20+ Maven Commands and Options (Cheat Sheet)

Published on August 3, 2022 · Updated on February 10, 2023

- Maven

By Pankaj

Introduction

Maven is one of the most popular project and dependency management tools for Java applications. Maven provides a lot of commands and options to help you in your day to day tasks.

This cheat sheet uses a sample Maven project to demonstrate some useful Maven commands. It was originally written for OpenJDK 13.0.1 and Maven 3.6.3. These commands have been verified with OpenJDK 19.0.1 and Maven 3.8.7.

Maven Commands Cheat Sheet

Maven Commands Cheat

Sheet

`mvn clean`

This command cleans the Maven project by deleting the target directory:

```
1. mvn clean
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-clean-plugin:2.5:clean (default-clean) @ maven-example-
jar ---

[INFO] Deleting /Users/sammy/Desktop/maven-examples/maven-example-
jar/target

[INFO] ------------------------------------------------------------------------
-----

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------
-----
```

`mvn compiler:compile`

This command compiles the Java source classes of the Maven project:

```
1. mvn compiler:compile
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-compiler-plugin:3.8.1:compile (default-cli) @ maven-
example-jar ---

[INFO] Changes detected - recompiling the module!
```

```
[INFO] Compiling 1 source file to /Users/sammy/Desktop/maven-
examples/maven-example-jar/target/classes

[INFO] ------------------------------------------------------------------------
-----

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------
-----
```

`mvn compiler:testCompile`

This command compiles the test classes of the Maven project:

```
1. mvn compiler:testCompile
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-cli) @ maven-
example-jar ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to /Users/sammy/Desktop/maven-
examples/maven-example-jar/target/test-classes

[INFO] ------------------------------------------------------------------------
-----

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------
-----
```

`mvn package`

This command builds the Maven project and packages it into a `JAR`, `WAR`, etc.:

```
1. mvn package
```

```
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ maven-
example-jar ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to /Users/sammy/Desktop/maven-
examples/maven-example-jar/target/classes

...

[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @
maven-example-jar ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 1 source file to /Users/sammy/Desktop/maven-
examples/maven-example-jar/target/test-classes

...

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-
example-jar ---

[INFO] Surefire report directory: /Users/sammy/Desktop/maven-
examples/maven-example-jar/target/surefire-reports



-------------------------------------------------------

 T E S T S

-------------------------------------------------------

Running com.example.maven.classes.AppTest
```

```
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.005 sec



Results :



Tests run: 1, Failures: 0, Errors: 0, Skipped: 0



[INFO]

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ maven-example-jar ---

[INFO] Building jar: /Users/sammy/Desktop/maven-examples/maven-example-
jar/target/maven-example-jar-0.0.1-SNAPSHOT.jar

[INFO] ------------------------------------------------------------------------
-----

[INFO] BUILD SUCCESS
```

The output shows the location of the `JAR` file just before the `"BUILD SUCCESS"` message. Notice the `package` goal executes `compile`, `testCompile`, and `test` goals before packaging the build.

`mvn install`

This command builds the Maven project and installs the project files (`JAR`, `WAR`, `pom.xml`, etc.) to the local repository:

```
1. mvn install
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @
maven-example-jar ---
```

```
...

[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ maven-example-jar ---

...

[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ maven-example-jar ---

...

[INFO] --- maven-compiler-plugin:3.8.1:testCompile (default-testCompile) @ maven-example-jar ---

...

[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ maven-example-jar ---

...

[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ maven-example-jar ---

...

[INFO] --- maven-install-plugin:2.4:install (default-install) @ maven-example-jar ---

[INFO] Installing /Users/sammy/Desktop/maven-examples/maven-example-jar/target/maven-example-jar-0.0.1-SNAPSHOT.jar to /Users/sammy/.m2/repository/com/example/maven/maven-example-jar/0.0.1-SNAPSHOT/maven-example-jar-0.0.1-SNAPSHOT.jar

[INFO] Installing /Users/sammy/Desktop/maven-examples/maven-example-jar/pom.xml to /Users/sammy/.m2/repository/com/example/maven/maven-example-jar/0.0.1-SNAPSHOT/maven-example-jar-0.0.1-SNAPSHOT.pom

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS
```

`mvn deploy`

This command deploys the artifact to the remote repository:

```
1. mvn deploy
2.
```

Copy

The remote repository should be configured properly in the project `pom.xml` file `distributionManagement` tag. The server entries in the Maven `settings.xml` file are used to provide authentication details.

`mvn validate`

This command validates the Maven project to ensure that everything is correct and all the necessary information is available:

```
1. mvn validate
2.
```

Copy

`mvn dependency:tree`

This command generates the dependency tree of the Maven project:

```
1. mvn dependency:tree
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ Mockito-
Examples ---

[INFO] com.example.mockito:Mockito-Examples:jar:1.0-SNAPSHOT

[INFO] +- org.junit.platform:junit-platform-runner:jar:1.2.0:test

[INFO] |  +- org.apiguardian:apiguardian-api:jar:1.0.0:test

[INFO] |  +- org.junit.platform:junit-platform-launcher:jar:1.2.0:test

[INFO] |  \- org.junit.platform:junit-platform-suite-api:jar:1.2.0:test
```

```
[INFO] |     \- org.junit.platform:junit-platform-commons:jar:1.2.0:test

[INFO] +- org.junit.jupiter:junit-jupiter-engine:jar:5.2.0:test

[INFO] |  +- org.junit.platform:junit-platform-engine:jar:1.2.0:test

[INFO] |  |  \- org.opentest4j:opentest4j:jar:1.1.0:test

[INFO] |  \- org.junit.jupiter:junit-jupiter-api:jar:5.2.0:test

[INFO] +- org.mockito:mockito-junit-jupiter:jar:2.19.0:test

[INFO] |  \- org.mockito:mockito-core:jar:2.19.0:test

[INFO] |     +- net.bytebuddy:byte-buddy:jar:1.8.10:test

[INFO] |     +- net.bytebuddy:byte-buddy-agent:jar:1.8.10:test

[INFO] |     \- org.objenesis:objenesis:jar:2.6:test

[INFO] \- org.testng:testng:jar:6.14.3:test

[INFO]    +- com.beust:jcommander:jar:1.72:test

[INFO]    \- org.apache-extras.beanshell:bsh:jar:2.0b6:test
```

`mvn dependency:analyze`

This command analyzes the maven project to identify the unused declared and used undeclared dependencies:

```
1. mvn dependency:analyze
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-dependency-plugin:2.8:analyze (default-cli) @ Mockito-
Examples ---

[WARNING] Used undeclared dependencies found:
```

```
[WARNING]    org.junit.jupiter:junit-jupiter-api:jar:5.2.0:test

[WARNING]    org.mockito:mockito-core:jar:2.19.0:test

[WARNING] Unused declared dependencies found:

[WARNING]    org.junit.platform:junit-platform-runner:jar:1.2.0:test

[WARNING]    org.junit.jupiter:junit-jupiter-engine:jar:5.2.0:test

[WARNING]    org.mockito:mockito-junit-jupiter:jar:2.19.0:test
```

It's useful in reducing the build size by identifying the unused dependencies and removing them from the `pom.xml` file.

`mvn archetype:generate`

This command generates skeleton Maven projects of different types, such as `JAR`, web application, Maven site, etc:

```
1. mvn archetype:generate
2.
```

Copy

Example of the output:

```
Output

[INFO] ------------------------------------------------------------------------
---------

[INFO] Using following parameters for creating project from Archetype:
maven-archetype-quickstart:1.4

[INFO] ------------------------------------------------------------------------
---------

...

[INFO] Project created from Archetype in dir: /Users/sammy/Desktop/maven-
examples/maven-example-jar

[INFO] ------------------------------------------------------------------------
-----
```

```
[INFO] BUILD SUCCESS
```

Recommended Reading: [Creating a Java Project using Maven Archetypes]

`mvn site:site`

This command generates a site for the project:

```
1. mvn site:site
2.
```

Copy

You will notice a `site` directory in the `target` directory after executing this command.

```
/Users/sammy/Desktop/maven-examples/maven-example-
jar/target/site/index.html
```

There will be multiple HTML files inside the `site` directory that provide information related to the project.

`mvn test`

This command runs the test cases of the project:

```
1. mvn test
2.
```

Copy

Example of the output:

```
Output

[INFO] -------------------------------------------------------

[INFO]  T E S T S

[INFO] -------------------------------------------------------

[INFO] Running TestSuite

first-element

second-element
```

```
Employee setName Argument = Sammy

...

[INFO] Results:

[INFO]

[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0

[INFO]

[INFO] ------------------------------------------------------------------------
-----

[INFO] BUILD SUCCESS
```

`mvn compile`

This command compiles the source Java classes of the project:

```
1. mvn compile
2.
```

Copy

Example of the output:

```
Output

[INFO] --- maven-compiler-plugin:3.7.0:compile (default-compile) @
Mockito-Examples ---

[INFO] Changes detected - recompiling the module!

[WARNING] File encoding has not been set, using platform encoding UTF-8,
i.e. build is platform dependent!

[INFO] Compiling 10 source files to /Users/sammy/Desktop/maven-
examples/Mockito-Examples/target/classes
```

It is similar to the previous `mvn compiler:compile` command, but runs the entire Maven lifecycle up to `compile`.

`mvn verify`

This command builds the project, runs all the test cases and run any checks on the results of the integration tests to ensure quality criteria are met:

```
1. mvn verify
2.
```

Copy

Maven Options

Maven provides a lot of command-line options to alter the Maven build process:

`mvn -help`

This command-line option prints the Maven usage and all the available options:

```
1. mvn -help
2.
```

Copy

Example of the output:

```
Output

usage: mvn [options] [<goal(s)>] [<phase(s)>]



Options:

  -am,--also-make                   If project list is specified, also

                                    build projects required by the

                                    list

  -amd,--also-make-dependents       If project list is specified, also

                                    build projects that depend on

                                    projects on the list

  -B,--batch-mode                   Run in non-interactive (batch)
```

```
                                mode (disables output color)
 -b,--builder <arg>             The id of the build strategy to
                                use
 -C,--strict-checksums          Fail the build if checksums don't
                                match
 -c,--lax-checksums             Warn if checksums don't match
 -cpu,--check-plugin-updates    Ineffective, only kept for
                                backward compatibility
```

`mvn -f dir/pom.xml package`

This command-line option builds a project from a different location:

```
1. mvn -f dir/pom.xml package
2.
```

Copy

Provide the `pom.xml` file location to build the project. It's useful when you have to run a Maven build from a script.

`mvn -o package`

This command-line option runs the Maven build in *offline mode*:

```
1. mvn -o package
2.
```

Copy

It's useful when you have all the required `JAR`s downloaded in the local repository and you don't want Maven to look for any `JAR`s in the remote repository.

`mvn -q package`

This command-line option runs the Maven build in *quiet mode*, so that only the test case results and errors are displayed:

```
1. mvn -q package
```

```
2.
```

Copy

`mvn -X package`

This command-line option prints the Maven version and runs the build in *debug mode*, so that all messages are displayed:

```
1. mvn -X package
2.
```

Copy

Example of the output:

```
Output

Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

Maven home: /Users/sammy/Downloads/apache-maven-3.6.3

Java version: 13.0.1, vendor: Oracle Corporation, runtime:
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home

Default locale: en_IN, platform encoding: UTF-8

OS name: "mac os x", version: "10.15.1", arch: "x86_64", family: "mac"

[DEBUG] Created new class realm maven.api

[DEBUG] Importing foreign pacakges into class realm maven.api

[DEBUG]   Imported: javax.annotation.* < plexus.core

[DEBUG]   Imported: javax.annotation.security.* < plexus.core

[DEBUG]   Imported: javax.enterprise.inject.* < plexus.core

[DEBUG]   Imported: javax.enterprise.util.* < plexus.core

[DEBUG]   Imported: javax.inject.* < plexus.core
```

`mvn -v`

This command-line option displays the Maven version information:

```
1. mvn -v
2.
```

Copy

Example of the output:

```
Output

Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)

Maven home: /Users/sammy/Downloads/apache-maven-3.6.3

Java version: 13.0.1, vendor: Oracle Corporation, runtime:
/Library/Java/JavaVirtualMachines/jdk-13.0.1.jdk/Contents/Home

Default locale: en_IN, platform encoding: UTF-8

OS name: "mac os x", version: "10.15.1", arch: "x86_64", family: "mac"
```

`mvn -V package`

This command-line option prints the Maven version and then continues with the build:

```
1. mvn -V package
2.
```

Copy

It's equivalent to the commands:

```
1. mvn -v;mvn package
2.
```

Copy

`mvn -DskipTests package`

This command-line option applies the `skipTests` system property to skip the unit test cases from the build cycle:

```
1. mvn -DskipTests package
2.
```

Copy

You can also skip the test cases execution:

```
1. mvn -Dmaven.test.skip=true package
2.
```

Copy

`mvn -T 4 clean install`

This command-line option tells Maven to run parallel builds using the specified thread count:

```
1. mvn -T 4 clean install
2.
```

Copy

It’s useful in multiple module projects where modules can be built in parallel. It can reduce the build time of the project.

References

- Maven Plugins
- Maven CLI Options Reference