export MAVEN\_HOME=/usr/local/apache-maven/apache-maven-3.5.0

export MAVEN=$MAVEN\_HOME/bin

About maven

We can also create custom maven plugins

Before maven:

Generally before maven if u want to compile a single class file, if that class is using any third party libraries, ,(first u should have downloaded them) u have to include those Libraries located path in the classpath as below

Ex:-

Import org.apaceh.lang3.StringUtils;

// the above class belongs to commonutils librarby , I have downloaded that jar and kept in lib folder

Class mani{

Main{ system.out.println(StringUtils.isnull(“mani”)); }

}

U have to include the jar path in classpath

java –classpath ./lib/\*: ./ Helloworld

Lifecycle

Maven is based on concept of build life cycle (LPG-lifecycle-phase-goal🡪each phase will have lot of goals)

Humans have many phases like our-childhood, young age, old age- and each phase have some tasks

In maven also a lifecycle=set of phases

Each phase =set of many plugin goals (like us in young age, we have goals called job, marriage, kids)

All the work is done by plugins,

Lifecycle and phases provide the framework to call plugin goals in a sequence

**There are three built-in lifecycles:**

* default: the main lifecycle, as it's responsible for project deployment
* clean: to clean the project and remove all files generated by the previous build
* site: to create the project's site documentation

**Each lifecycle consists of a sequence of phases.**

The default build lifecycle consists of 23 phases, as it's the main build lifecycle.

On the other hand, the *clean* life cycle consists of 3 phases, while the *site* lifecycle is made up of 4 phases.

Each of these build lifecycles is defined by a different list of build phases, wherein a build phase represents a stage in the lifecycle.

For example, the default lifecycle comprises of the following phases (for a complete list of the lifecycle phases, refer to the [Lifecycle Reference](https://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html#Lifecycle_Reference)):

Below are the phases of Default life cycle

* validate - validate the project is correct and all necessary information is available
* compile - compile the source code of the project
* test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* package - take the compiled code and package it in its distributable format, such as a JAR.
* verify - run any checks on results of integration tests to ensure quality criteria are met
* install - install the package into the local repository, for use as a dependency in other projects locally
* deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

These lifecycle phases (plus the other lifecycle phases not shown here) are executed sequentially to complete the default lifecycle. Given the lifecycle phases above, this means that when the default lifecycle is used, Maven will first validate the project, then will try to compile the sources, run those against the tests, package the binaries (e.g. jar), run integration tests against that package, verify the integration tests, install the verified package to the local repository, then deploy the installed package to a remote repository.

Site life cycle

This site life cycle is used to create a web-site for ur project , defined with plugin bindings, this I least used

Basic commands

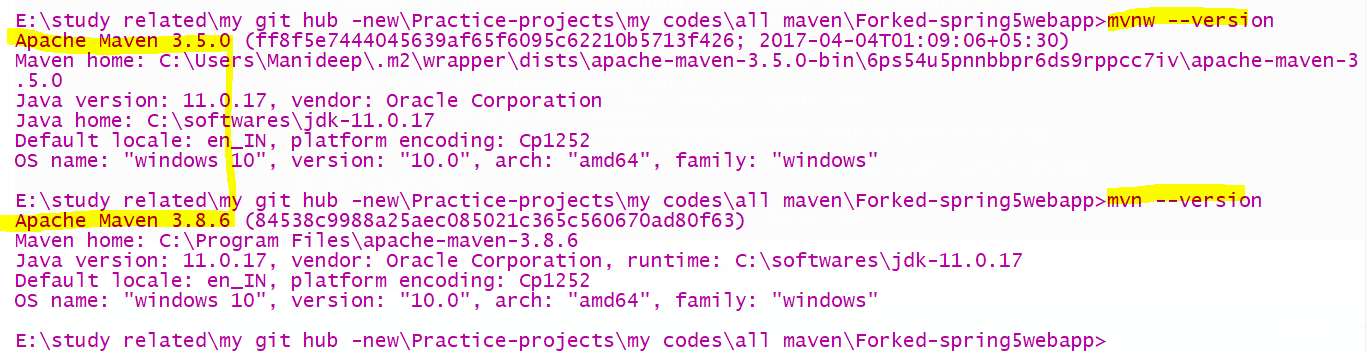
|  |  |
| --- | --- |
| To see the dependencies list | mvn dependency:tree |
|  | mvn dependency:go-offline |
|  | mvn dependency:purge-local-repository |
| To pull all the sources-jars | mvn dependency:sources |
|  |  |

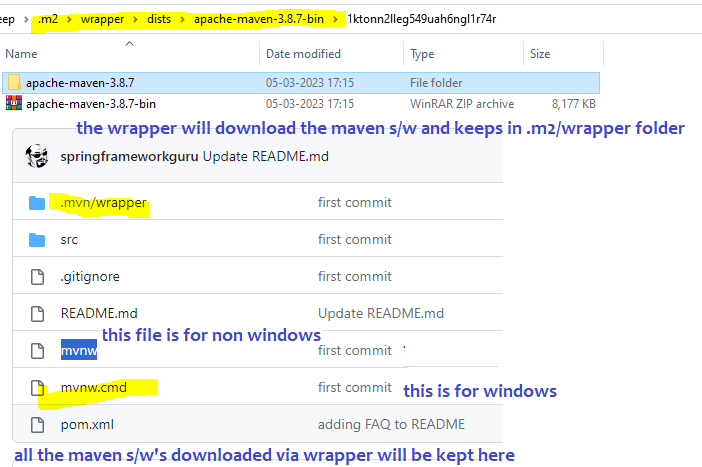
Maven wrapper

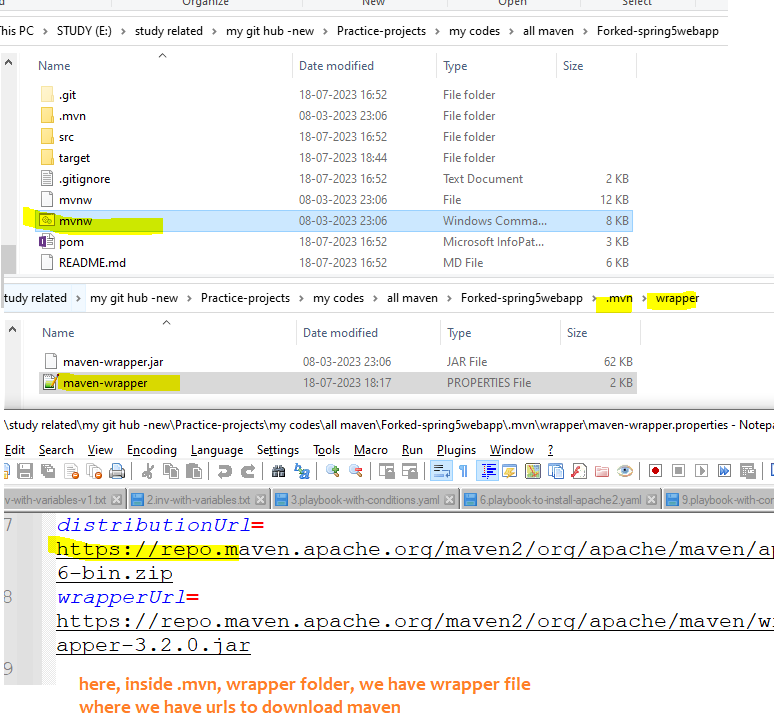
Maven wrapper is just a wrapper around maven software, this wrapper allows us to work with maven without having maven installed.

Because this wrapper itself will download and install maven

If u don’t have maven installed, then we can work with maven wrapper, which internally work with maven







Here if u see “mvnw” means it Is a tool called maven wrapper (just like javac), , this maven wrapper will download the original maven

When u download the project, then itself in the code itself u will get mvnw (maven wrapper ) file

Now U can do both mnv test as well as mvnw test

This wrapper is inspired from gradle wrapper,

Archetypes

He selected sitesimple archetype

## Excluding a dependency

1. <dependency>
2. <groupId>sample.ProjectA</groupId>
3. <artifactId>Project-A</artifactId>
4. <version>1.0</version>
5. <scope>compile</scope>
6. <exclusions>
7. <exclusion> <!-- declare the exclusion here -->
8. <groupId>sample.ProjectB</groupId>
9. <artifactId>Project-B</artifactId>
10. </exclusion>
11. </exclusions>
12. </dependency>

Plugins

1 plugin have many goals

1. clean plugin have goal called “clean”
2. default plugin have goals named “compile” , “testcompile””

|  |  |
| --- | --- |
| Sure fire plugin | To run unit test |
| Sure fire plugin with tags | <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-plugin</artifactId>  <version>2.22.0</version>  <configuration>  <argLine>  --illegal-access=permit  </argLine>  <!--<groups>model</groups>--> <!This means only tests which are tagged with model will be executed @Tag(“model”)>  <!--<excludedGroups>controllers</excludedGroups>-->  </configuration>  </plugin> |
| Fail safe plugin | To handle integration tests |
| Site plugin | When u run “site“ then a clean big report will be generated about the tests how many ignored and coverage |
|  |  |
|  |  |

Site plugin : source code to add this plugin is found at

<https://github.com/springframeworkguru/testing-java-junit5/blob/surefire-report/pom.xml>

To generate like below website u need to add

|  |  |
| --- | --- |
| <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-site-plugin</artifactId>  <version>3.7.1</version>  </plugin>  </plugins>  </build> | <reporting>  <plugins>  <plugin>  <groupId>org.apache.maven.plugins</groupId>  <artifactId>maven-surefire-report-plugin</artifactId>  <version>2.22.0</version>  </plugin>  </plugins>  </reporting>  </project> |

