

# Maven

Dipendenze, Configurazioni e Build

# Dipendenze



# Cos'è una dipendenza e come usarla

In relazione al nostro progetto una dipendenza è una libreria o framework, in poche parole altro codice scritto da qualcun'altro che ci serve per risolvere un problema.

Esempio: <https://projectlombok.org/>

# Configurazione e Build



# Come serve per generare un eseguibile java

Java è un linguaggio compilato quindi è necessario usare un programma di compilazione del codice.

L'installazione di java include anche il compilatore chiamato javac.

Una volta compilato è necessario impacchettare le classi in un eseguibile java, che può essere di diverso tipo: jar, war, ear, etc.etc.

Per eseguire una applicazione java è necessario avere la jvm installata nel proprio sistema.

# Passi per generare un eseguibile

- Compilazione classi
  - `javac <lista classi java> -cp <classpath>`
- Scrittura Manifest
  - Creazione file META-INF/MANIFEST.MF
- Impacchettamento
  - `jar -cvfm <nome eseguibile> META-INF/MANIFEST.MF <lista classi class>`
- Esecuzione
  - `java -jar <nome eseguibile>`

Cosa si fa se modifichiamo qualcosa del codice? Se modifico la versione di java?

Ripetere più volte le stessi processi => metodo alternativo: genero uno script che mi automatizza la generazione del jar, ma se aggiungo una nuova dipendenza?

# Maven

# Obiettivi

- Rendere il processo di build facile
  - I processi di compilazione e impacchettamento sono oscurati allo sviluppatore
- Creare un sistema uniforme per le build
  - Usando maven i progetti hanno una struttura specifica e le build sono gestite dal project object model (POM)
- Rendere più diretta la comunicazione delle informazioni del progetto
  - Nel POM sono contenute tutte le informazioni relative al progetto
- Incoraggiare ad usare le giuste pratiche di sviluppo
  - Aiuta a gestire il workflow dei rilasci e alla risoluzione dei problemi
  - Alcune automazioni forzano l'uso delle convenzioni per poter funzionare
  - Suggerisce la giusta configurazione della struttura dei progetti
  - ...



# Installazione

Download: <https://maven.apache.org/download.cgi>

Installazione: <https://maven.apache.org/install.html>

Su linux installazione è semplificata:

- apri terminale e usa il package manager della distribuzione per installarlo:
  - debian/ubuntu: `sudo apt install maven`
  - arch: `sudo pacman -S maven`

# Creazione progetto

Dalla documentazione ufficiale si può vedere quanto sia semplice usare maven, forse una delle parti più difficili (soprattutto da ricordare) è il comando per creare un nuovo progetto.

Per convenzione un progetto deve avere queste informazioni:

- Il gruppo che sta sviluppando il progetto (es: com.mycompany.app)
- Il nome del progetto, chiamato artefatto (es: my-app)
- Una tipologia

Prima di creare un progetto servono queste informazioni.

# Comando di creazione progetto

Apriamo un terminale e creiamo il progetto:

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

Eseguiamo e aspettiamo che tutte le dipendenze vengano scaricate.

# Lavorare sul nuovo progetto

Importiamo il progetto su eclipse e vediamo la sua struttura.

```
Maven-Start/  
├─ pom.xml  
└─ src  
    ├─ main  
    │   └─ java  
    │       └─ com  
    │           └─ plansoft  
    │               └─ mavenstart  
    │                   └─ App.java  
    └─ test  
        └─ java  
            └─ com  
                └─ plansoft  
                    └─ mavenstart  
                        └─ AppTest.java
```

# Lifecycle di maven

Ogni esecuzione di maven corrisponde all'esecuzione di un goal.

Quello che abbiamo eseguito noi è il goal chiamato archetype:generate.

Un goal è associato ad una fase specifica, chiamato anche step, del build lifecycle di maven.

Esistono 3 lifecycles:

- Clean
- Default
- Site

[Per vedere nello specifico andare nel sito maven nella sezione lifecycle.](#)

# Goals e Steps del lifecycle

Come già detto ogni goal che possiamo usare è associato a uno step specifico del lifecycle.

Ci sono alcuni step che hanno di default associato un goal, ma è possibile installare plugin che sostituiscano i goals di default.

[Per vedere nello specifico andare nel sito di maven seguendo la sezione built-in lifecycle bindings.](#)

# Il Project Object Model

Cos'è? È un file XML in cui sono contenute tutte le informazioni e le configurazioni relative al progetto necessarie a maven per poter eseguire build.

Com'è formato:

Pom Root

Versione del SuperPom

L'identificativo del del gruppo che  
sta lavorando al progetto


L'identificativo del progetto

La versione del progetto

```
<project>
  <modelVersion>4.0.0</modelVersion>

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

# Variabili



```
<project>
  ...
  <properties>
    <nomeVariabile>contenuto</nomeVariabile>
  </properties>
  ...
</project>
```

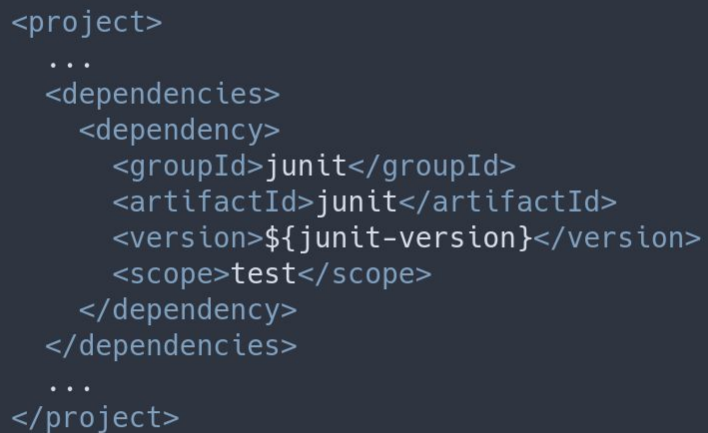


# Ereditarietà



```
<project>
  ...
  <parent>
    <groupId>com.mycompany.app</groupId>
    <artifactId>my-app</artifactId>
    <version>1</version>
  </parent>
  ...
</project>
```

# Dipendenze



```
<project>
  ...
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>${junit-version}</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  ...
</project>
```

# Maven Central / Maven Search

Search 

[Advanced Options](#) | [Classic Search](#) 

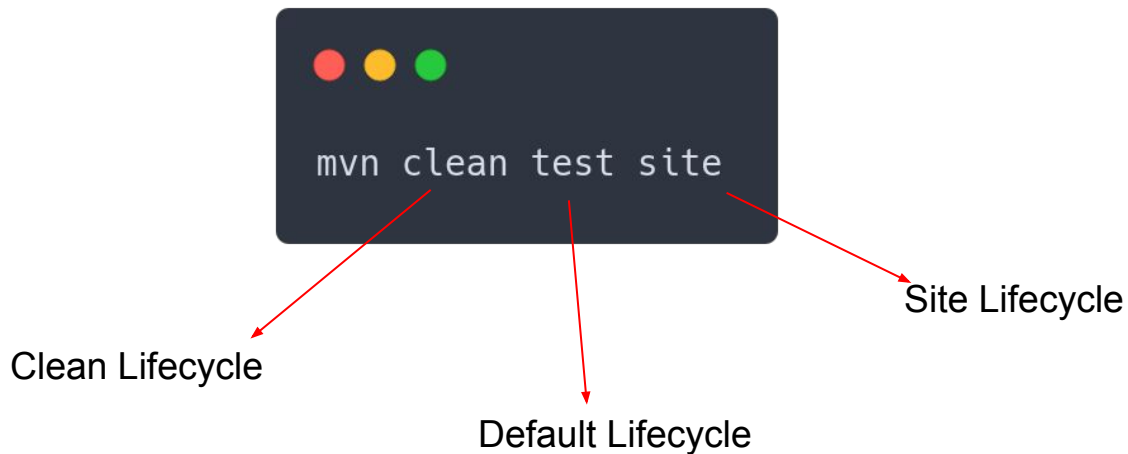
*Official search by the maintainers of Maven Central Repository*



<https://search.maven.org/>

# Eseguire una build

Per build si intende l'esecuzione di un goal di uno o più lifecycle:



# Creare eseguibile con maven

Modificare la sezione del plugin manager relativa al plugin di maven per la generazione dei jar:

```
<project>
...
  <configuration>
    <archive>
      <manifest>
        <addClasspath>true</addClasspath>
        <classpathPrefix>libs</classpathPrefix>
        <mainClass>
          com.mycompany.app.App
        </mainClass>
      </manifest>
    </archive>
  </configuration>
...
</project>
```