Build Atuomation con Apache Maven

Table of Contents

Verificare l'installazione di Maven

- Aprire un'interfaccia a riga di comando ed eseguire il seguente comando

```
mvn --version
```

- comparirà un risultato simile a:

```
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: /home/bertazzo/Programmi/apache-maven-3
Java version: 11.0.14, vendor: Ubuntu, runtime: /usr/lib/jvm/java-11-openjdk-amd64
Default locale: it_IT, platform encoding: UTF-8
OS name: "linux", version: "5.13.0-37-generic", arch: "amd64", family: "unix
```

dove vengono riportati:

- la versione di Maven
- la versione di Java
- la variabile JAVA_HOME
- le caratteristiche del sistema operativo

Project Archetype - Universal reuse of business logic

- Aprire un'interfaccia a riga di comando ed eseguire

```
 mvn archetype:generate -DgroupId=com.mycompany.app -DartifactId=my-app
-DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4
-DinteractiveMode=false
```

WARNING	È richiesta una connessione ad internet

Il comando esegue il goal *generate* del plugin archetype che come descritto dalla documentazione:

- *creates a Maven project from an archetype: asks the user to choose an archetype from the archetype catalog, and retrieves it from the remote repository. Once retrieved, it is processed to create a working Maven project.*

I parametri specificati sono:

- **groupId**: permette di specificare il group id del progetto da creare
- **artifactId**: permette di specificare l'artifact id del progetto da creare
- **archetypeArtifactId**: permette di selezionare che archetype (template di progetto) utilizzare per la creazione del progetto

Se il plugin archetype non è presente nel repository locale, maven lo scarica automaticamente da un repository remoto. Se l'artefatto che definisce l'archetype non è presente nel repository locale, maven lo scarica automaticamente dal repository remoto.

Il risultato sarà un progetto maven con la seguente struttuara:

```
my-app/
├──── pom.xml
└──── src
    ├──── main
    │    └──── java
    │         └──── com
    │              └──── mycompany
    │                   └──── app
    │                        └──── App.java
    └──── test
         └──── java
              └──── com
                   └──── mycompany
                        └──── app
                             └──── AppTest.java
```

TIP	provare ad eseguire il comando senza specificare i parametri *archetypeArtifactId* e *interactiveMode*. Risultato: maven mostrerà il catalogo di tutti gli archetype che possono essere utilizzati e ci guiderà nella definizione del progetto

POM - Project Object Model

Il progetto creato ha un il file pom.xml. Il contenuto del file è il seguente:

```
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>

  <name>my-app</name>
  <!-- FIXME change it to the project's website -->
  <url>http://www.example.com</url>

  <properties>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
  </properties>

  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>

  <build>
    <pluginManagement><!-- lock down plugins versions to avoid using Maven defaults
(may be moved to parent pom) -->
      <plugins>
        <!-- clean lifecycle, see https://maven.apache.org/ref/current/maven-
core/lifecycles.html#clean_Lifecycle -->
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
```

```
        <!-- default lifecycle, jar packaging: see
https://maven.apache.org/ref/current/maven-core/default-
bindings.html#Plugin_bindings_for_jar_packaging -->
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
          <version>2.22.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-jar-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-install-plugin</artifactId>
          <version>2.5.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>2.8.2</version>
        </plugin>
        <!-- site lifecycle, see https://maven.apache.org/ref/current/maven-
core/lifecycles.html#site_Lifecycle -->
        <plugin>
          <artifactId>maven-site-plugin</artifactId>
          <version>3.7.1</version>
        </plugin>
        <plugin>
          <artifactId>maven-project-info-reports-plugin</artifactId>
          <version>3.0.0</version>
        </plugin>
      </plugins>
    </pluginManagement>
  </build>
</project>
```

- Notare che il progetto è stato creato con il GroupId, ArtifactId specificati, e la versione di default. Questi tre parametri sono conosciuti come GAV.
- Il GAV permette di identificare univocamente l'artefatto creato dall'esecuzione del processo di build all'interno del repository.
- Il parametro packaging permette di definire che l'artefatto prodotto dal processo di build è un jar (java archive)
- Nella sezione dependencies vengono specificate le dipendenze che vengono utilizzate dal

progetto

Build Lifecycle (default)

Posizionarsi nella cartella *my-app* Eseguire il seguente comando:

```
mvn install
```

Notare che vengono eseguite (quasi tutte) le fasi definite nel *Build Lifecycle default*

```
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mycompany.app:my-app >----------------------
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ my-app ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /tmp/my-app/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ my-app
---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory /tmp/my-app/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ my-app ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ my-app ---
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running com.mycompany.app.AppTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.129 s - in
com.mycompany.app.AppTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO]
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ my-app ---
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ my-app ---
[INFO] Installing /tmp/my-app/target/my-app-1.0-SNAPSHOT.jar to
/home/bertazzo/.m2/repository/com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-
SNAPSHOT.jar
[INFO] Installing /tmp/my-app/pom.xml to
/home/bertazzo/.m2/repository/com/mycompany/app/my-app/1.0-SNAPSHOT/my-app-1.0-
SNAPSHOT.pom
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  5.959 s
[INFO] Finished at: 2022-03-29T10:08:15+02:00
[INFO] ------------------------------------------------------------------------
```

WARNING	Se il processo di compilazione fallisce, consultare la documentazione del plugin Maven compiler

il comando genera la cartella target contenente:

```
target/
├─── classes
├─── maven-archiver
├─── maven-status
├─── my-app-1.0-SNAPSHOT.jar
├─── surefire-reports
└─── test-classes
```

tra i vari file troviamo:

- la cartella classes contenente i compilati dei sorgenti presenti nella cartella src/main, generati tramite l'esecuzione del goal *compile*
- la cartella test-classes contenente i compilati dei sorgenti presenti nella cartella src/test, generati tramite l'esecuzione del goal *test*
- la cartella surfire-reports conenente gli esiti dei test di unità
- l'artefatto (.jar) generato tramite il goal *package* che viene copiato nel repository locale dal goal *install*

WARNING	Questi sono file generati e non dovrebbero essere versionati nel VCS. Per maggiori dettagli vedi gitignore e gitignore.io.

Super POM

Come si può vedere Maven utilizza il principio di *convention over configuration* e utilizza delle proprietà che non sono specificate nel file pom.xml

Le configurazioni di default sono specificate nel super pom. Per vedere queste configurazioni è possibile eseguire il comando:

```
mvn help:effective-pom
```

Com'è possibile vedere dall'output generato, le proprietà definite di default sono molte e sono presenti anche tutte le configurazioni dei plugin utilizzati.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- ============================================================================ -->
<!--                                                                              -->
<!-- Generated by Maven Help Plugin on 2022-03-29T10:00:38+02:00                  -->
<!-- See: http://maven.apache.org/plugins/maven-help-plugin/                      -->
<!--                                                                              -->
```

```
<!-- ============================================================================ -->
<!-- ============================================================================ -->
<!--                                                                              -->
<!-- Effective POM for project 'com.mycompany.app:my-app:jar:1.0-SNAPSHOT'  -->
<!--                                                                              -->
<!-- ============================================================================ -->
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://www.example.com</url>
  <properties>
    <maven.compiler.source>1.7</maven.compiler.source>
    <maven.compiler.target>1.7</maven.compiler.target>
    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
  </properties>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <repositories>
    <repository>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>central</id>
      <name>Central Repository</name>
      <url>https://repo.maven.apache.org/maven2</url>
    </repository>
  </repositories>
  <pluginRepositories>
    <pluginRepository>
      <releases>
        <updatePolicy>never</updatePolicy>
      </releases>
      <snapshots>
        <enabled>false</enabled>
      </snapshots>
      <id>central</id>
      <name>Central Repository</name>
      <url>https://repo.maven.apache.org/maven2</url>
    </pluginRepository>
```

```
  </pluginRepositories>
  <build>
    <sourceDirectory>/tmp/my-app/src/main/java</sourceDirectory>
    <scriptSourceDirectory>/tmp/my-app/src/main/scripts</scriptSourceDirectory>
    <testSourceDirectory>/tmp/my-app/src/test/java</testSourceDirectory>
    <outputDirectory>/tmp/my-app/target/classes</outputDirectory>
    <testOutputDirectory>/tmp/my-app/target/test-classes</testOutputDirectory>
    <resources>
      <resource>
        <directory>/tmp/my-app/src/main/resources</directory>
      </resource>
    </resources>
    <testResources>
      <testResource>
        <directory>/tmp/my-app/src/test/resources</directory>
      </testResource>
    </testResources>
    <directory>/tmp/my-app/target</directory>
    <finalName>my-app-1.0-SNAPSHOT</finalName>
    <pluginManagement>
      <plugins>
        <plugin>
          <artifactId>maven-antrun-plugin</artifactId>
          <version>1.3</version>
        </plugin>
        <plugin>
          <artifactId>maven-assembly-plugin</artifactId>
          <version>2.2-beta-5</version>
        </plugin>
        <plugin>
          <artifactId>maven-dependency-plugin</artifactId>
          <version>2.8</version>
        </plugin>
        <plugin>
          <artifactId>maven-release-plugin</artifactId>
          <version>2.5.3</version>
        </plugin>
        <plugin>
          <artifactId>maven-clean-plugin</artifactId>
          <version>3.1.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-resources-plugin</artifactId>
          <version>3.0.2</version>
        </plugin>
        <plugin>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.0</version>
        </plugin>
        <plugin>
          <artifactId>maven-surefire-plugin</artifactId>
```

```
        <version>2.22.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-jar-plugin</artifactId>
        <version>3.0.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-install-plugin</artifactId>
        <version>2.5.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-deploy-plugin</artifactId>
        <version>2.8.2</version>
      </plugin>
      <plugin>
        <artifactId>maven-site-plugin</artifactId>
        <version>3.7.1</version>
      </plugin>
      <plugin>
        <artifactId>maven-project-info-reports-plugin</artifactId>
        <version>3.0.0</version>
      </plugin>
    </plugins>
  </pluginManagement>
  <plugins>
    <plugin>
      <artifactId>maven-clean-plugin</artifactId>
      <version>3.1.0</version>
      <executions>
        <execution>
          <id>default-clean</id>
          <phase>clean</phase>
          <goals>
            <goal>clean</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <artifactId>maven-resources-plugin</artifactId>
      <version>3.0.2</version>
      <executions>
        <execution>
          <id>default-testResources</id>
          <phase>process-test-resources</phase>
          <goals>
            <goal>testResources</goal>
          </goals>
        </execution>
        <execution>
          <id>default-resources</id>
```

```
        <phase>process-resources</phase>
        <goals>
          <goal>resources</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
  <plugin>
    <artifactId>maven-jar-plugin</artifactId>
    <version>3.0.2</version>
    <executions>
      <execution>
        <id>default-jar</id>
        <phase>package</phase>
        <goals>
          <goal>jar</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
  <plugin>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.8.0</version>
    <executions>
      <execution>
        <id>default-compile</id>
        <phase>compile</phase>
        <goals>
          <goal>compile</goal>
        </goals>
      </execution>
      <execution>
        <id>default-testCompile</id>
        <phase>test-compile</phase>
        <goals>
          <goal>testCompile</goal>
        </goals>
      </execution>
    </executions>
  </plugin>
  <plugin>
    <artifactId>maven-surefire-plugin</artifactId>
    <version>2.22.1</version>
    <executions>
      <execution>
        <id>default-test</id>
        <phase>test</phase>
        <goals>
          <goal>test</goal>
        </goals>
      </execution>
```

```
      </executions>
    </plugin>
    <plugin>
      <artifactId>maven-install-plugin</artifactId>
      <version>2.5.2</version>
      <executions>
        <execution>
          <id>default-install</id>
          <phase>install</phase>
          <goals>
            <goal>install</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <artifactId>maven-deploy-plugin</artifactId>
      <version>2.8.2</version>
      <executions>
        <execution>
          <id>default-deploy</id>
          <phase>deploy</phase>
          <goals>
            <goal>deploy</goal>
          </goals>
        </execution>
      </executions>
    </plugin>
    <plugin>
      <artifactId>maven-site-plugin</artifactId>
      <version>3.7.1</version>
      <executions>
        <execution>
          <id>default-site</id>
          <phase>site</phase>
          <goals>
            <goal>site</goal>
          </goals>
          <configuration>
            <outputDirectory>/tmp/my-app/target/site</outputDirectory>
            <reportPlugins>
              <reportPlugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-project-info-reports-plugin</artifactId>
              </reportPlugin>
            </reportPlugins>
          </configuration>
        </execution>
        <execution>
          <id>default-deploy</id>
          <phase>site-deploy</phase>
```

```
            <goals>
              <goal>deploy</goal>
            </goals>
            <configuration>
              <outputDirectory>/tmp/my-app/target/site</outputDirectory>
              <reportPlugins>
                <reportPlugin>
                  <groupId>org.apache.maven.plugins</groupId>
                  <artifactId>maven-project-info-reports-plugin</artifactId>
                </reportPlugin>
              </reportPlugins>
            </configuration>
          </execution>
        </executions>
        <configuration>
          <outputDirectory>/tmp/my-app/target/site</outputDirectory>
          <reportPlugins>
            <reportPlugin>
              <groupId>org.apache.maven.plugins</groupId>
              <artifactId>maven-project-info-reports-plugin</artifactId>
            </reportPlugin>
          </reportPlugins>
        </configuration>
      </plugin>
    </plugins>
  </build>
  <reporting>
    <outputDirectory>/tmp/my-app/target/site</outputDirectory>
  </reporting>
</project>
```

TIP esistono altri 2 *Build Lifecycle*: clean, site. È possibile eseguirli tramite il comando `mvn site` e `mvn clean`

Plugin - Universal reuse of business logic

Tutte le configurazione definite nel pom.xml ed ereditate dal ta possono essere utilizzate dai plugin.

Per questo se abbiamo l'esigenza di eseguire un plugin, è sufficiente eseguirlo e questo funzionerà leggendo le configurazioni dal pom.xml (o utilizzando le configurazioni di default).

Supponiamo di voler eseguire l'analisi statica del codice al progetto appena creato con il plugin checkstyle.

Dalla documentazione si può vedere che il plugin permette di eseguire i seguenti comandi:

- **checkstyle:checkstyle** is a reporting goal that performs Checkstyle analysis and generates a report on violations.
- **checkstyle:checkstyle-aggregate** is a reporting goal that performs Checkstyle analysis and

generates an aggregate HTML report on violations in a multi-module reactor build.

- **checkstyle:check** is a goal that performs Checkstyle analysis and outputs violations or a count of violations to the console, potentially failing the build. It can also be configured to re-use an earlier analysis.

Proviamo ad eseguire il goal *check*

```
mvn checkstyle:check
```

Il comando eseguirà l'analisi statica del progetto utilizzando la configurazione di default:

```
[INFO] There are 11 errors reported by Checkstyle 8.29 with sun_checks.xml ruleset.
```

Plugin e Build Lifecycle

Come ripotato nella documentazione del plugin, andiamo a configurare il plugin in modo da controllare che:

- in tutti i file ci sia una riga con scritto `// prova`
- creare un file **checkstyle.xml** con il seguente contenuto

```
<?xml version="1.0"?>
<!DOCTYPE module PUBLIC "-//Puppy Crawl//DTD Check Configuration 1.2//EN"
"http://www.puppycrawl.com/dtds/configuration_1_2.dtd">
<module name="Checker">
    <module name="RegexpHeader">
    <property name="headerFile" value="${checkstyle.header.file}"/>
  </module>
</module>
```

TIP per maggiori dettagli vedi: http://checkstyle.sourceforge.net/config_header.html

- Aggiungre un file LICENSE.txt con il seguente contenuto:

```
// prova
```

- modificare il file pom.xml aggiungendo le seguenti configurazioni:

```
...
    <build>
...
        </pluginManagement>
        <plugins>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-checkstyle-plugin</artifactId>
                <version>3.1.2</version>
                <configuration>
                    <failsOnError>true</failsOnError>
                    <configLocation>checkstyle.xml</configLocation>
                    <consoleOutput>true</consoleOutput>
                </configuration>
                <executions>
                    <execution>
                        <phase>verify</phase>
                        <goals>
                            <goal>checkstyle</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
</project>
```

- Nel tag build è stata specificata la configurazione del plugin *maven-checkstyle-plugin*
- Nella parte *executions* è stato specificato di eseguire il goal *checkstyle* nella della fase **verify**
- Provare ad eseguire il plugin eseguendo il seguente comando:

```
mvn verify
```

- Notare che è stato eseguito il plugin:

```
[INFO] --- maven-checkstyle-plugin:3.1.2:checkstyle (default) @ my-app ---
```

- Verrà segnalato il seguente errore:

```
[ERROR] /tmp/my-app/src/main/java/com/mycompany/app/App.java:1: Line does not match
expected header line of '// prova'. [RegexpHeader]
```

Dipendenze

Proviamo ad aggiungere una dipendenza al progetto.

Supponiamo di voler utilizzare il framework selenium. Come descritto nel sito di progetto è richiesta la dipendenza:

```
<dependency>
    <groupId>org.seleniumhq.selenium</groupId>
    <artifactId>selenium-java</artifactId>
    <version>3.13.0</version>
</dependency>
```

La libreria è identificata da un GroupID, un ArtifactID e una versione.

Modifichiamo il file pom.xml e aggiungiamo la dipendenza:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1.0-SNAPSHOT</version>

    <name>my-app</name>
    <!-- FIXME change it to the project's website -->
    <url>http://www.example.com</url>

    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <maven.compiler.source>1.7</maven.compiler.source>
        <maven.compiler.target>1.7</maven.compiler.target>
    </properties>

  <dependencies>
      <dependency>
          <groupId>junit</groupId>
          <artifactId>junit</artifactId>
          <version>4.11</version>
          <scope>test</scope>
      </dependency>
      <dependency>
          <groupId>org.seleniumhq.selenium</groupId>
          <artifactId>selenium-java</artifactId>
          <version>3.13.0</version>
      </dependency>
  </dependencies>
....
</project>
```

eseguendo il seguente comando, verrà scaricata la dipendenza e tutte le dipendenze transitive richieste:

```
mvn compile
```

eseguendi il comando:

```
mvn dependency:tree
```

È possibile vedere quali dipendenze sono utilizzate dal progetto:

```
[INFO] --- maven-dependency-plugin:2.8:tree (default-cli) @ my-app ---
[INFO] com.mycompany.app:my-app:jar:1.0-SNAPSHOT
[INFO] +- junit:junit:jar:4.11:test
[INFO] |  \- org.hamcrest:hamcrest-core:jar:1.3:test
[INFO] \- org.seleniumhq.selenium:selenium-java:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-api:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-chrome-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-edge-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-firefox-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-ie-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-opera-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-remote-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-safari-driver:jar:3.13.0:compile
[INFO]    +- org.seleniumhq.selenium:selenium-support:jar:3.13.0:compile
[INFO]    +- net.bytebuddy:byte-buddy:jar:1.8.3:compile
[INFO]    +- org.apache.commons:commons-exec:jar:1.3:compile
[INFO]    +- commons-codec:commons-codec:jar:1.10:compile
[INFO]    +- commons-logging:commons-logging:jar:1.2:compile
[INFO]    +- com.google.code.gson:gson:jar:2.8.4:compile
[INFO]    +- com.google.guava:guava:jar:25.0-jre:compile
[INFO]    |  +- com.google.code.findbugs:jsr305:jar:1.3.9:compile
[INFO]    |  +- org.checkerframework:checker-compat-qual:jar:2.0.0:compile
[INFO]    |  +- com.google.errorprone:error_prone_annotations:jar:2.1.3:compile
[INFO]    |  +- com.google.j2objc:j2objc-annotations:jar:1.1:compile
[INFO]    |  \- org.codehaus.mojo:animal-sniffer-annotations:jar:1.14:compile
[INFO]    +- org.apache.httpcomponents:httpclient:jar:4.5.5:compile
[INFO]    +- org.apache.httpcomponents:httpcore:jar:4.4.9:compile
[INFO]    +- com.squareup.okhttp3:okhttp:jar:3.10.0:compile
[INFO]    \- com.squareup.okio:okio:jar:1.14.1:compile
```

Se il progetto viene imporato in un IDE, sarà possibile vedere che il classpath del progetto contiene tutte le dipendenze richieste dal progetto e dalle dipendenze (dipendenze transitive).

Il progetto creato è un artefatto Maven. Può essere a sua volta specificato come dipendenza da altri progetti Maven.