

Maven

Java Foundation

Presented by

Valerio Cammarota

IBM Client Innovation Center - Italy

16/11/2023

IBM Client Innovation Center
Italy

Agenda



Che cosa è Maven?



Componenti principali



POM.xml



Installazione Maven



Creazione progetto



Operazioni base

Che cosa è Maven?

- ❖ Maven è un progetto **open source**, sviluppato dalla Apache, che permette di **organizzare** in modo efficiente un progetto Java;
- ❖ Alcuni dei principali vantaggi di Maven:
 - Standardizzazione della struttura di un progetto;
 - Compilazione, test ed esportazione automatizzati;
 - Gestione e download automatico delle librerie necessarie al progetto;
- ❖ **Archetipo**: *«Il termine **archetipo** viene dal latino antico archetȳpum, a sua volta derivato dal greco antico ἀρχέτυπος, composto da **arché**, cioè «inizio, principio originario» e **typos**, «modello, marchio, esemplare». Dunque, nel suo significato originale, un archetipo è un primo modello, una prima forma, la matrice di un concetto, di un testo o di un'icona.»* cit. Wikipedia
 - In Maven l'archetipo (**archetype**) è il modello originale (**template**) a partire dal quale viene costruito il nostro progetto con struttura standard.

Componenti principali

- ❖ **pom.xml**: Un Project Object Model o POM è l'unità di lavoro fondamentale in Maven. È un file XML che contiene informazioni sul progetto, comprese le informazioni sulle dipendenze rispetto ad altre librerie, e dettagli di configurazione utilizzati da Maven per costruire il progetto
- ❖ **Goal**: singola funzione che può essere eseguita sul progetto
- ❖ **Jelly script**: linguaggio XML con il quale vengono definiti i goal
- ❖ **Plug-in**: mette a disposizione dei goal specifici
- ❖ **Repository**: directory strutturata destinata alla gestione delle librerie. Un repository può essere locale o remoto

Componenti principali: **pom.xml**

<project>

```
<modelVersion>4.0.0
</modelVersion>

<groupId>com.mycompany.app
</groupId>

<artifactId>my-app
</artifactId>

<version>1.0-SNAPSHOT
</version>

<name>my-app</name>
[...]
```

- ❖ **project**: è l'elemento di primo livello in tutti i file pom.xml
- ❖ **modelVersion**: indica quale versione del modello a oggetti si sta utilizzando
- ❖ **groupId**: indica l'identificatore univoco dell'organizzazione o del gruppo che ha creato il progetto e si basa sul dominio completo dell'organizzazione
- ❖ **artifactId**: indica il nome di base univoco dell'artefatto primario generato dal progetto, l'artefatto principale per un progetto è in genere un file JAR
- ❖ **version**: indica la versione dell'artefatto generato dal progetto
- ❖ **name**: indica il nome visualizzato utilizzato per il progetto

Componenti principali: **pom.xml**

[...]

```
<url>www.example.com</url>
```

```
<properties>[...]
```

```
</properties>
```

```
<dependencies>
```

```
    <dependency>
```

```
        [...]
```

```
    </dependency>
```

```
</dependencies>
```

```
<build>
```

```
    [...]
```

```
</build>
```

```
</project>
```

- ❖ **url**: indica dove si trova il sito del progetto ed è spesso usato nella documentazione generata da Maven
- ❖ **properties**: contiene i valori dei placeholder accessibili ovunque all'interno di un POM
- ❖ **dependencies**: i figli di questo elemento elencano le dipendenze
- ❖ **build**: gestisce cose come la dichiarazione della struttura della directory del tuo progetto e la gestione dei plugin

Come funziona: esempio



Installazione Maven: Windows

❖ Procediamo al download di Apache Maven 3.8.7:

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

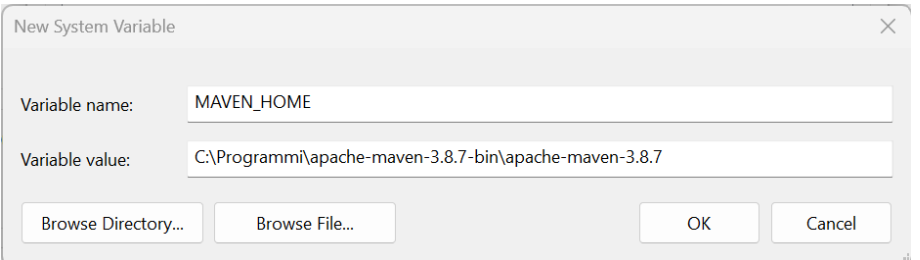
❖ Scompattiamo l'archivio

❖ Aggiungiamo la variabile

di sistema:

- **Name:** MAVEN_HOME
- **Value:** C:\{{path}}\apache-maven-3.8.7

	Link
Binary tar.gz archive	apache-maven-3.8.7-bin.tar.gz
Binary zip archive	apache-maven-3.8.7-bin.zip
Source tar.gz archive	apache-maven-3.8.7-src.tar.gz
Source zip archive	apache-maven-3.8.7-src.zip



New System Variable

Variable name: MAVEN_HOME

Variable value: C:\Programmi\apache-maven-3.8.7-bin\apache-maven-3.8.7

Browse Directory... Browse File... OK Cancel

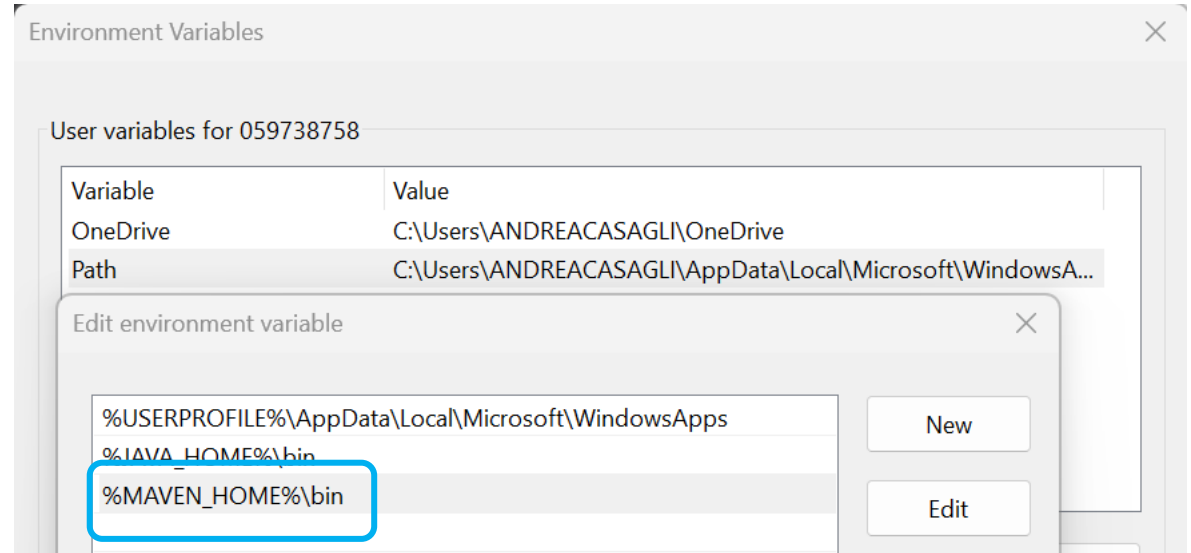
Installazione Maven: Windows

❖ Aggiungiamo nella variabile

d'ambiente Path `%MAVEN_HOME%\bin`

❖ Per verificare la corretta installazione
avviamo CMD ed eseguiamo il comando

```
mvn -version
```



```
Microsoft Windows [Version 10.0.22621.1105]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ANDREACASAGLI>mvn -version
Apache Maven 3.8.7 (b89d5959fcde851dcb1c8946a785a163f14e1e29)
Maven home: C:\Programmi\apache-maven-3.8.7-bin\apache-maven-3.8.7
Java version: 19.0.1, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk-19
Default locale: en_US, platform encoding: UTF-8
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"

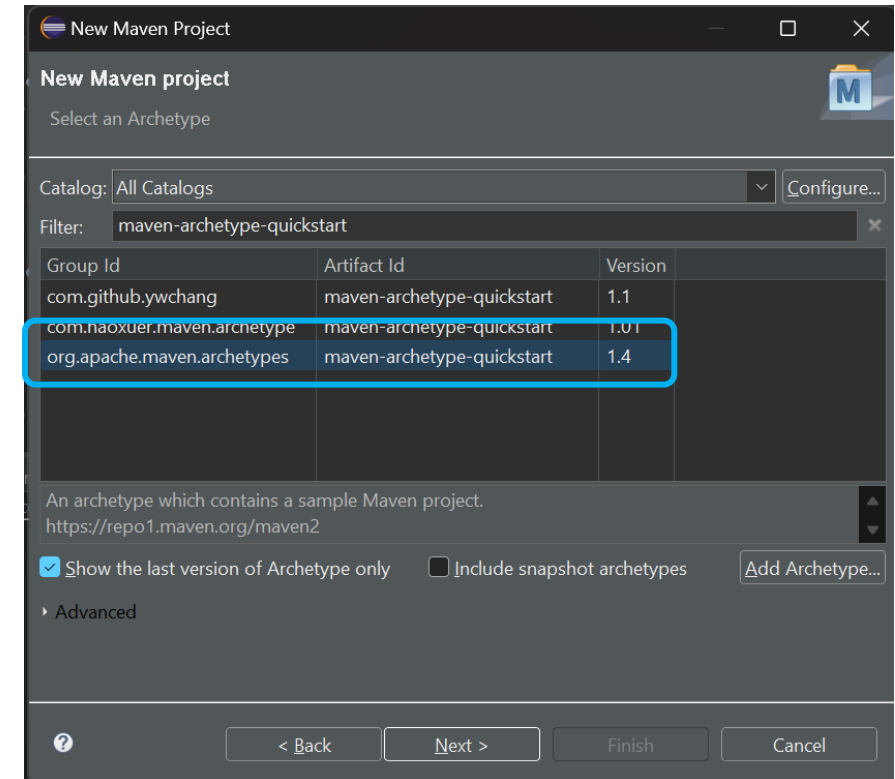
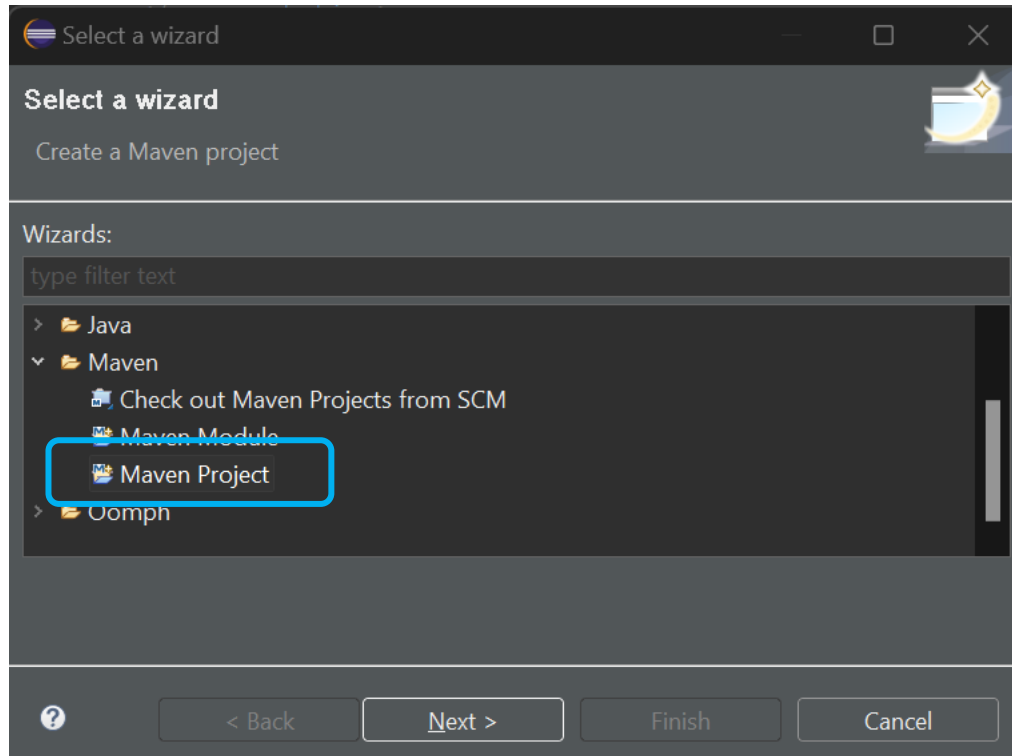
C:\Users\ANDREACASAGLI>
```

Installazione Maven: macOS

- ❖ Installare, se non è già presente, Homebrew seguendo la seguente guida: <https://brew.sh/>
- ❖ Usare poi il seguente comando per installare Maven: `brew install maven`
- ❖ Verificare il tipo di shell utilizzata con il seguente comando:
 - `echo $SHELL`
- ❖ Impostare la variabile d'ambiente `MAVEN_HOME` con uno dei seguenti comandi (in base al tipo di shell) e riavviare il terminale:
 - `echo 'export MAVEN_HOME=/usr/local/Cellar/maven/3.8.7' >> ~/.zshenv`
 - `echo 'export MAVEN_HOME=/usr/local/Cellar/maven/3.8.7' >> ~/.zshrc`
 - `echo 'export MAVEN_HOME=/usr/local/Cellar/maven/3.8.7' >> ~/.bash_profile`
 - `echo 'export MAVEN_HOME=/usr/local/Cellar/maven/3.8.7' >> ~/.bashrc`
- ❖ Infine, impostare la variabile `PATH` usando il seguente comando (in base al tipo di shell usato):
 - ❖ `echo 'export PATH=$MAVEN_HOME/bin:$PATH' >> ~/.<tipo_shell>`
- ❖ Infine testare lanciando il comando: `mvn -v`

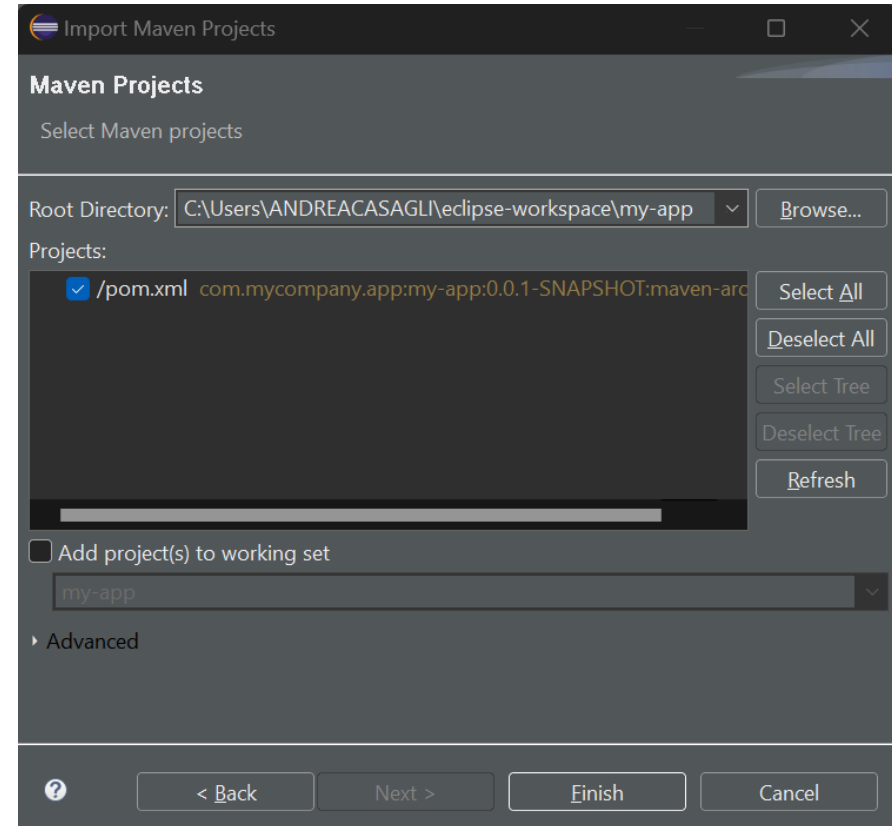
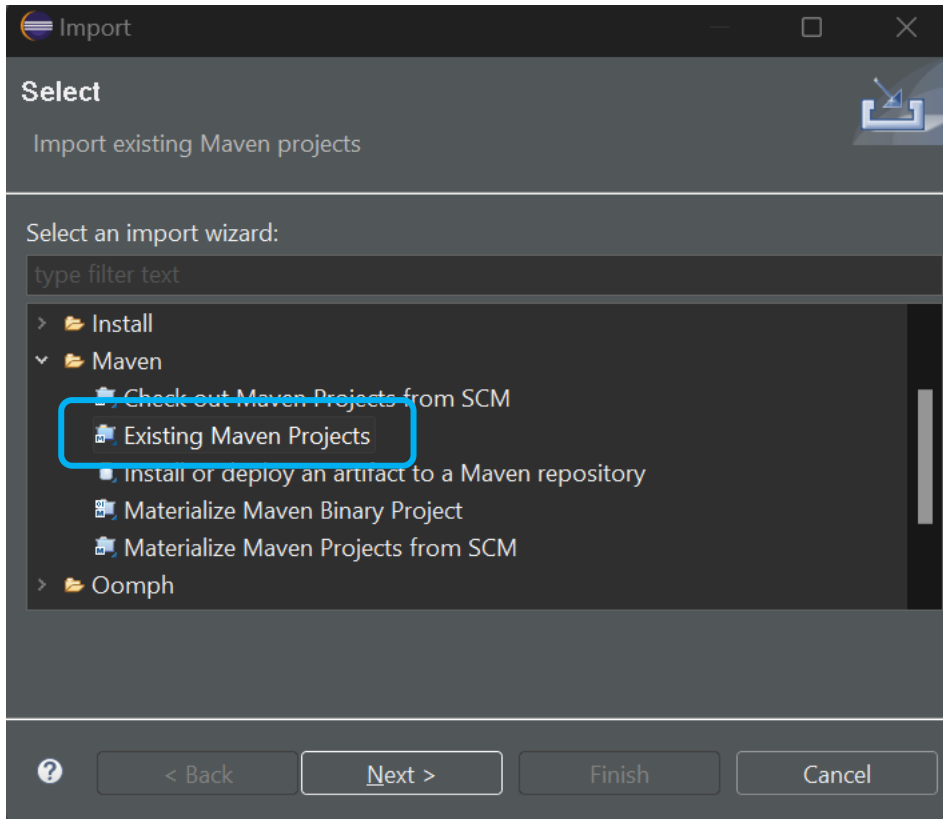
Creazione progetto

❖ Creiamo il primo progetto maven utilizzando l'archetipo base **maven-archetype-quickstart**

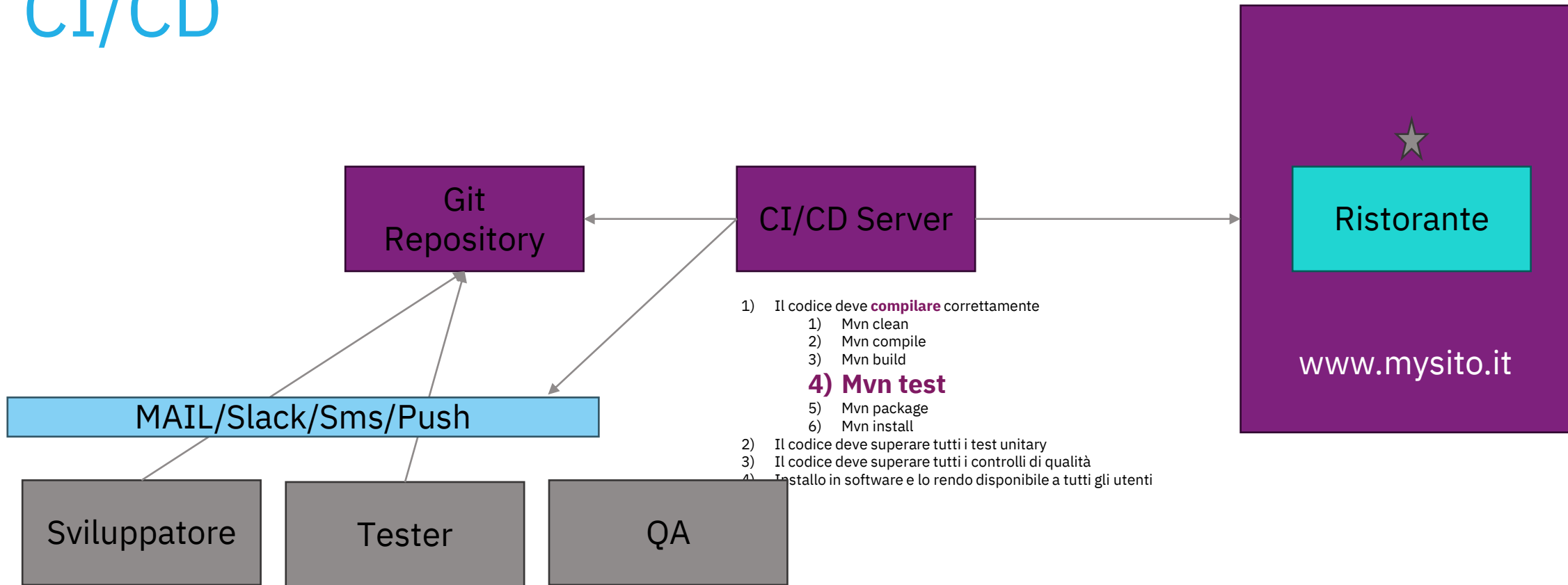


Importazione progetto

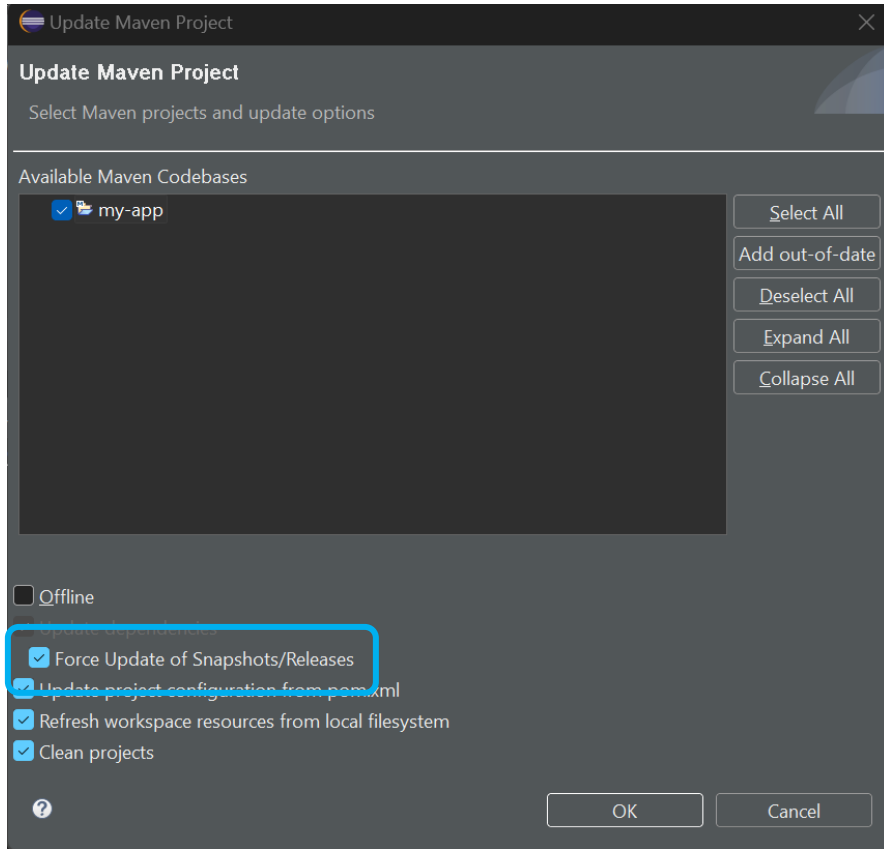
❖ Importiamo un progetto Maven presente nel workspace: `File > Import`



Maven CI/CD



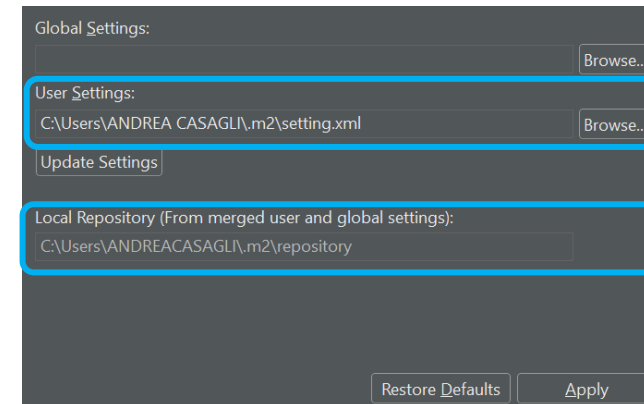
Operazioni base: Update Project



❖ Tasto dx sul Progetto > Maven > Update Project

L'operazione di Update Project consente di sincronizzare le impostazioni di Eclipse con quanto contenuto nel POM. Facendo ciò vengono verificate e scaricate eventuali nuove dipendenze e viene aggiornata la cartella repository.

La locazione del repository e del file setting.xml (che contiene le informazioni per scaricare le dipendenze) può essere verificata o modificata da: Window > Preferences > Maven > User Settings



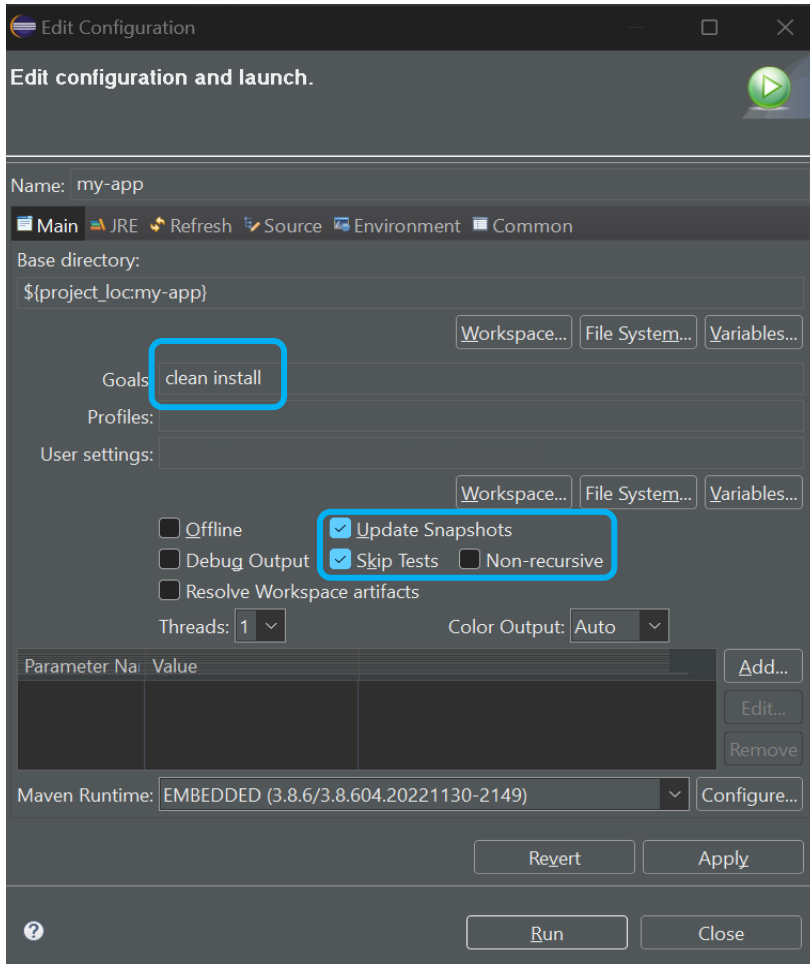
Operazioni base: Lifecycle e Build Project

- ❖ **Maven Lifecycle:** sequenza di passaggi per raggiungere un determinato obiettivo.
- ❖ Esistono tre cicli di vita della build principali:
 - **Default:** il ciclo di vita principale, poiché è responsabile della distribuzione del progetto
 - **Clean:** per pulire il progetto e rimuovere tutti i file generati dalla build precedente
 - **Site:** per creare la documentazione del sito del progetto

Ogni ciclo di vita è composto da più fasi. Di seguito le fasi principali della build di default:

- **validate:** controlla se sono disponibili tutte le informazioni necessarie per la compilazione;
 - **compile:** compila il codice sorgente;
 - **test:** eseguire unit test;
 - **package:** impacchetta il codice sorgente compilato nel formato distribuibile (jar, war, ...)
 - **verify:** verifica che siano rispettati i controlli di qualità;
 - **install:** installa il pacchetto nel repository locale;
 - **deploy:** copia il pacchetto nel repository remote per la condivisione con altri sviluppatori;
- ❖ Se invoco una fase qualsiasi, vengono eseguite tutte le fasi precedenti fino a quella invocata
 - ❖ Le fasi e i goals possono essere invocati insieme sulla stessa linea di comando.

Operazioni base: Lifecycle e Build Project



❖ Tasto dx sul Progetto > Run As > Maven Build...

Nel campo Goal digitare `clean install`. Mettere il check all'opzione Update Snapshots.

Una volta lanciata la prima volta il comando sarà disponibile senza ulteriori configurazioni in :Tasto dx sul Progetto > Run As > Maven Build

Operazioni base: Lifecycle e Build Project

- ❖ Il comando `clean install` esegue, in sequenza:
 - La fase `clean` del build Lifecycle `clean`;
 - Tutte le fasi fino alla fase `install` (compresa la `install` stessa) del build Lifecycle `default`;
- ❖ Ad ogni fase possono essere associati uno o più goal:

```
[INFO] -----< myCompany:myApp3 >-----
[INFO] Building myApp3 0.0.1-SNAPSHOT
[INFO] -----[ jar ]-----
[INFO]
[INFO] --- maven-clean-plugin:3.1.0:clean (default-clean) @ myApp3 ---
[INFO] Deleting C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\target
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:resources (default-resources) @ myApp3 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\src\main\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:compile (default-compile) @ myApp3 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\target\classes
[INFO]
[INFO] --- maven-resources-plugin:3.0.2:testResources (default-testResources) @ myApp3 ---
[INFO] Using 'UTF-8' encoding to copy filtered resources.
[INFO] skip non existing resourceDirectory C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\src\test\resources
[INFO]
[INFO] --- maven-compiler-plugin:3.8.0:testCompile (default-testCompile) @ myApp3 ---
[INFO] Changes detected - recompiling the module!
[INFO] Compiling 1 source file to C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\target\test-classes
[INFO]
[INFO] --- maven-surefire-plugin:2.22.1:test (default-test) @ myApp3 ---
[INFO] --- maven-jar-plugin:3.0.2:jar (default-jar) @ myApp3 ---
[INFO] Building jar: C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\target\myApp3-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- maven-install-plugin:2.5.2:install (default-install) @ myApp3 ---
[INFO] Installing C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\target\myApp3-0.0.1-SNAPSHOT.jar to C:
[INFO] Installing C:\Users\ANDREACASAGLI\workspace\JavaAcademy2023\myApp3\pom.xml to C:\Users\ANDREACASAGLI\m2\
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time: 15.089 s
[INFO] Finished at: 2023-01-28T16:26:14+01:00
[INFO] -----
```

Esercizio

- ❖ Riscrivere il programma per la conversione di un oggetto Java in formato JSON.
- ❖ Importare all'interno del pom.xml le dovute dipendenze;
- ❖ Effettuare l'update del progetto;
- ❖ Scrivere la classe con I due metodi:
 - ❖ **Serializzazione**: conversione da oggetto Java a JSON;
 - ❖ **Deserializzazione**: conversione da JSON a oggetto Java;
- ❖ Scrivere il test JUNIT per verificare la correttezza dei due metodi.

```
<dependency>  
<groupId>com.fasterxml.jackson.core</groupId>  
<artifactId>jackson-databind</artifactId>  
<version>2.15.3</version>  
</dependency>
```

```
<dependency>  
<groupId>com.fasterxml.jackson.dataformat</groupId>  
<artifactId>jackson-dataformat-xml</artifactId>  
<version>2.11.1</version>  
</dependency>
```

```
<groupId>junit</groupId>  
<artifactId>junit</artifactId>  
<version>4.13.2</version>  
<scope>test</scope>  
</dependency>
```

Esercizio – Pokemon

Pakomon

❖ Abbiamo una classe astratta che rappresenta il **Pokemon (nome, tipologia, vita, <mossa>) – Attacca - Cura:**

- ❖ Bulbasaur;
- ❖ Squirtle;
- ❖ Charmender;
- ❖ Pickacu;

❖ **GameManager:** gestisce la partita tra due pokemon;

❖ Posizionarsi sul branch **feature/pokemon;**

❖ **Creare un package pokemon;**

❖ **GameManager** -> Davide V. - $SALUTE\ PERSA = (ATTACCO(P1) * POTENZA(MOSSA)) / DIFESA(P2)$

❖ **Bulbasaur** -> Davide B.

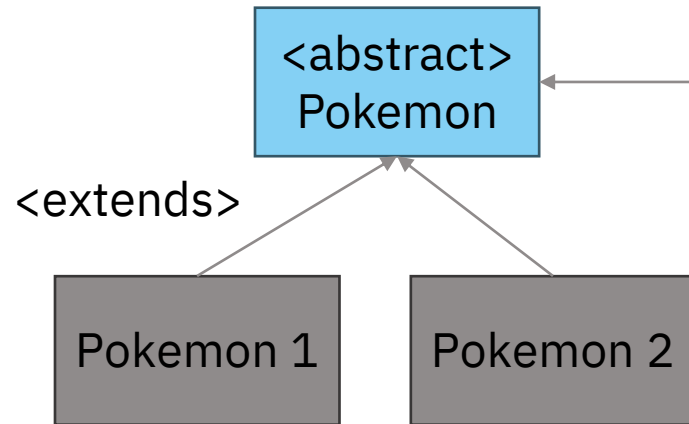
❖ **Charmender** -> Gennaro

❖ **Squirtle** -> Luca

❖ **Pickacu** -> Flavio

❖ **????** -> Marco

Esercizio – Pokemon



```
void battle(Pokemon p1, Pokemon p2);
```

Sort tra p1 e p2



Se p1 ha ancora vite contrattacca

