4-SNAPSHOT.jar file, this is maven's automatic dependency management at work. Jar files are only downloaded if they have not been previously downloaded by maven. One of the benefits of using maven will be that you will receive automatic updates to the HJlib jar files as we progress through the course.

Running unit tests

To run the unit tests, run the **`mvn clean compile test`** command:

```
[12:33:31 ~/projects/comp322-s2015-projects/lab1]
shamsimam@MacbookPro $ mvn clean compile test
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building lab1 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
[INFO] ...
[INFO] --- maven-surefire-plugin:2.17:test (default-test) @ lab1 ---
[INFO] Surefire report directory: /Users/shamsimam/projects/comp322-s2015-projects/lab1/target/surefire-
reports

-------------------------------------------------------
 T E S T S
-------------------------------------------------------
objc[23579]: Class JavaLaunchHelper is implemented in both /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk
/Contents/Home/jre/bin/java and /Library/Java/JavaVirtualMachines/jdk1.8.0_25.jdk/Contents/Home/jre/lib
/libinstrument.dylib. One of the two will be used. Which one is undefined.
Running edu.rice.comp322.Lab1Test
Lab1Test.testSimple() starts...
Lab1Test.testSimple() ends.
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.163 sec - in edu.rice.comp322.Lab1Test

Results :

Tests run: 1, Failures: 0, Errors: 0, Skipped: 0

[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 3.917s
[INFO] Finished at: Sun Jan 04 12:36:59 CST 2015
[INFO] Final Memory: 24M/211M
[INFO] ------------------------------------------------------------------------
```

If all the tests run successfully, a successful build is reported. As you work through your labs and assignments running the unit tests will provide an inital sanitary check on whether your implementations work as exepected in terms of providing the correct result.

Running the main program

If you instead want to run your individual programs, we will have configured the maven pom.xml file to allow your to run the individual classes. Please refer to the README files for instructions on how to run the individual programs. For example, running the HelloWorld example from lab1 requires the following command: **`mvn clean compile exec:exec -Phelloworld`**

```
[12:37:02 ~/projects/comp322-s2015-projects/lab1]
shamsimam@MacbookPro $ mvn clean compile exec:exec -Phelloworld
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] Building lab1 1.0-SNAPSHOT
[INFO] ------------------------------------------------------------------------
[INFO]
...
[INFO] --- exec-maven-plugin:1.3.2:exec (default-cli) @ lab1 ---
First: HelloWorld
Second: HelloWorld
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 2.599s
[INFO] Finished at: Sun Jan 04 12:40:54 CST 2015
[INFO] Final Memory: 17M/214M
[INFO] ------------------------------------------------------------------------
```

Using IntelliJ with your Maven project

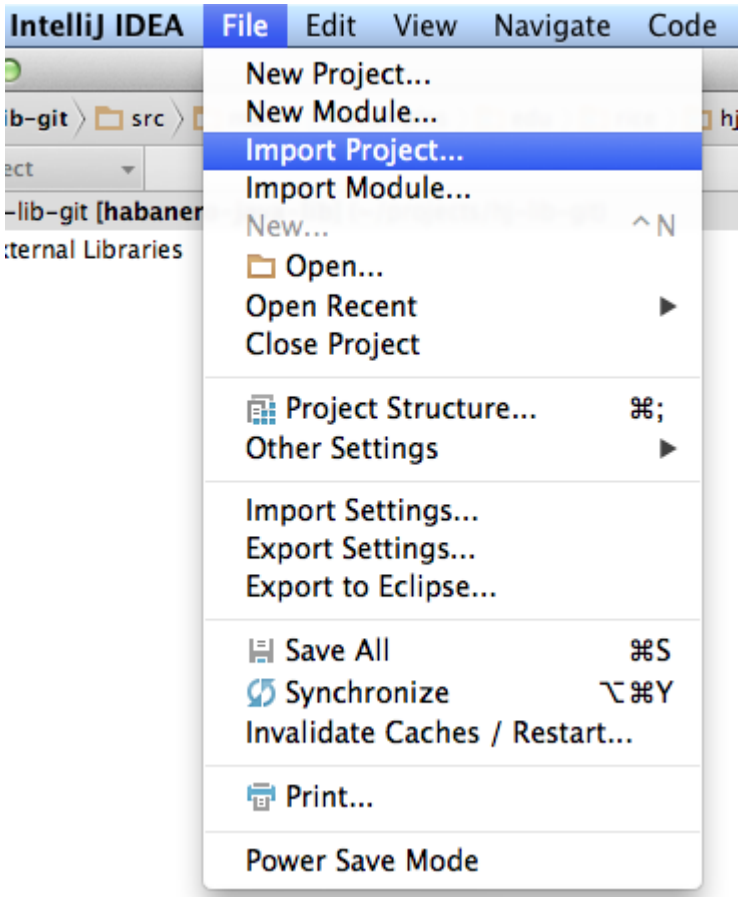
IntelliJ IDEA fully integrates with Maven version 2.2 and later versions, allowing you to create or import Maven modules, download artifacts and perform the goals of the build lifecycle and plugins. If you want to use an existing Maven project, you can import it directly by opening its pom.xml file. When a Maven project is imported, IntelliJ IDEA analyzes the pom.xml file and automatically downloads the necessary dependencies.

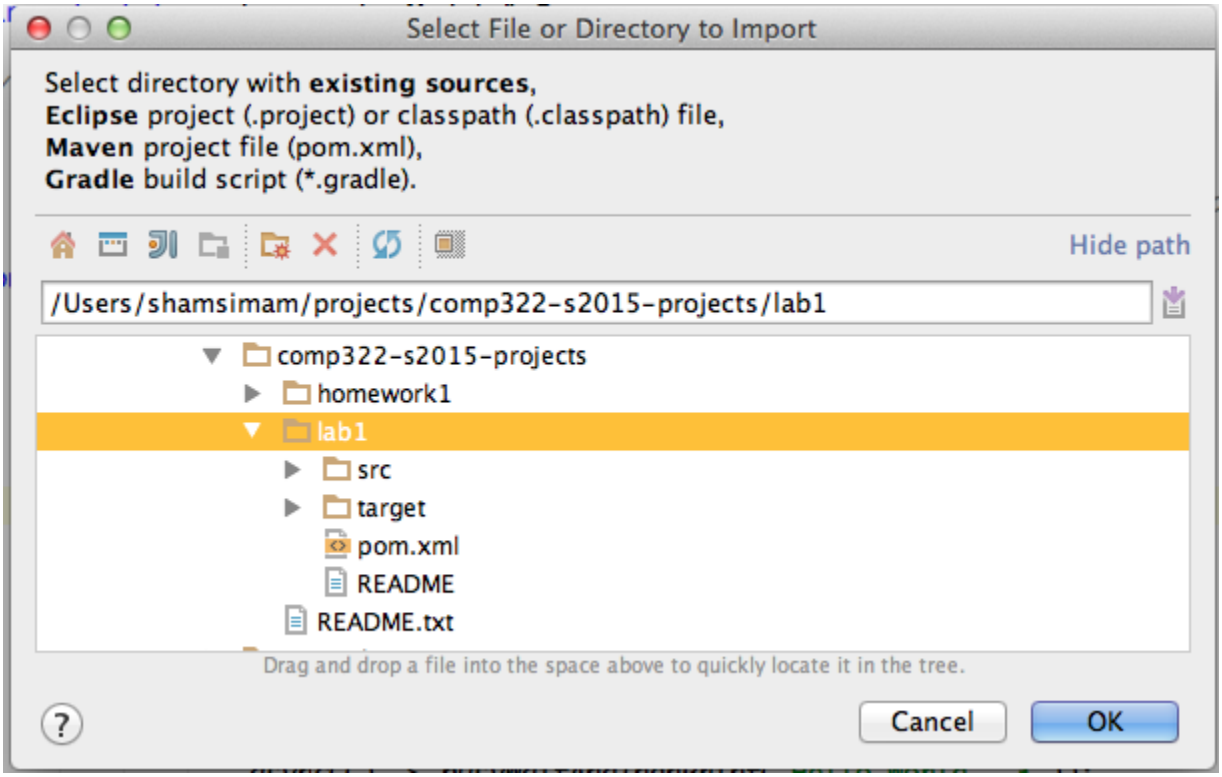
IntelliJ IDEA provides two ways of running the Maven goals:

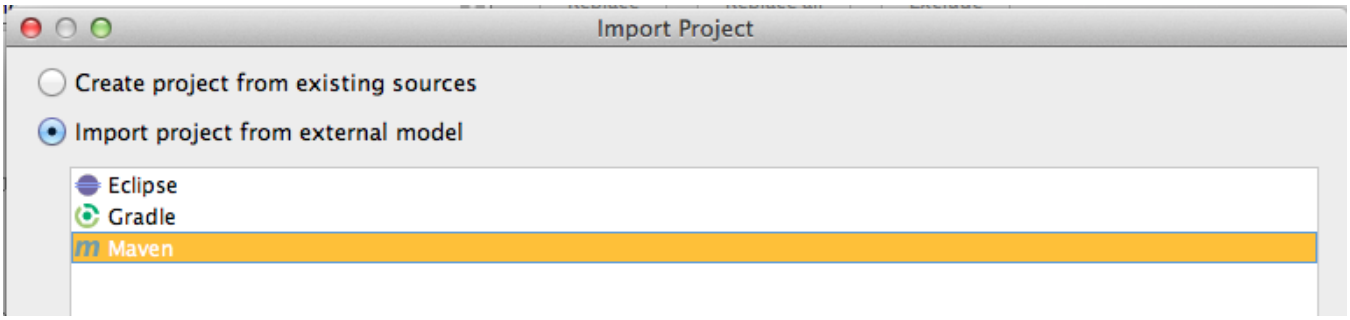
- Create run/debug configuration and launch it.

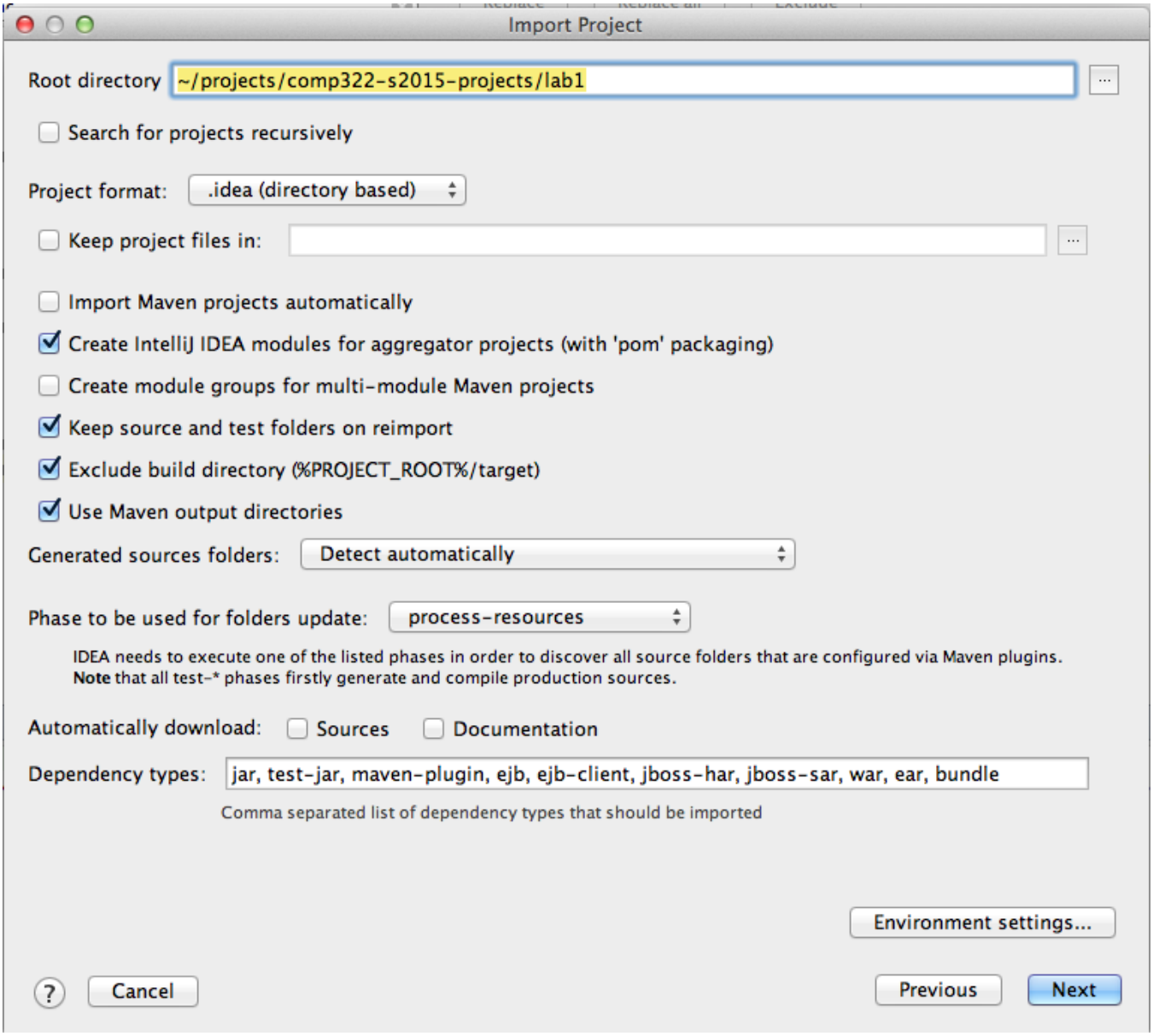
- Use the Run Maven Build command in the Maven Projects tool window. This way doesn't require any run/debug configuration.

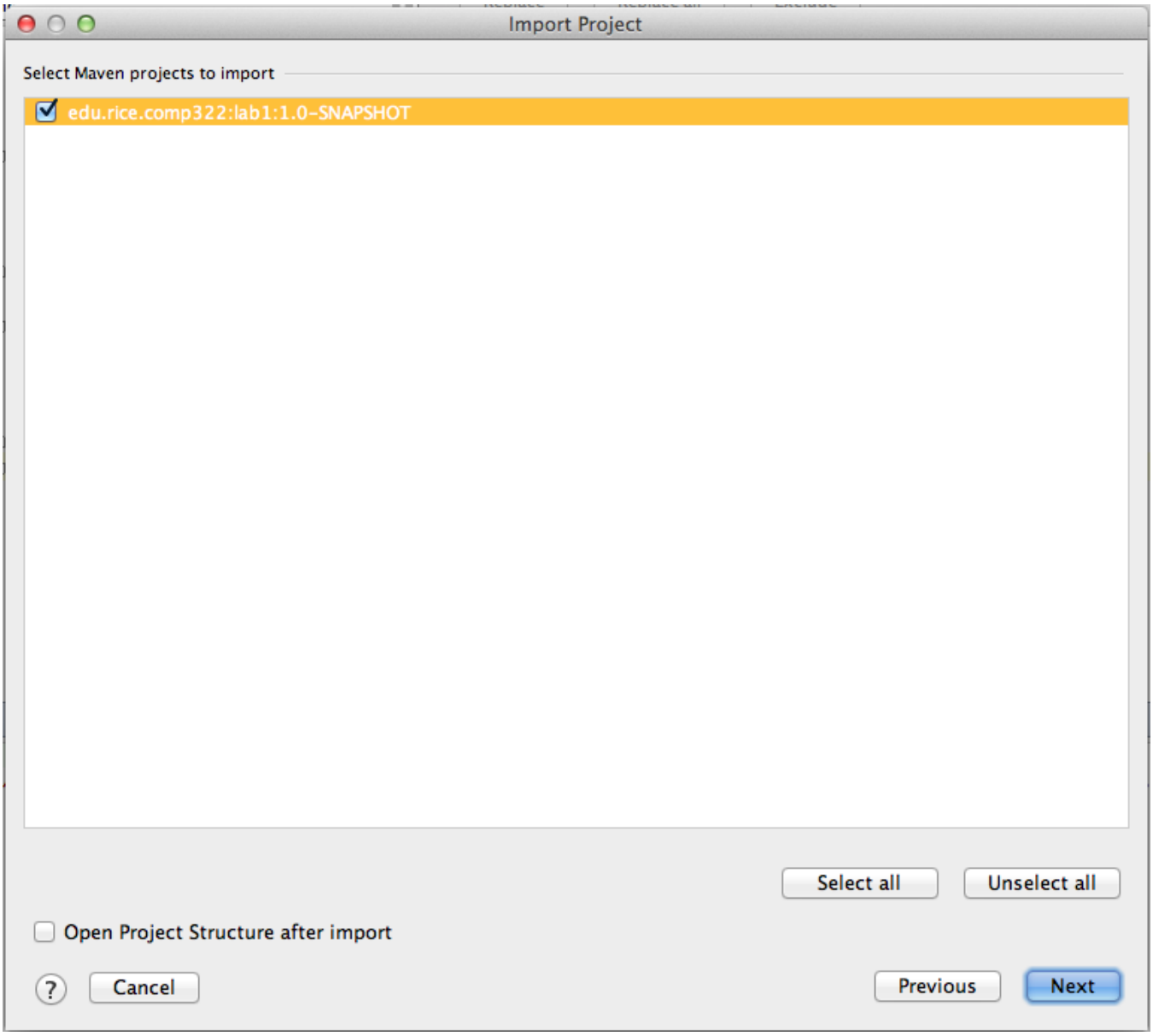
To import lab1 as a maven project simply follow the following steps as shown in the images:



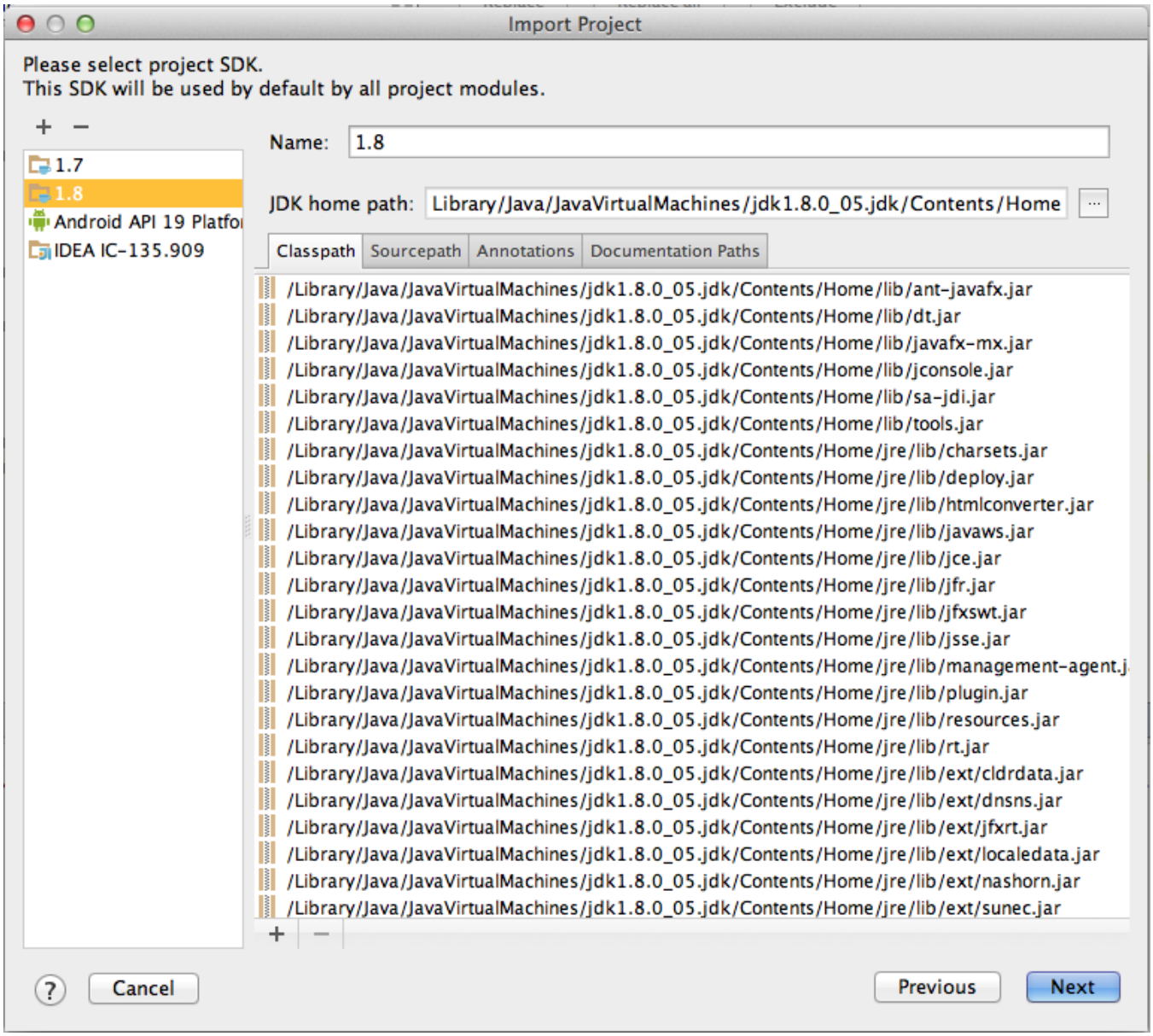


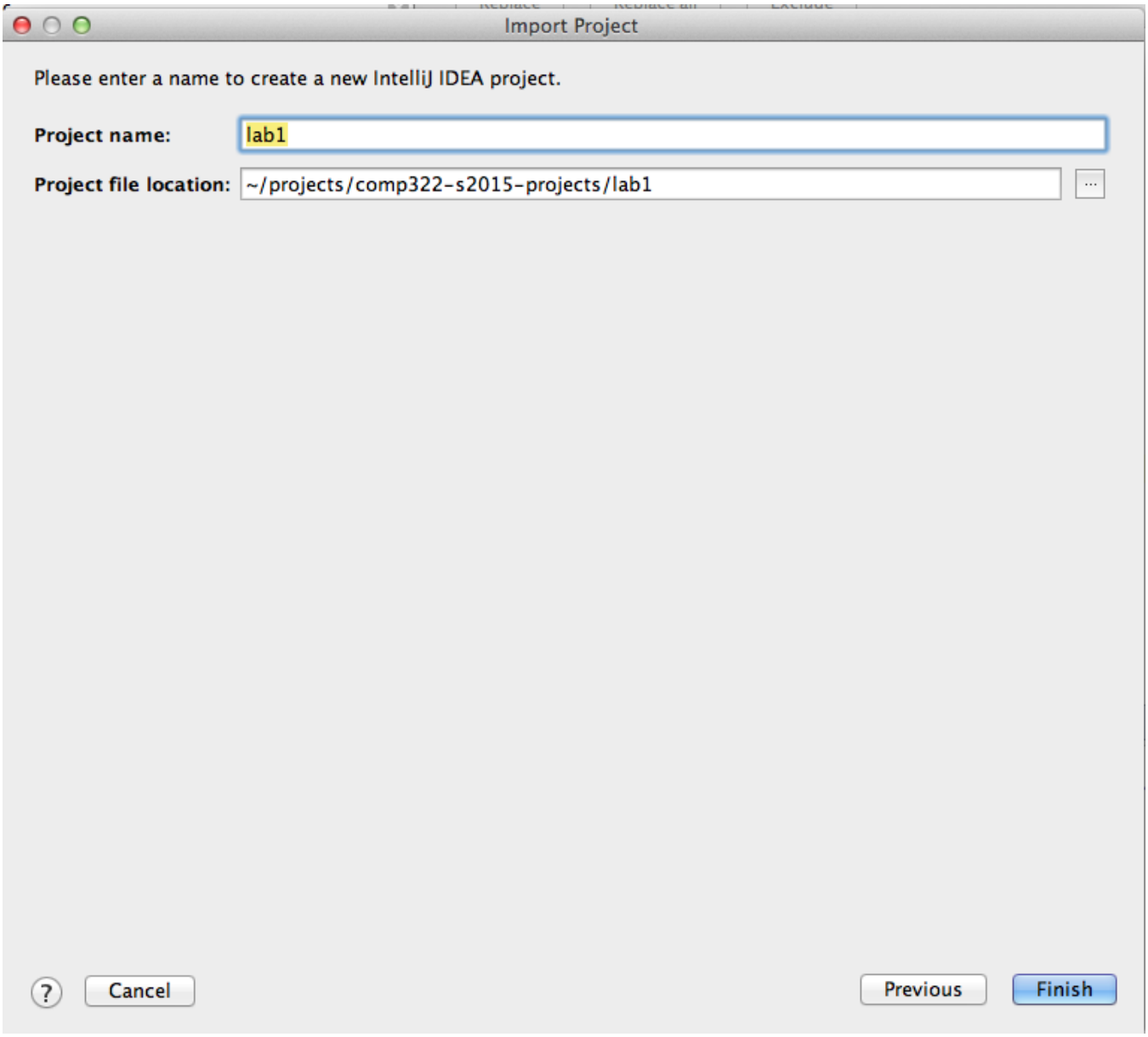

Import Project
Create project from existing sources
Import project from external model
Eclipse
Gradle
Maven


Import Project
Root directory
~/projects/comp322-s2015-projects/lab1
Search for projects recursively
Project format:
.idea (directory based)
Keep project files in:
Import Maven projects automatically
Create IntelliJ IDEA modules for aggregator projects (with 'pom' packaging)
Create module groups for multi-module Maven projects
Keep source and test folders on reimport
Exclude build directory (%PROJECT_ROOT%/target)
Use Maven output directories
Generated sources folders:
Detect automatically
Phase to be used for folders update:
process-resources
IDEA needs to execute one of the listed phases in order to discover all source folders that are configured via Maven plugins.
Note that all test-* phases firstly generate and compile production sources.
Automatically download:
Sources
Documentation
Dependency types:
jar, test-jar, maven-plugin, ejb, ejb-client, jboss-har, jboss-sar, war, ear, bundle
Comma separated list of dependency types that should be imported
Environment settings...
Cancel
Previous
Next

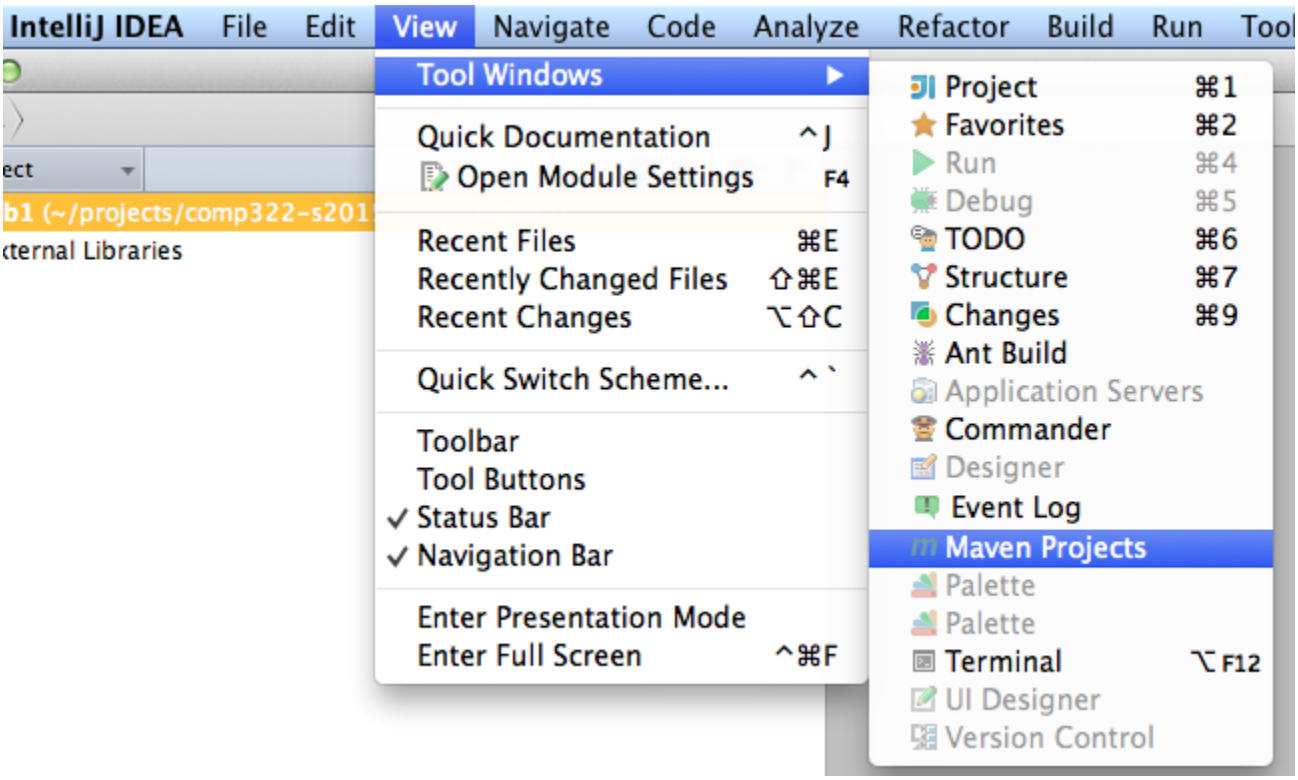


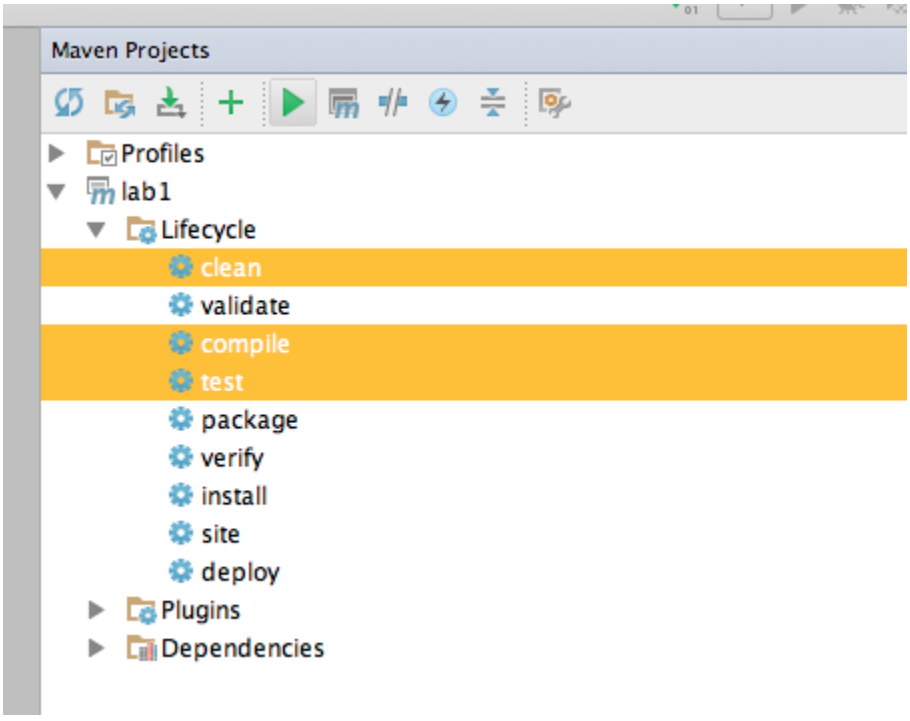
Ensure that you are using the JDK8 distribution in your machine.


Import Project
Please select project SDK.
This SDK will be used by default by all project modules.
1.7
1.8
Android API 19 Platfo
IDEA IC-135.909
Name: 1.8
JDK home path: Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home
Classpath
Sourcepath
Annotations
Documentation Paths
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/ant-javafx.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/dt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/javafx-mx.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/jconsole.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/sa-jdi.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/lib/tools.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/charsets.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/deploy.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/htmlconverter.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/javaws.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/jce.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/jfr.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/jfxswt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/jsse.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/management-agent.j
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/plugin.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/resources.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/rt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/cldrdata.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/dnsns.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/jfxrt.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/localedata.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/nashorn.jar
/Library/Java/JavaVirtualMachines/jdk1.8.0_05.jdk/Contents/Home/jre/lib/ext/sunec.jar
Cancel
Previous
Next



Open the maven plugin window to execute the maven build commands from IntelliJ. Of course, you can still right click on the individual files to run them by editing the run configuration to include the javaagent as described in the download and set up article.


IntelliJ IDEA
File
Edit
View
Navigate
Code
Analyze
Refactor
Build
Run
Tool Windows
Quick Documentation ^J
Open Module Settings F4
Recent Files ⌘E
Recently Changed Files ⇧⌘E
Recent Changes ⌥⇧C
Quick Switch Scheme... ^`
Toolbar
Tool Buttons
Status Bar
Navigation Bar
Enter Presentation Mode
Enter Full Screen ^⌘F
Project ⌘1
Favorites ⌘2
Run ⌘4
Debug ⌘5
TODO ⌘6
Structure ⌘7
Changes ⌘9
Ant Build
Application Servers
Commander
Designer
Event Log
Maven Projects
Palette
Palette
Terminal ⌥F12
UI Designer
Version Control


Maven Projects
Profiles
lab1
Lifecycle
clean
validate
compile
test
package
verify
install
site
deploy
Plugins
Dependencies