

# Build Automation con Apache Maven

## Table of Contents

Verificare l'installazione di Maven .....	1
Project Archetype - Universal reuse of business logic .....	2
POM - Project Object Model .....	3
Build Lifecycle (default) .....	3
Super POM .....	5
Plugin - Universal reuse of business logic .....	11
Plugin e Build Lifecycle .....	11
Dipendenze .....	14

## Verificare l'installazione di Maven

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

```
mvn --version
```

- comparirà un risultato simile a:

```
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=2048M; support
was removed in 8.0
Apache Maven 3.3.9 (bb52d8502b132ec0a5a3f4c09453c07478323dc5; 2015-11-
10T17:41:47+01:00)
Maven home: /home/bertazzo/Programmi/apache-maven-3
Java version: 1.8.0_131, vendor: Oracle Corporation
Java home: /usr/lib/jvm/java-8-oracle/jre
Default locale: it_IT, platform encoding: UTF-8
OS name: "linux", version: "4.15.0-36-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 -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 struttura:

```
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 file pom.xml. Il contenuto del file è il seguente:

```
<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/maven-
v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</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](#)

```
Java HotSpot(TM) 64-Bit Server VM warning: ignoring option MaxPermSize=2048M; support
was removed in 8.0
[INFO] Scanning for projects...
[INFO]
[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO]
[INFO] --- maven-resources-plugin:2.6:resources (default-resources) @ my-app ---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory /home/bertazzo/my-app/src/main/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:compile (default-compile) @ my-app ---
[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 1 source file to /home/bertazzo/my-app/target/classes
[INFO]
[INFO] --- maven-resources-plugin:2.6:testResources (default-testResources) @ my-app
---
[WARNING] Using platform encoding (UTF-8 actually) to copy filtered resources, i.e.
build is platform dependent!
[INFO] skip non existing resourceDirectory /home/bertazzo/my-app/src/test/resources
[INFO]
[INFO] --- maven-compiler-plugin:3.1:testCompile (default-testCompile) @ my-app ---
[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 1 source file to /home/bertazzo/my-app/target/test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.12.4:test (default-test) @ my-app ---
[INFO] Surefire report directory: /home/bertazzo/my-app/target/surefire-reports

-----
T E S T S
-----

Running com.mycompany.app.AppTest
Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.008 sec

Results :

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

[INFO]
[INFO] --- maven-jar-plugin:2.4:jar (default-jar) @ my-app ---
[INFO] Building jar: /home/bertazzo/my-app/target/my-app-1.0-SNAPSHOT.jar
[INFO]
```

```
[INFO] --- maven-install-plugin:2.4:install (default-install) @ my-app ---
[INFO] Installing /home/bertazzo/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 /home/bertazzo/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: 2.263 s
[INFO] Finished at: 2018-10-23T22:15:58+02:00
[INFO] Final Memory: 16M/209M
[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 surefire-reports contenente 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 2018-10-23T22:12:35+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
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>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</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>/home/bertazzo/my-app/src/main/java</sourceDirectory>
  <scriptSourceDirectory>/home/bertazzo/my-
app/src/main/scripts</scriptSourceDirectory>
  <testSourceDirectory>/home/bertazzo/my-app/src/test/java</testSourceDirectory>
  <outputDirectory>/home/bertazzo/my-app/target/classes</outputDirectory>
  <testOutputDirectory>/home/bertazzo/my-app/target/test-
classes</testOutputDirectory>
  <resources>
    <resource>
      <directory>/home/bertazzo/my-app/src/main/resources</directory>
    </resource>
  </resources>
  <testResources>
    <testResource>
      <directory>/home/bertazzo/my-app/src/test/resources</directory>
    </testResource>
  </testResources>
  <directory>/home/bertazzo/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>2.3.2</version>
      </plugin>
    </plugins>
  </pluginManagement>
</build>

```

```

</pluginManagement>
<plugins>
  <plugin>
    <artifactId>maven-clean-plugin</artifactId>
    <version>2.5</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>2.6</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>2.4</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>3.1</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>2.12.4</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>2.4</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>2.7</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>3.3</version>
    <executions>
        <execution>
            <id>default-site</id>
            <phase>site</phase>
            <goals>
                <goal>site</goal>
            </goals>
            <configuration>
                <outputDirectory>/home/bertazzo/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>/home/bertazzo/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>/home/bertazzo/my-app/target/site</outputDirectory>
        <reportPlugins>
            <reportPlugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-project-info-reports-plugin</artifactId>
            </reportPlugin>
        </reportPlugins>
    </configuration>
</plugin>

```

```
</plugin>
</plugins>
</build>
<reporting>
  <outputDirectory>/home/bertazzo/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 configurazioni definite nel `pom.xml` ed ereditate dal `parent` 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: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 6.18 with sun_checks.xml ruleset.
```

## Plugin e Build Lifecycle

Come riportato nella [documentazione del plugin](#), andiamo a configurare il plugin in modo da controllare che:

- in tutti i file **.md** ci sia una riga con scritto **Tecnologie open-source**
- 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="Header">
    <property name="header" value="//Tecnologie Open-Source" />
    <property name="fileExtensions" value="java" />
  </module>
</module>
```

**TIP** per maggiori dettagli vedi: [http://checkstyle.sourceforge.net/config\\_header.html](http://checkstyle.sourceforge.net/config_header.html)

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

```

<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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-checkstyle-plugin</artifactId>
        <version>2.17</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:2.17:checkstyle (default) @ my-app ---  
[WARNING] File encoding has not been set, using platform encoding UTF-8, i.e. build is  
platform dependent!  
[INFO] Starting audit...  
Audit done.  
[WARNING] Unable to locate Source XRef to link to - DISABLED
```

## 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/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>my-app</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</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
```

eseguendo il comando:

```
mvn dependency:tree
```

È possibile vedere quali dipendenze sono utilizzate dal progetto:

```

[INFO] -----
[INFO] Building my-app 1.0-SNAPSHOT
[INFO] -----
[INFO]
[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:3.8.1: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 importato 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.