2018

**Maven Explained**

**Environment: Windows**

**Vibranarayanan**

Table of Contents

[Daily Learning tracker 3](#_Toc514092074)

[What is Maven 4](#_Toc514092075)

[What is POM 5](#_Toc514092076)

[Maven Objectives 5](#_Toc514092077)

[Maven Build life cycle 5](#_Toc514092078)

[Install Maven 6](#_Toc514092079)

[Prerequisites 6](#_Toc514092080)

[Download Maven 6](#_Toc514092081)

[Setting up Environment Variable 6](#_Toc514092082)

[Verify Maven Installation 7](#_Toc514092083)

[Creating Maven Project Command Line 8](#_Toc514092084)

[Project Initiation 8](#_Toc514092085)

[Folder structure verification after mvn generation 10](#_Toc514092086)

[Java code path 10](#_Toc514092087)

[Junit path 10](#_Toc514092088)

[POM file path 10](#_Toc514092089)

[POM file 10](#_Toc514092090)

[Explore Maven goals/commands (Command Line) 10](#_Toc514092091)

[mvn clean 11](#_Toc514092092)

[mvn compile 11](#_Toc514092093)

[mvn test-compile 12](#_Toc514092094)

[Explore Maven with eclipse IDE 12](#_Toc514092095)

[Maven project creation using eclipse 12](#_Toc514092096)

[Creating Java class file 15](#_Toc514092097)

[Resolve test class dependency in POM.xml 17](#_Toc514092098)

[Checking dependency update 18](#_Toc514092099)

[Before saving POM.xml 18](#_Toc514092100)

[After Saving POM.xml 18](#_Toc514092101)

[How to create a Jar 19](#_Toc514092102)

[Before mvn clean command execution 19](#_Toc514092103)

[How to execute MVN command 19](#_Toc514092104)

[mvn clean 19](#_Toc514092105)

[mvn compile 20](#_Toc514092106)

[mvn test-compile 21](#_Toc514092107)

[mvn test 21](#_Toc514092108)

[surefire-reports 21](#_Toc514092109)

[mvn install 21](#_Toc514092110)

[Transitive dependency 22](#_Toc514092111)

[Dependency hierarchy 22](#_Toc514092112)

[Viewing Junit's POM file 23](#_Toc514092113)

[Excluding Maven Dependencies 23](#_Toc514092114)

[Maven Scope in dependency management 24](#_Toc514092115)

[Test 24](#_Toc514092116)

[Compile 24](#_Toc514092117)

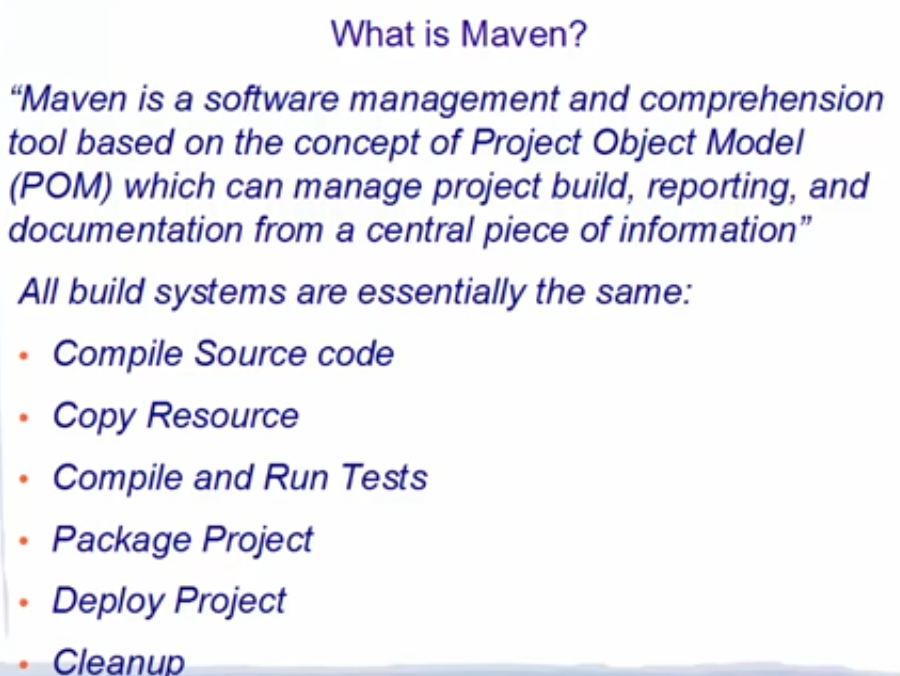
[Runtime 24](#_Toc514092118)

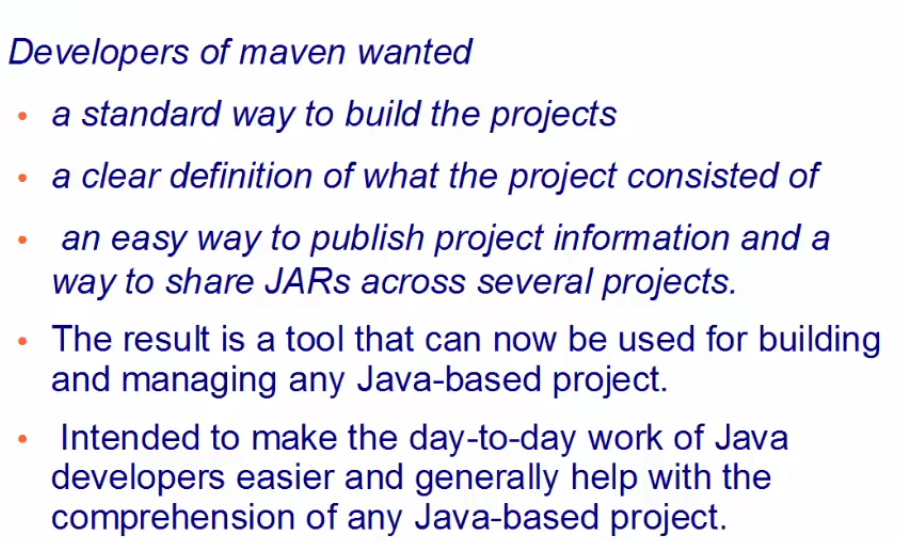
[Provided 24](#_Toc514092119)

# Daily Learning tracker

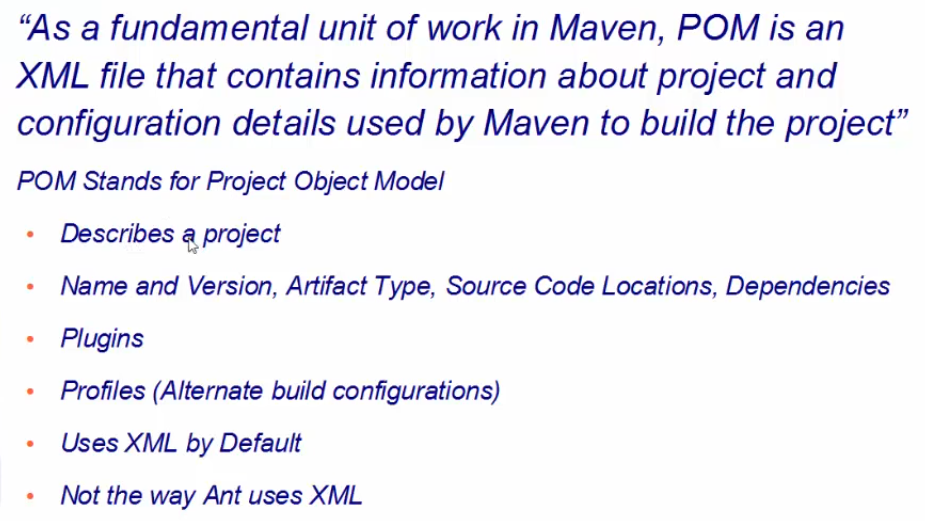
|  |  |
| --- | --- |
| **Date** | **Topics** |
| 05/11/2018 | * Learn maven basics * install maven * create maven project * Explore mvn commands * document learnings/hand-on   Pages (1 - 11) |
| 05/14/2018 | * Explore Maven with eclipse   + eclipse installation and setup * Transitive dependency * Exclusions * Scope   Pages (12 - 24 ) |

# What is Maven





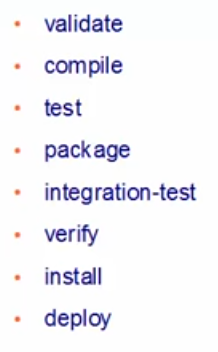
# What is POM



# Maven Objectives

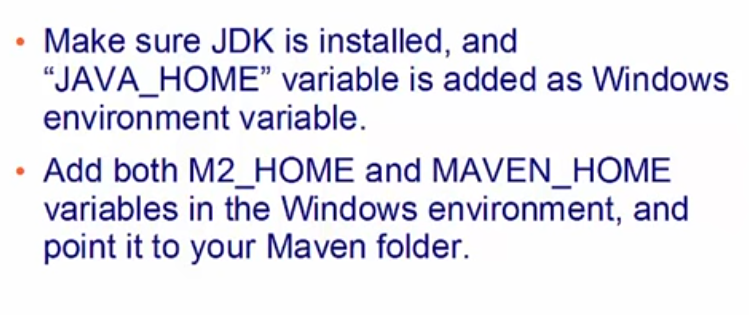


## Maven Build life cycle



# Install Maven

## Prerequisites

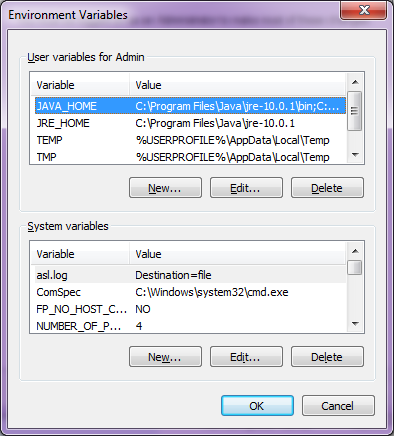


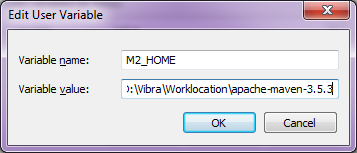
## Download Maven

URL: https://maven.apache.org/download.cgi  
  
Download 2nd option Binary Zip and extract it in your desire location.

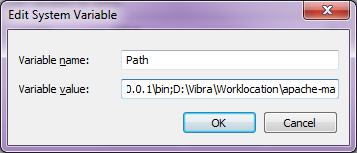
## Setting up Environment Variable

Go to computer properties and select Environment variable.

U  
Under user variable click new and add M2\_HOME variable with Maven extracted directory.



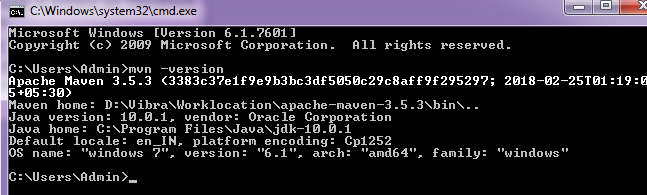
Click Ok and if needed added MAVEN\_HOME. since few projects use this variable with this name.



go to bin directory of maven and copy and append that path with existing path variable. done change existing variable. add ; at the end and past the path

Example: ; D:\Vibra\Worklocation\apache-maven-3.5.3\bin

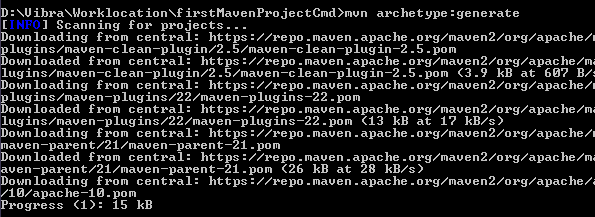
## Verify Maven Installation

execute mvn -version. This will list maven version. seems now all set with maven instalation.

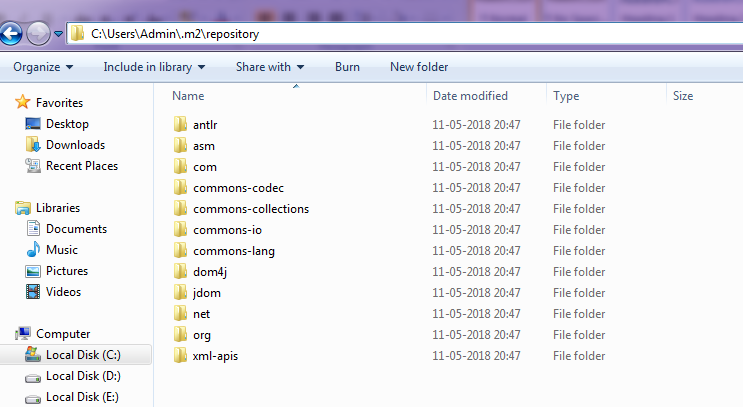
# Creating Maven Project Command Line

## Project Initiation

Step 1: create a project folder Ex: " D:\Vibra\Worklocation\**firstMavenProjectCmd**"  
Step 2: Navigate this folder in command line. and type mvn archetype: generate and enter  

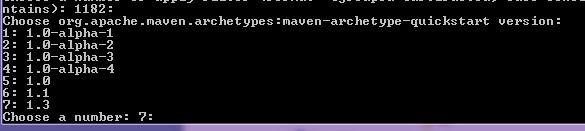
this will download required plug-ins and store it in your local for the first time.



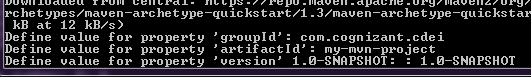
These archetype plug-ins will be stored in your use directory ".m2"

once all archetype plug-ins downloaded this will ask for template id just select enter number shown (1182) or press enter.

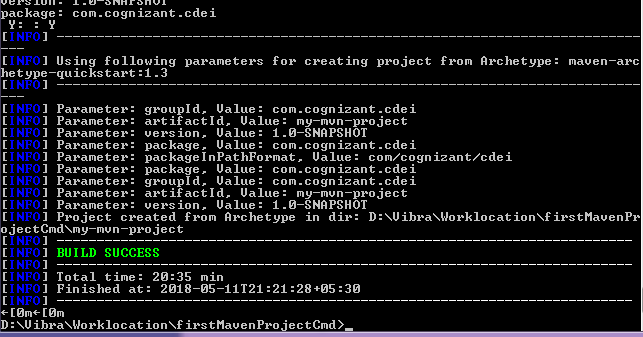




you can select required arche type. we will go with latest one (7) and enter. this will load required items.  
System will ask for groupid, artifactid and version. based on project need/hierarchy you can enter details given below.

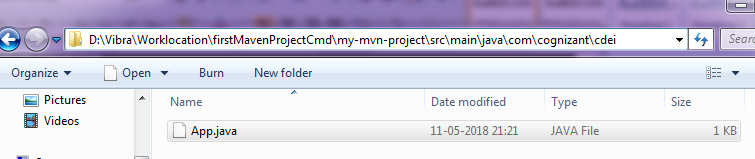


Once you enter all the details tie will start building folder structure for you. you can verify these in selected project location.



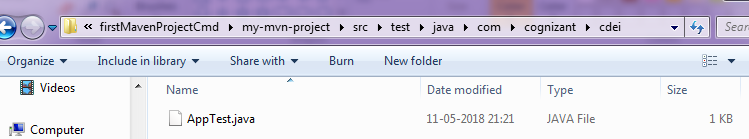
## Folder structure verification after mvn generation

### Java code path

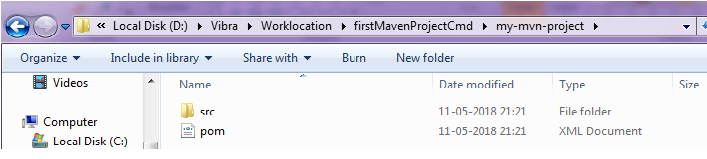


App.java created for us to use and all folder structure created based on our input given in groupid, artifactid and version.

### Junit path



### POM file path



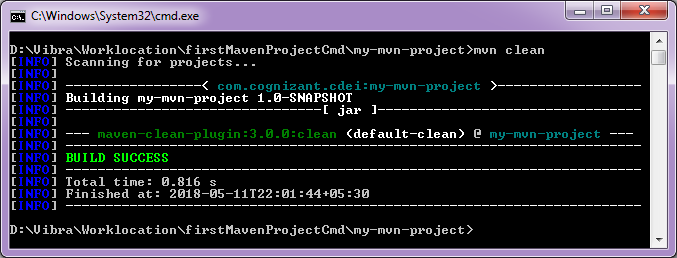
### POM file

  
This POM file created in this process with all the dependencies.

# Explore Maven goals/commands (Command Line)

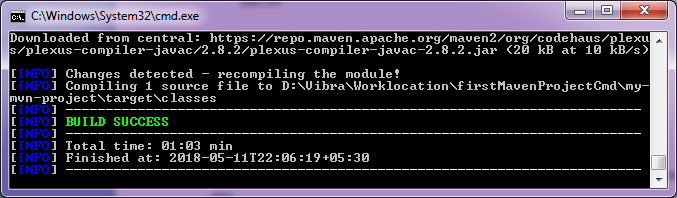
below commands can be used as goals in maven Jenkins Job.

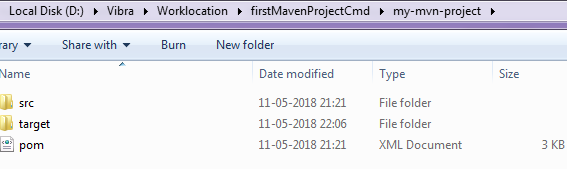
## mvn clean

Go to newly created project's POM file location. This command will clear already created artefacts if any.

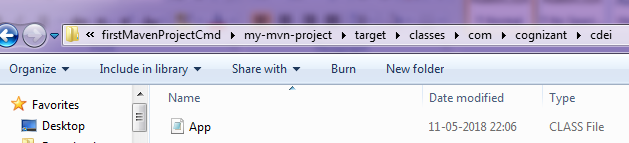
## mvn compile

This command will download required dependencies and compile our java and create .class files.





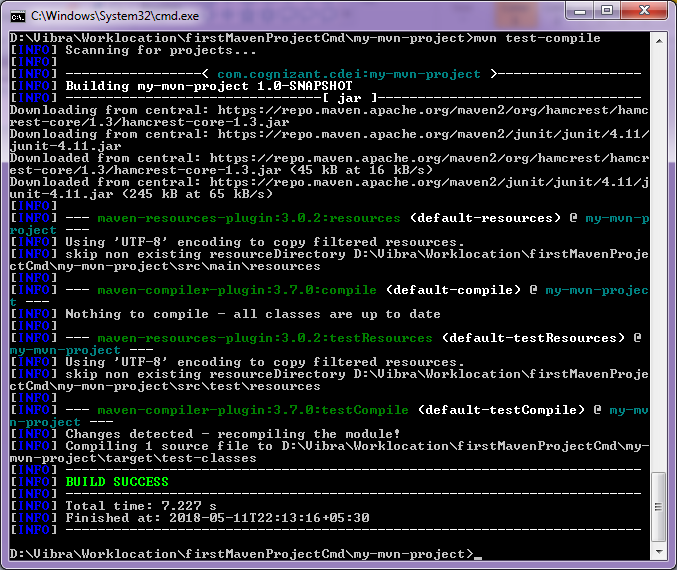
You can see target folder is created under our project, after running this mvn compile command.



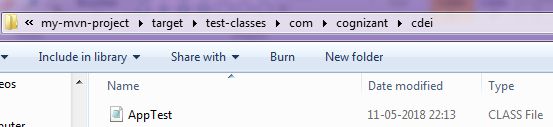
Compiled class file generated under target location.

## mvn test-compile

This command will compile available Junit test case



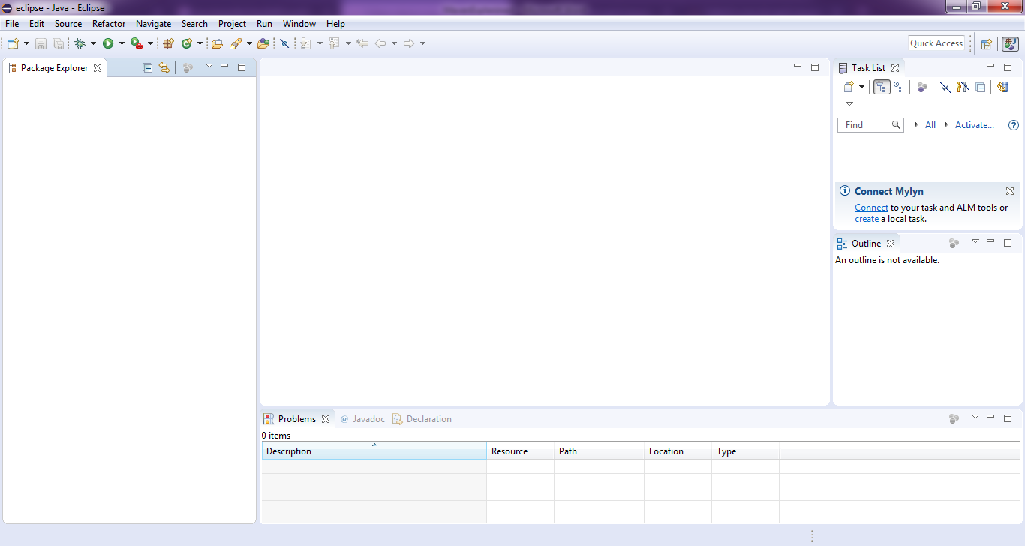
test-classes created under target folder and class files available for further use.



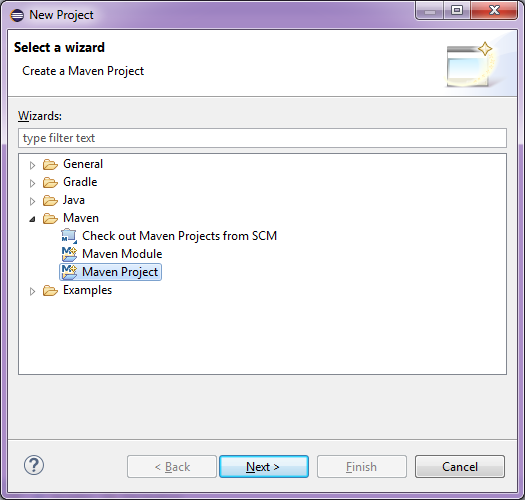
# Explore Maven with eclipse IDE

## Maven project creation using eclipse

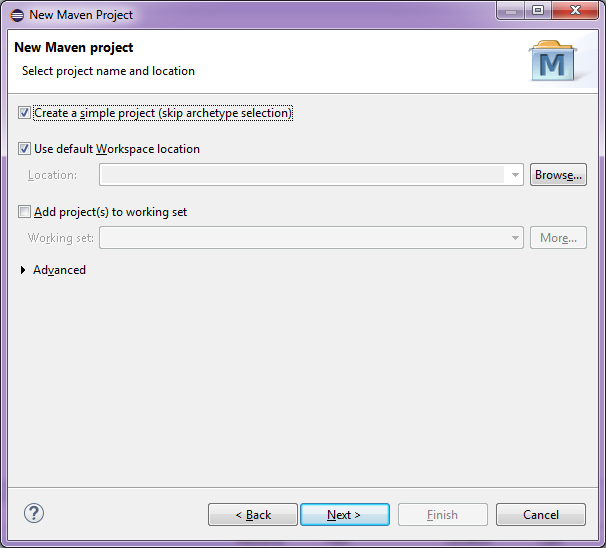
download eclipse from eclipse site and install it in your system. skip introduction screens and go to project view by closing all initial screens.



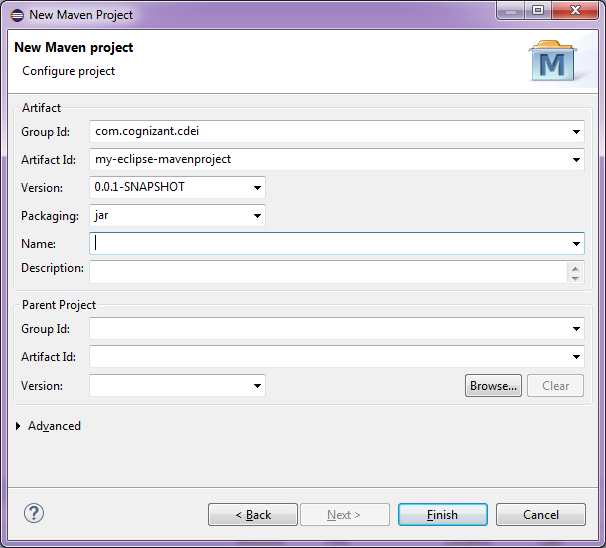
Select File >> New >> Project

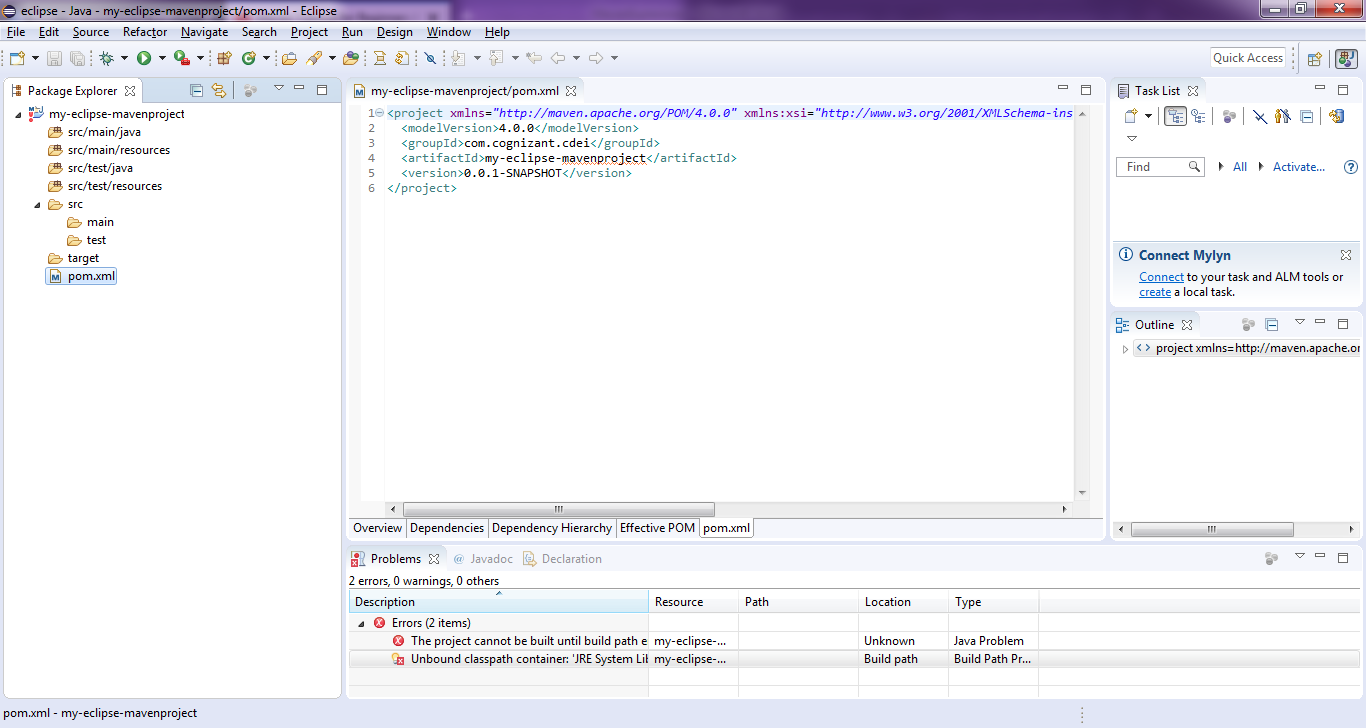


In wizard select/expand Maven project and click next



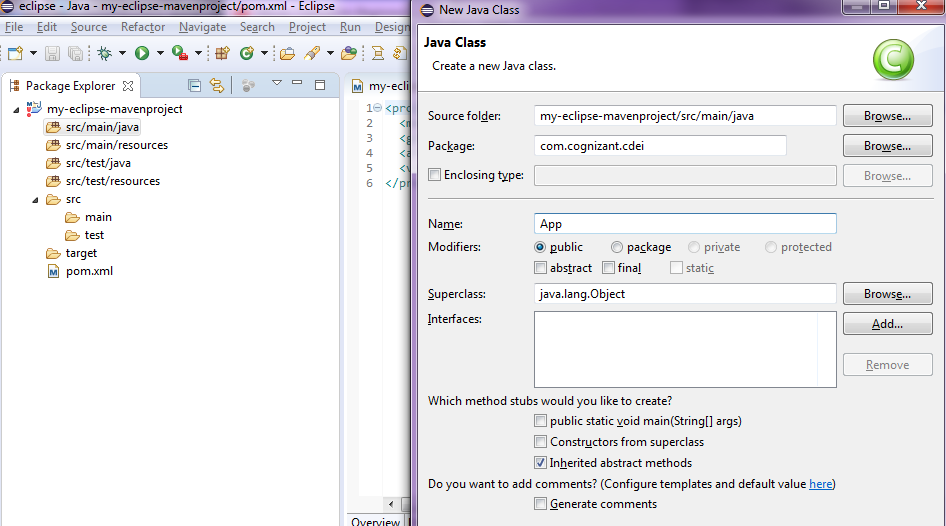
Select simple project and leave other options as it is.

  
Enter gropup id and artifact id and leave other values and click Finish.

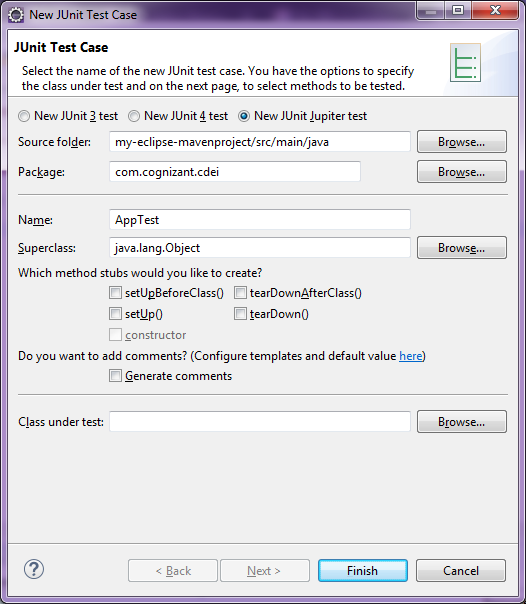


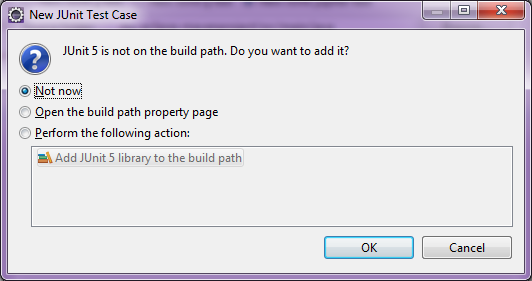
project will be created with folder structure and with POM.xml.

### Creating Java class file

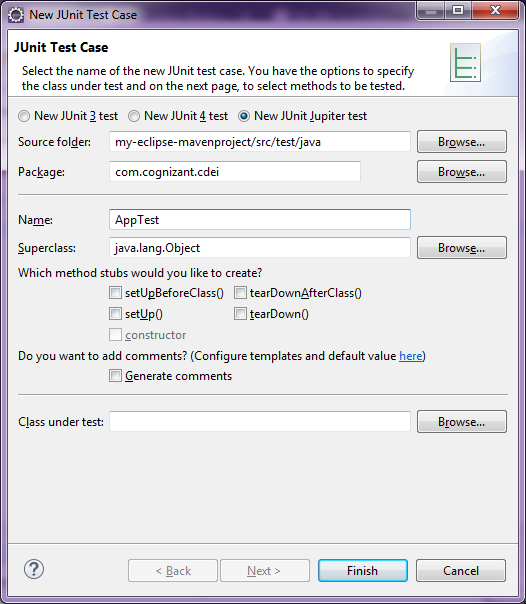


in left menu right click on src/main/java and select "create Class" option and enter details like package and name and click Finish

  
Create Junit test class and click finish



select "Not Now option"



Right click on srt/test package and create new test case. Enter package name and name and click finish.



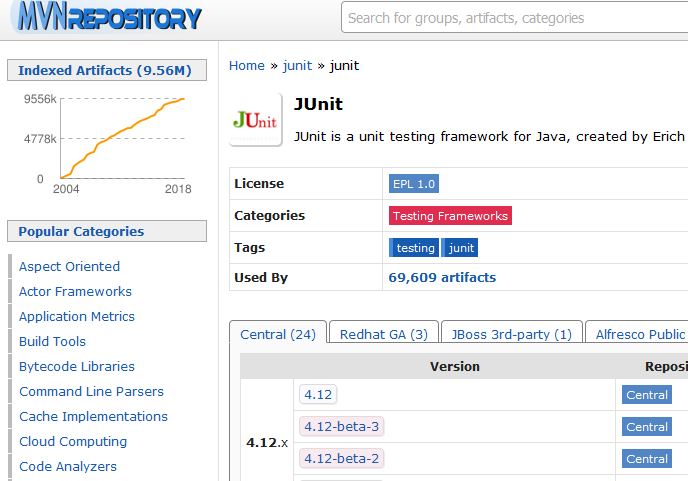
Once done you can see test class created, with error this is due to junit dependency not added in POM.

## Resolve test class dependency in POM.xml

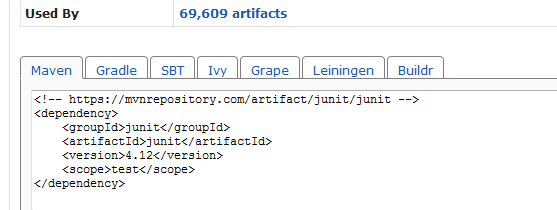
Open POM file and add tag <dependencies></dependencies>.

inside dependencies we need to add junit related dependency.

to get require dependency search for maven junit dependeccy. it will show the results for juit click on this first link "https://mvnrepository.com/artifact/junit/junit" and select the latest version.

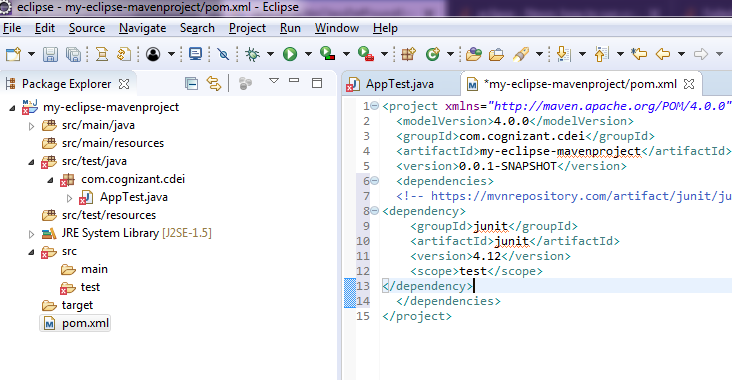


Once you click on latest version that will take you to below page copy dependency content and past it POM(inside dependencies tag)

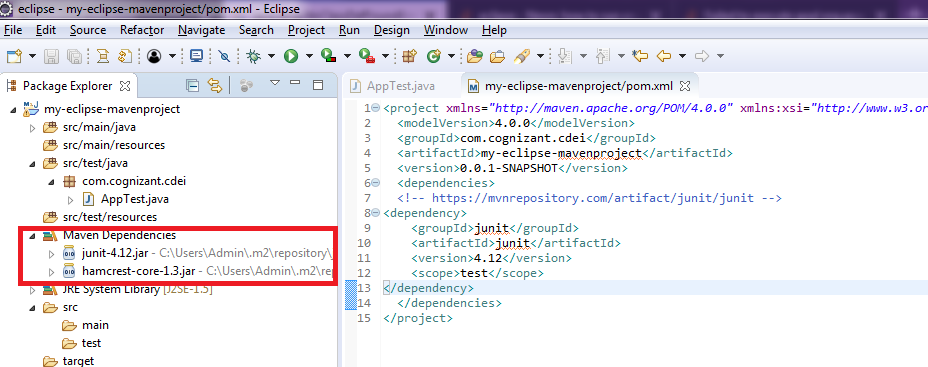


## Checking dependency update

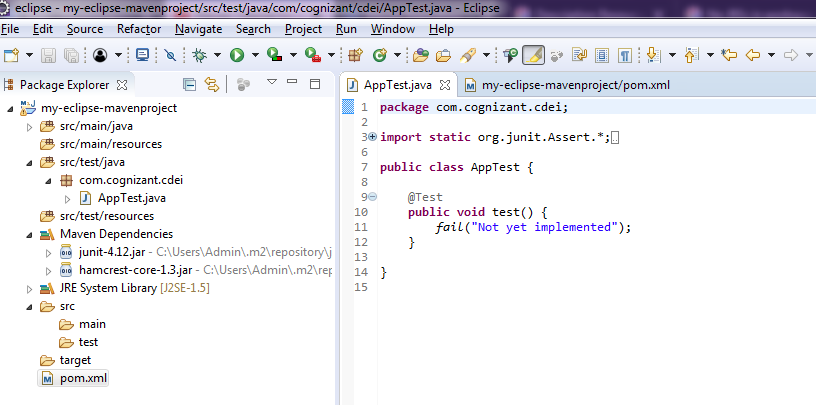
### Before saving POM.xml



## After Saving POM.xml



Maven dependency added with Junit and hamcrest-core jars automatically. and errors in test class resolved automatically.



## How to create a Jar

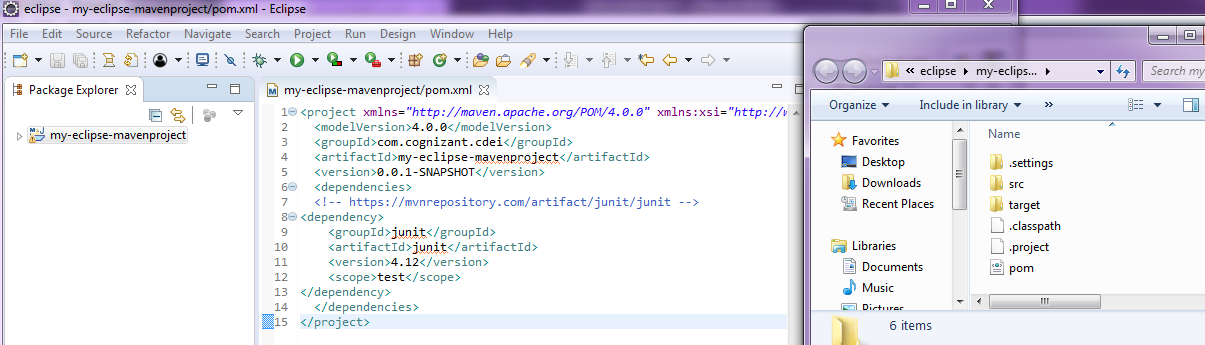
- Create a java class

- Compile created java class files into .class files

- Create java test case file

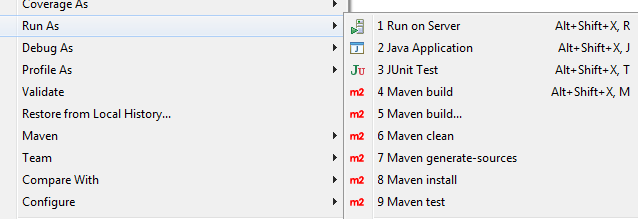
- Compile test files and create .class files. fix any test failures.

### Before mvn clean command execution



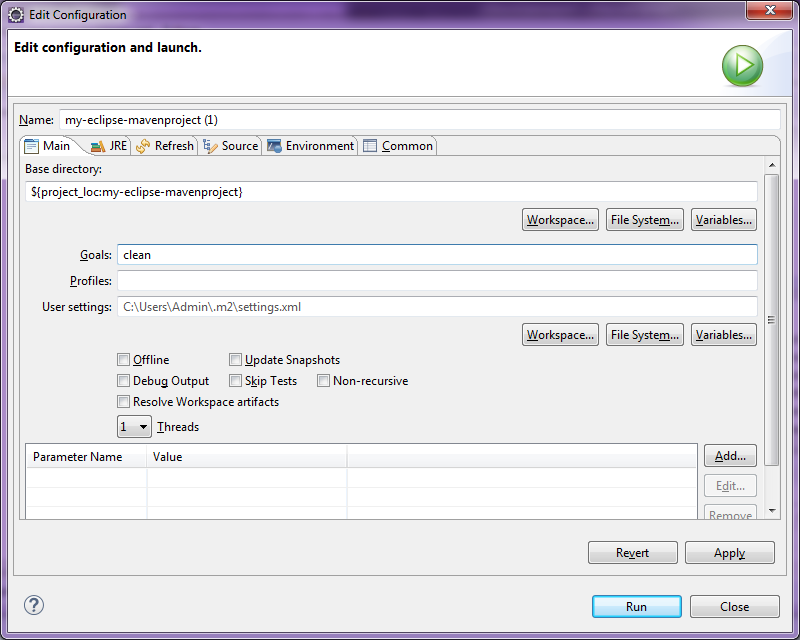
### How to execute MVN command

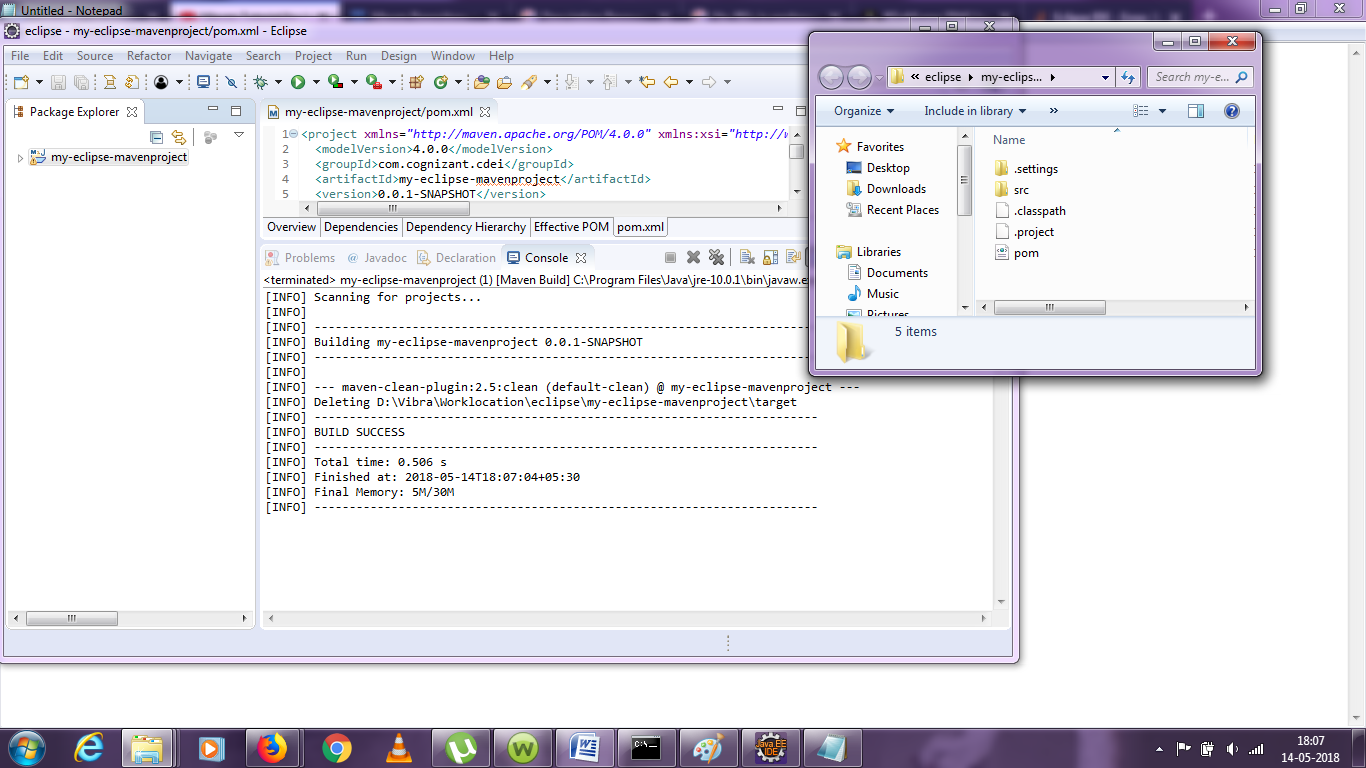
Right click on project >> Run as >> Maven Build



### mvn clean

in below screen, type clean in Goals and click run



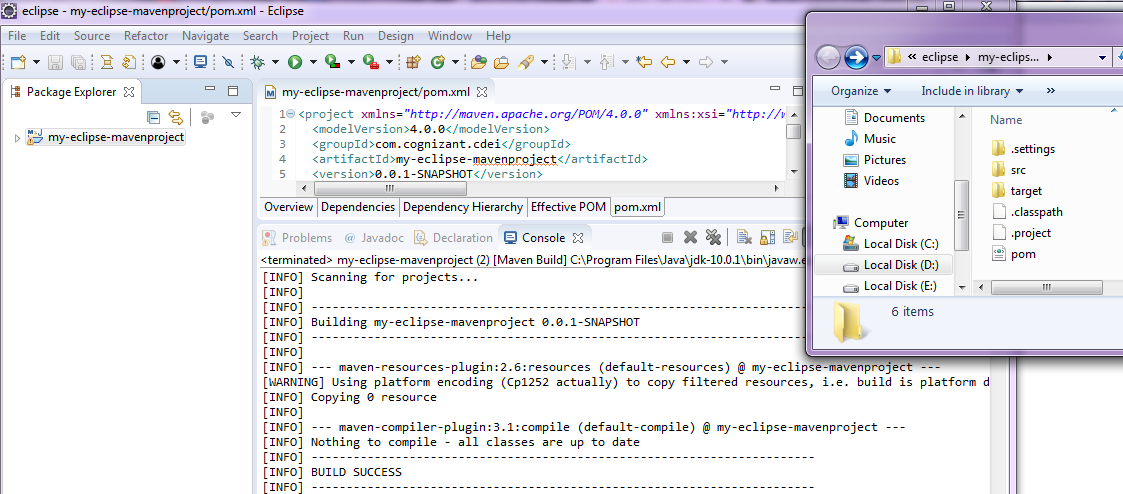


after clean command execution, target files in project dir removed and mvn logs updated with success.

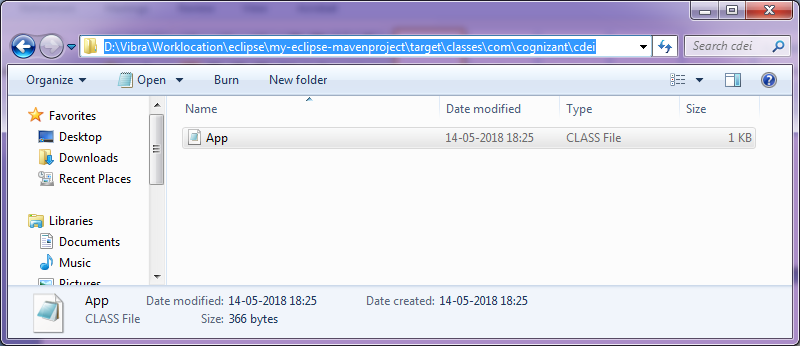
### mvn compile

Right click project >> Run as >> maven build

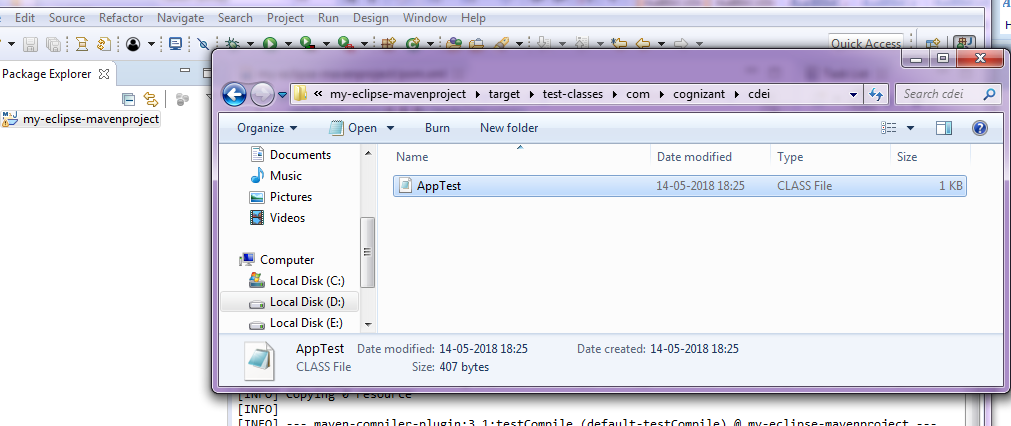
type compile in Goal and click run



target folder is created and inside target App.class is available which is a compiled version of App.java.

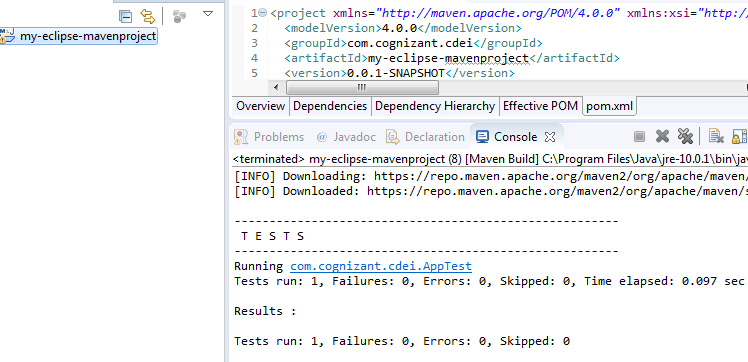


### mvn test-compile

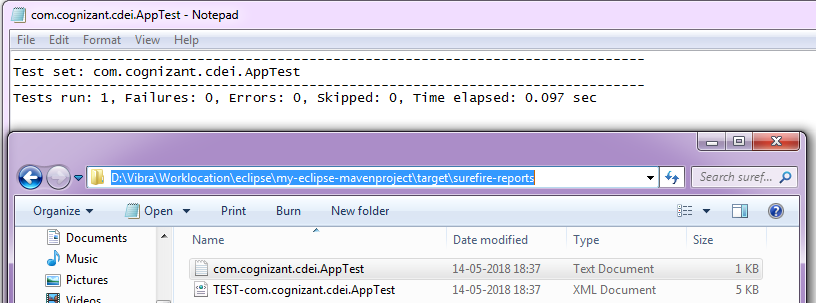


### mvn test

This will run compiled class and will provide the results.



### surefire-reports

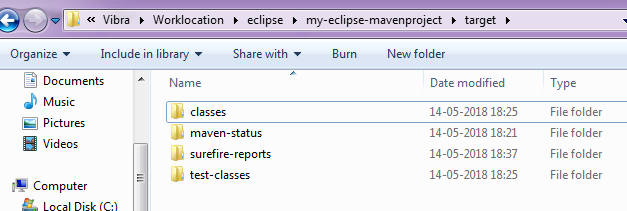


test results will be stored in surefire-reports folder under targets.

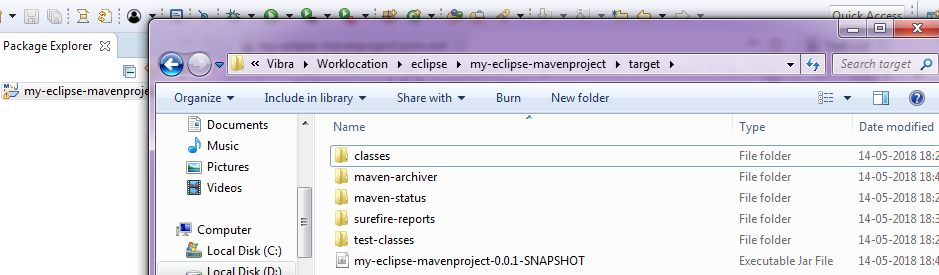
### mvn install

this command will create jar file as needed.

#### Target folder before execution of install command



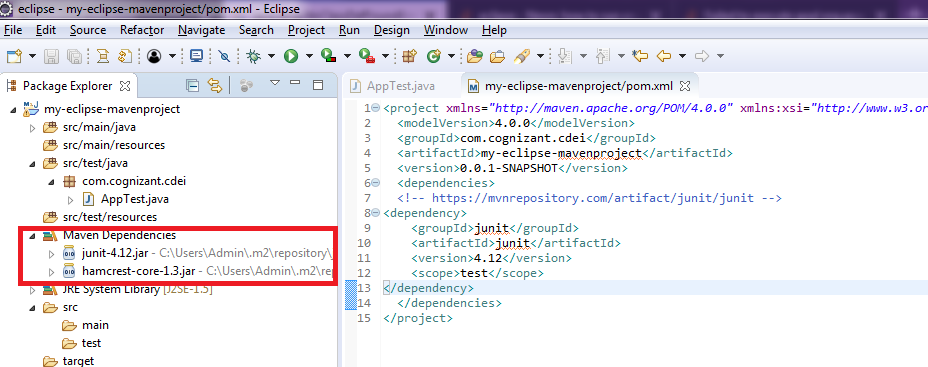
#### Target folder after execution of install command



jar file is created and it is available for use/sharing.

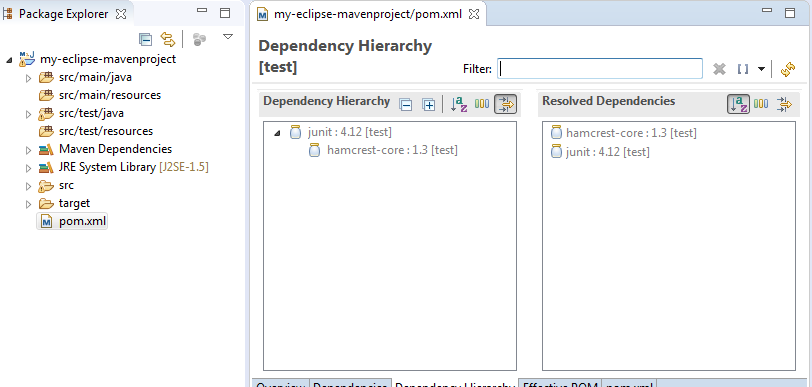
# Transitive dependency

we added junit dependency as part of this POC. you have observed that Junit jar added after adding dependency details in POM file. however you can notice one another jar added "hamcrest-core-1.3.jar" added automatically. since this jar is dependent on junit. this kind of dependency called Transitive dependency.



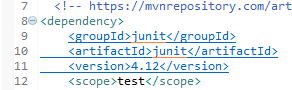
### Dependency hierarchy

This hierarchy can be viewed by clicking pom file and select dependency hierarchy tab.



### Viewing Junit's POM file

Click on POM and press "Ctrl" and hover on dependency, you can see this, link enable for junit. you can click and see junit's POM file and its dependency.

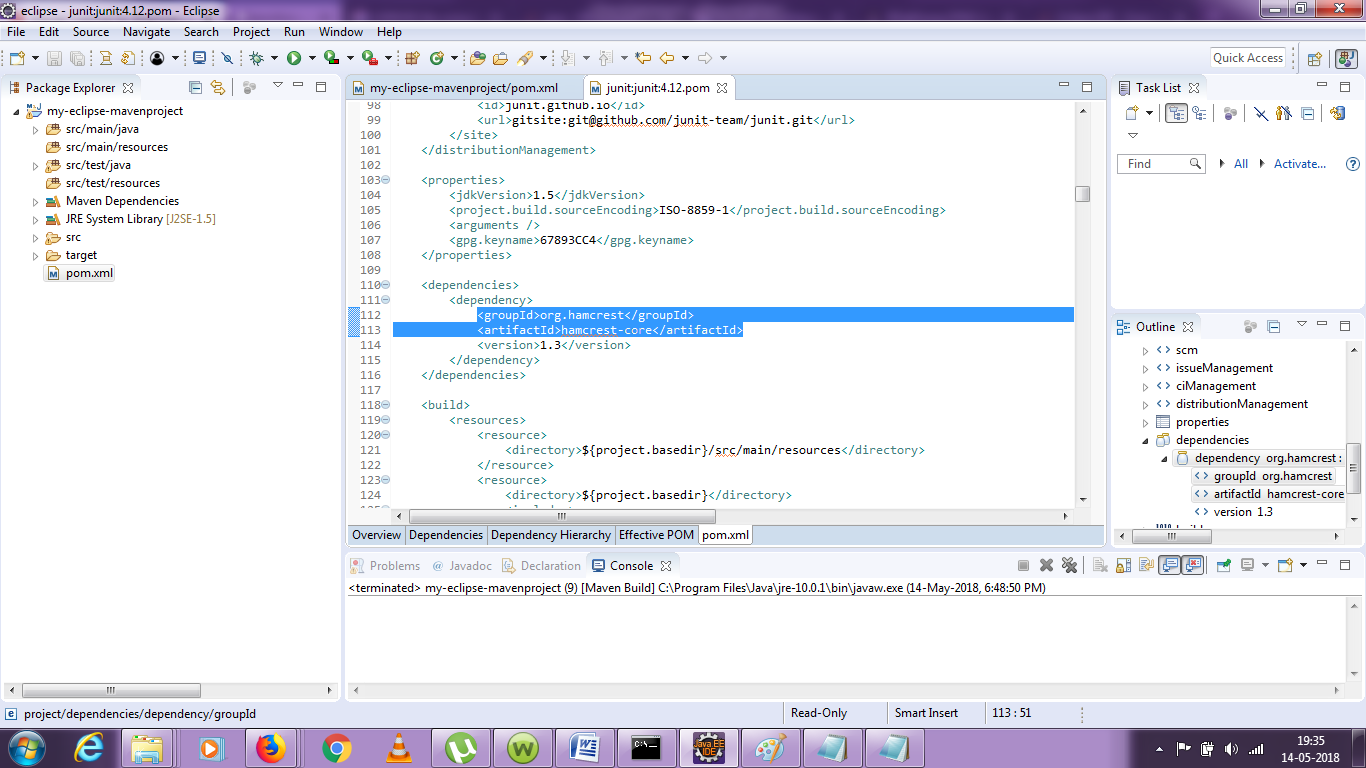


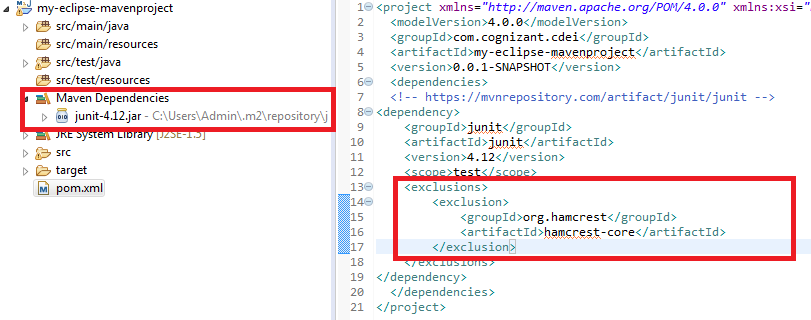


Junit POM file.

## Excluding Maven Dependencies

using this option we can remove/excluded un-wanted jars. consider we added junit dependency and . by default hamcrest jar added. using exclusion option we can remove this jar. To exclude we need gropid and artifact id. we can get these by navigating to junit's POM. copy those and past it in our POM.





now harmcrest jar is removed, likewise we can use exclusion option

## Maven Scope in dependency management

### Test

in our POC for junit dependency scope marked as test, which means this code is scoped to test alone not for other compiled code.

### Compile

Compile is the default scope .If scope is not defined, it will considered as compile.

### Runtime

As name implies, this option will scope required dependency during runtime not in this compile time.

### Provided

much like compile, however these will be provided by another continer like JDK during runtime.