How to Create an Eclipse RCP project with Maven/Tycho Integration

Adel Ferdjoukh

Altran Technologies

november 2020

Table of contents

About this tutorial I

Context

Very often, developers arrive on a huge RCP project with already thousands of lines of code. It is not always easy to start an RCP project from scratch. Especially if one wants to use a build tool such as Maven/Tycho.

Objectives

- Learn how to start an RCP project from scratch
- Understand the function of each plugin of an RCP project
- Use the Maven/Tycho layout to automate build and release
- Give a ready-to-use template for RCP projet with Maven/Tycho integration
- Discover use useful tips

About this tutorial II

Duration

2-3 hours

Target Audience

Software Developers in Java, Eclipse RCP and Model Driven Engineering

Prerequisite

- Your computer
- Download the latest version of Eclipse (google search). Among the available packages, you can choose the following :
  - Eclipse IDE for Java and DSL Developers
  - Eclipse Modelling Tools
- Programming in Java and OOP
- Some Maven knowloedge

About this tutorial III

Related Topics

MDE Foundations Training by the same Author

Template Project

In order to fully understand this tutorial, we created a example of RCP project with the Maven/Tycho integration

Github repository

git clone https://github.com/ferdjoukh/RCPMavenTychoStructure.git

Build the RCP project

1. Clone the git repository that contains the full example
2. If your have installed maven in your computer, open a terminal and execute mvn clean install
3. If maven is not installed, open you eclipse and import the project. Right click on the parent pom, then maven build
4. Choose goals : clean install
5. The first build takes between 1 and 3 minutes depending on the machine

After the success check the content of folder : *releng/org.example.awesomeProject.product/target/products*

This folder contains the zipped version of the built applications (for linux and windows).

Anatomy of an RCP Project I

Why Maven/Tycho ?

Maven is used to automatically build, release and deliver your java project. In RCP since the projects contains tens of plugins, doing the build manually would be a mess. So the pair Maven/Tycho helps to automate the process.

Tycho Layout

- Parent Project
  - *bundles/*
    - plugin 1
    - plugin 2
    - pom.xml
  - *features/*
    - feature 1
    - pom.xml
  - *releng/*
    - target platform porject
    - repository project
    - product project
    - pom.xml
  - *tests/*
    - test project
    - pom.xml
  - pom.xml

Parent project and pom.xml

Bundles

It is a folder that contains all the source-code plug-ins of your projects : model, generated code, GUI code, business code, etc.

Features

A folder that contains the features projects of your application. A feature in the Eclipse ecosystem is a collection of plug-ins that accomplish a set of features.

Releng

A folder that contains all the projects that are mandatory to build your application. It contains at least :

– A product definition
– A target platform
– A repository project

Tests

A folder that contains all the tests plug-ins of the project.

Materialize and Archive Product I

These are the most interesting plugins of Maven/Tycho, they are used to build and zip the eclipse product that results from the compilation of your application.

To learn how to create these goals, check product *pom.xml* file

Create a Project Step by Step I

1. Create the parent project (plugin project)
   - Call it org.example.awesomeProject
   - Specify location (recommand different location from the workspace. Instead use the git repository folder)
   - Unselect the java source option. The parent project does not contain source code.
2. Add the maven nature to the parent project
   - Right click -> Configure -> Convert to Maven Project
   - Choose packaging = pom
   - Put version = 1.0.0-SNAPSHOT (= plugin version)
   - For pom.xml you can use the given pom.xml of *awesome* project
3. Leave eclipse and go to project folder
   - Delete the project from eclipse workspace
   - Create 4 folders : bundles, features, releng and tests

Create a Project Step by Step II

- Delete META-INF, build.properties files and folders
- Import the folder into Eclispe again.

4. Create a new plugin (stored in bundles folder)
    - name it : *org.example.awesomeProject.gui*
    - Be carreful to the location of this plugin. Use : *.../org.example.awesomeProject/bundles/org.example.awesomeProject.gui/*
5. Create a pom.xml file for bundles folder (use given example)
    - Add *gui* plugin to the list of modules
6. Create pom.xml for features, releng and tests folders
7. Create a feature project
    - Add *gui* plugin to the content of feature
    - Create a pom.xml for it (inspire from given example)

Create a Project Step by Step III

8. Create an Update Site project inside releng folder
   - Add the previous feature to it
9. Create an empty project inside releng
   - Create a target definition
   - Populate the target platform (inspire from given example)
10. Create an empty project for the product
    - Create a product definition file
    - Add the create features
    - Add feature *org.eclipse.pltaform* to its Contents
    - Click on add required
    - Create the *pom.xml* file (inspire from given example)
11. Build your product using maven

Create a Project Step by Step IV

- Use goals : clean install

Tips I

1. **Include other files or folders in the final product**
   - Copy you files into the features project
   - Open the build.properties files and copy the following lines

```
root.folder.examples = ./examples
root.win32.win32.x86_64.folder.JRE_1.8_181_64b= ./JRE_1.8_181_
```

2. **Use a specific Java VM to run your product**
   - Once your VM was identified (in the previous example, it is included with the source code)
   - Manual : open the eclipse.ini file and add the lines

```
-vm JRE_1.8_181_64b\bin\javaw.exe
```

   - Generated : Open the product file, go to launching tab, choose win32 and put this line to program arguments

```
-vm JRE_1.8_181_64b\bin\javaw.exe
```

Tips II

3. **Unpack a plugin after the installation**
   - Open MANIFEST.MF file of your plugin
   - Add the following line :

```
Eclipse-BundleShape: dir
```