Massey University

159.251 Software Design and Construction

Tutorial 4 – Using Maven to automate your build

Prerequisites (what you are expected to know before you come to the tutorial)

be able to checkout projects from GIT

a. using a command line and then importing the project from the file system into Eclipse
b. or by using an Eclipse GIT plugin (http://www.eclipse.org/egit/)
   for IntelliJ, read here: https://www.jetbrains.com/help/idea/maven.html

Objectives

1. convert an existing project into Maven project
2. implement a goal
3. integrate your code quality checker frameworks (PMD, FindBugs etc..) into your existing maven project.
4. Generate reports

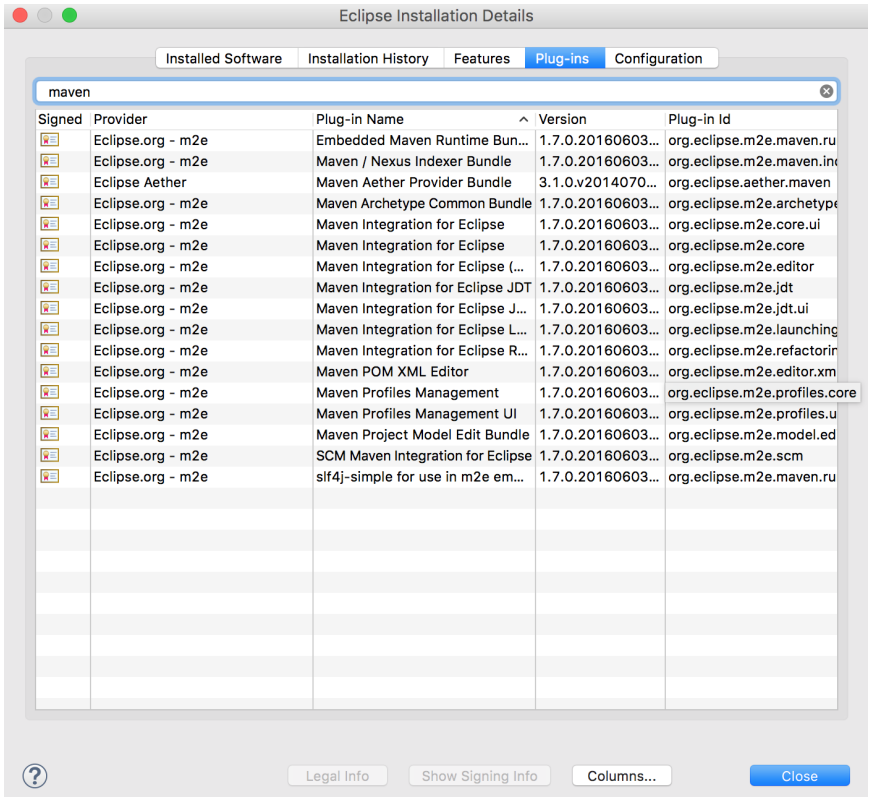
Description

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

First, check that you have Eclipse Maven plugin installed in your Eclipse environment. To check, in Eclipse go to

Help -> Installation Details -> Plug-ins

Search for Maven (or **m2e** = the name of the plugin) in the search bar.



If Maven is not installed, you will then need to install it from the following update site:
http://download.eclipse.org/technology/m2e/releases

If you want to to install Maven in your computer (and not only your Eclipse!), follow the instructions listed here
https://maven.apache.org/install.html

TASK 1 :

Clone a project from :

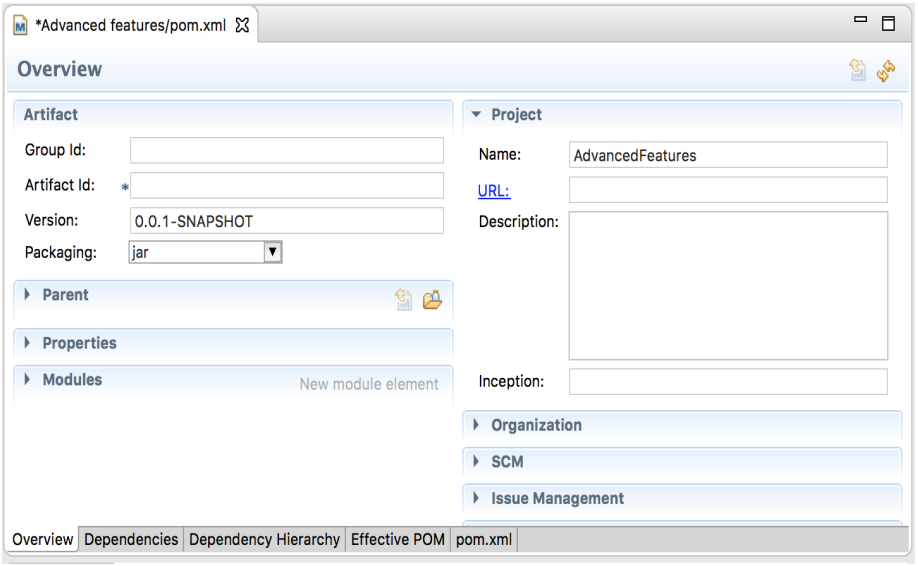
https://aatahir@bitbucket.org/aatahir/159251.git

This is a normal Java Eclipse project, you will need to **convert this project into Maven project**.

To convert into Maven, right click on

Project -> Configure -> Convert to Maven Project

This will create a POM.xml file in your project. This file contains information about dependencies, Plugins etc that are needed by your project. Notice that this is an XML file (read more: https://en.wikipedia.org/wiki/XML)



Go to **Dependencies**, and then add the following information

Group Id: org.apache.maven.plugins
Artifact Id: maven-pmd-plugin
Version: 3.8

This will add the PMD maven plugin to your project.

Now after you finish, go to pom.xml and check if these have been added correctly.
You can do this directly (by editing) in the pom file by adding the information into the xml format.
Here what you need (notice the code in read and the comment highlighted):

```
<project>
  ...
  <build>
    <!-- To define the plugin version in your parent POM -->
    <pluginManagement>
      <plugins>
        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-pmd-plugin</artifactId>
          <version>3.8</version>
        </plugin>
        ...
      </plugins>
    </pluginManagement>
    <!-- To use the plugin goals in your POM or parent POM -->
```

```
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.8</version>
    </plugin>
    ...
  </plugins>
 </build>
 ...
 <!-- To use the report goals in your POM or parent POM -->
 <reporting>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-pmd-plugin</artifactId>
      <version>3.8</version>
    </plugin>
    ...
  </plugins>
 </reporting>
 ...
</project>
```

---

insert the highlighted code into the right places.

Maven runs through defined goals. You have to specify which goals do you want to run. PMD Plugin goals are:

Plugin Documentation

Goal	Report?	Description

pmd:check	No	Fail the build if there were any PMD violations in the source code.
pmd:cpd	Yes	Creates a report for PMD's CPD tool. See http://pmd.sourceforge.net/cpd.html for more detail.
pmd:cpd-check	No	Fail the build if there were any CPD violations in the source code.
pmd:help	No	Display help information on maven-pmd-plugin. Call `mvn pmd:help -Ddetail=true -Dgoal=<goal-name>` to display parameter details.
pmd:pmd	Yes	Creates a PMD report.

To test this, run the build with the default target in Eclipse.
right click on **Project -> run as -> maven build**

it will then ask you for a defined goal**. Your first goal is to check**, then you should run the following:
pmd:check

This should fail, not because it doesn't work but because it found violations (read the description in the table above!).
The next task is to generate reports about those found violations. This will generate an HTML file (pmd.html) under ~/target/site

Read the table above to know which goal you should use to generate a report!

Task 2: Adding PMD maven plugin to an existing pom file.

checkout the Apache Commons Lang project from git:

```
git clone https://github.com/apache/commons-lang.git
```

This project already contains a pom.xml file. What you need to do now is to edit this file so we can find violations using pmd and then generate a report through maven.

But first make sure it is already a maven project.
Right click on Project -> run as
If you can see maven then the setup is ok. If this does not shown then this project needs to be converted into a maven project.
Right click on **Project -> Configure -> Convert to Maven Project**

This will convert your project into a maven project.

Your task is to modify the pom file and add PMD maven plugin. Follow the steps mentioned above in Task 1.

You can find more information on https://maven.apache.org

Challenge for good students

1. follow the same steps to add findbugs and (https://gleclaire.github.io/findbugs-maven-plugin/) and Checkstyle (https://maven.apache.org/plugins/maven-checkstyle-plugin/)
2. Try to understand more about Maven and why it is important. We will have a lecture on automated build at the end of this course. You also learn about another tool called ANT (http://ant.apache.org/). There are other build tools that are also popular such as Gradle. Here is a nice comparison of the three tools: https://technologyconversations.com/2014/06/18/build-tools/

What to show:

Reports from the two projects generated by maven. Be ready to answer some questions as well!