Lab 3 - Building with Maven Tycho

The goal of this lab is to set up a build process for the application based on Maven Tycho.

Start by pointing Eclipse at the `webapps/root/labs/lab-3` folder contained in the tutorial root. Import the projects into the workspace.

Make sure that there are no spaces in the path to this workspace. A bug in the Maven Tycho publishing process will cause problems if you have spaces in your path.

In this lab you will do the following:

1. Build the bootstrapper application using using Tycho.

2. Build the additional feature using Tycho.

Build the core application using using Tycho.

Prepare Maven and repositories for the lab.

Verify that Maven is correctly installed by entering `mvn -version` at a command prompt. Check that you get a valid response and that the version of Maven is 3.0 or greater.

If Maven is not correctly installed, you may need to install Maven 3 or configure your path so that it's on your system path. An archive containing Maven 3 can be found in the `osgi-arch-patterns/maven-dist` folder.

Create a release engineering project and build the bootstrapper bundle.

Note that we have modified the `eclipse.product` file to enable it to work with Maven and p2 repositories. Start-level and auto-start information has been added on the product **Configuration** page. Without this Maven Tycho will not create a proper `config.ini` file.

1. Select **File > New > Project..** from the main menu. Select **General > Project** from the list of wizards and click **Next**.

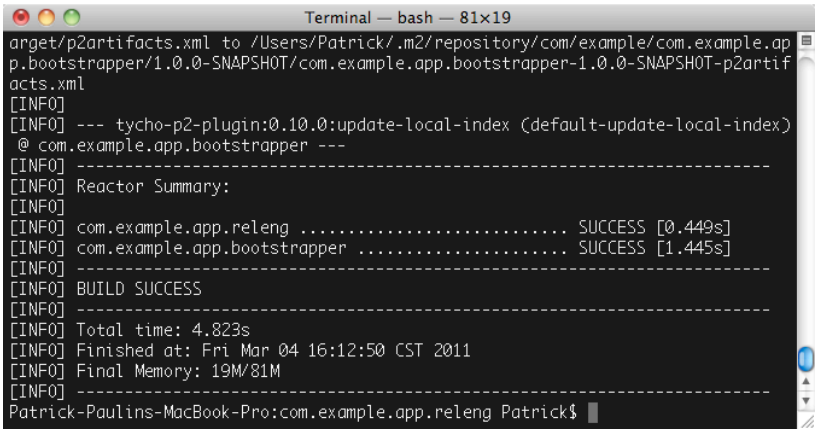
2. On the first page of the wizard, enter `com.example.app.releng` as the **Project name**. Leave everything else as is and click **Finish**.

3. Copy the file `com.example.app.releng.pom.xml` from the `extra-files` project into the new project. Rename it to `pom.xml`.

4. Locate the `configuration` element for the `target-platform-configuration` plugin. Comment out the environments that are not appropriate for your machine. Note that you may need to add `_64` to the end of the `arch` element if you're running a 64 bit JVM. This has been done for the Mac OS X environment already.

5. Copy the file `com.example.app.bootstrapper.pom.xml` into the `com.example.app.bootstrapper` bundle. Rename it to `pom.xml`.

6. We can now build the bootstrapper bundle. From a command prompt, move to the `com.example.app.releng` folder in your workspace. Enter `mvn clean package` and verify that the build completes successfully.

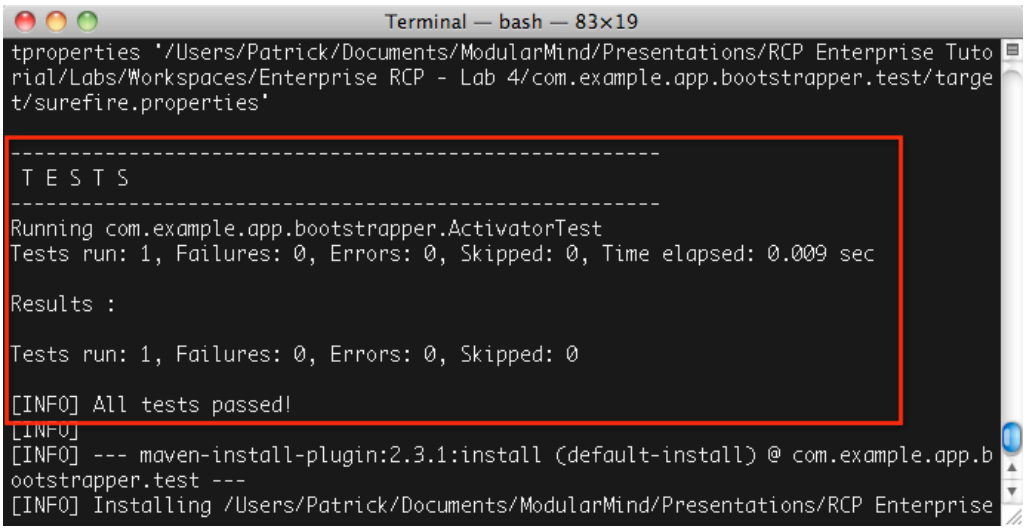


Add the test fragment to the build.

1. Copy the file `com.example.app.bootstrapper.test.pom.xml` into the `com.example.app.bootstrapper.test` project and rename it to `pom.xml`.

2. Open the parent `pom.xml` file in the `com.example.app.releng` project. Locate the `modules` element and add a new `module` sub-element that references the `com.example.app.bootstrapper.test` fragment.

3. Re-run the build from the command line. The build should complete successfully and you should also see output indicating that the unit test ran successfully.



Add the feature to the build.

1. Copy the file `com.example.app.feature.pom.xml` into the `com.example.app.feature` project and rename it to `pom.xml`.

2. Open the parent `pom.xml` file in the `com.example.app.releng` project. Locate the `modules` element and add a new `module` sub-element that references the `com.example.app.feature` project.

3. Re-run the build from the command line. The build should complete successfully and you should also see output indicating that the feature was built successfully.

Add the product to the build.

1. Copy the file `com.example.app.product.pom.xml` into the `com.example.app.product` project and rename it to `pom.xml`.

2. Open the `eclipse.product` file and enter `com.example.app.bootstrapper.product.id` into the **ID** field under **General Information**. This field is necessary for the Maven p2 director plugin to function properly.

3. Open the parent `pom.xml` file in the `com.example.app.releng` project. Locate the `modules` element and add a new `module` sub-element that references the `com.example.app.product` project.

4. Open the `Application` class in the `com.example.app.bootstrapper` bundle. Locate the line where the UPDATE_SITE_URL is declared (should be around line 87).

5. We need to modify this variable because Tycho places the generated update site in a sub-folder of the project. Add `/target/site` to the end of the URL.

6. Re-run the build from the command line. The build should complete successfully and you should also see output indicating that the product was built successfully.

Install and run bootstrapper.

1. Refresh the `com.example.app.product` project and locate the `target/products` folder. In this folder is a zip archive containing the application.

2. Extract this archive somewhere on your local machine.

3. For Mac OS X users only, Maven Tycho does not create a properly configured `eclipse.ini` file. Locate the appropriate INI file in the `extra-files` project and copy it into the extracted application. Rename it to `eclipse.ini`.

   In Finder, you will need to right click on the `Eclipse` application bundle and select **Show Package Contents** on the context menu. The INI file is located in the `Contents/MacOS` directory.

   For those of you interested in why this is necessary, see the following bug report:

   https://issues.sonatype.org/browse/TYCHO-595

4. Launch the application if and note that you get an error. The perspective cannot be found because the update site has not been built yet. We'll do that next.

Build the additional feature using Tycho.

1. Create a project called `com.example.app.perspectives.releng`. Copy the file `com.example.app.perspectives.releng.pom.xml` into the new project and rename it to `pom.xml`.

2. Open the `pom.xml` file and locate the `repository` element. Modify the `url` sub-element so that it points to the `local-p2-repository` folder we created earlier.

3. Copy the file `com.example.app.perspectives.pom.xml` into the `com.example.app.perspectives` project and rename it to `pom.xml`.

4. Copy the file `com.example.app.perspectives.feature.pom.xml` into the `com.example.app.perspectives.feature` project and rename it to `pom.xml`.

5. Copy the file `com.example.app.perspectives.p2.pom.xml` into the `com.example.app.perspectives.p2` project and rename it to `pom.xml`.

6. Before doing a build, we need to repair the `site.xml` file in the `com.example.app.perspectives.p2` project. This file is modified when doing a **PDE Build > Build Site** command, and the change causes the Maven Tycho build process to fail.

   To fix the problem, remove `com.example.app.perspectives.feature` from the list in the `site.xml` file and re-add it.

7. In a terminal window, navigate to the `com.example.app.perspectives.releng` directory. Do a `mvn clean package` and the build should complete successfully.

8. Refresh the `com.example.app.perspectives.p2 project`. You'll see the update site has been created as both a folder and as an archive.

9. Re-run the application. The new feature should be installed from the update site and the perspective should now appear. Delete the installed application. This lab is now complete.