Maven

Bernardo Cuteri

Che cos'è Maven?

- Project management tool per Java (Strumento di gestione dei progetti)
- Basato sul concetto di **Project Object Model (POM)**
- Gestisce il processo di building di un progetto (dal codice al programma)
- Supporta nativamente il riutilizzo e l'integrazione dei progetti

Maven build lifecycle

Maven è un sistema di building. Le principali fasi del build lifecycle di Maven sono:

- **validate**: valida che il progetto sia corretto e le informazioni necessarie siano disponibili
- **compile**: compila i sorgenti del progetto
- **test**: effettua il testing dei sorgenti compilati utilizzando un framework di unit testing
- **package**: prende il codice compilato e lo racchiude in un formato distribuibile detto package o archivio (ad esempio come JAR)
- **install**: installa il package nel repository locale, così da poter usare il package in altri progetti locali
- **deploy**: copia il package finale nel repository remoto consentendo la condivisione del package con altri sviluppatori e progetti

Maven build lifecycle (cont.)

Altre fasi importanti:

- **site**: per la generazione della documentazione
- **clean**: per la pulizia (rimuove i file di output)

Maven build lifecycle (cont.)

- Le fasi del build life-cycle formano una catena
- Per eseguire una fase vengono eseguite tutte le fasi precedenti: ad esempio, effettuando la fase di package, verranno eseguite le fasi di validate, compile e test.

Maven keywords

- POM
- Coordinate Maven
- Dipendenze
- Archetype
- Goal e Plugin

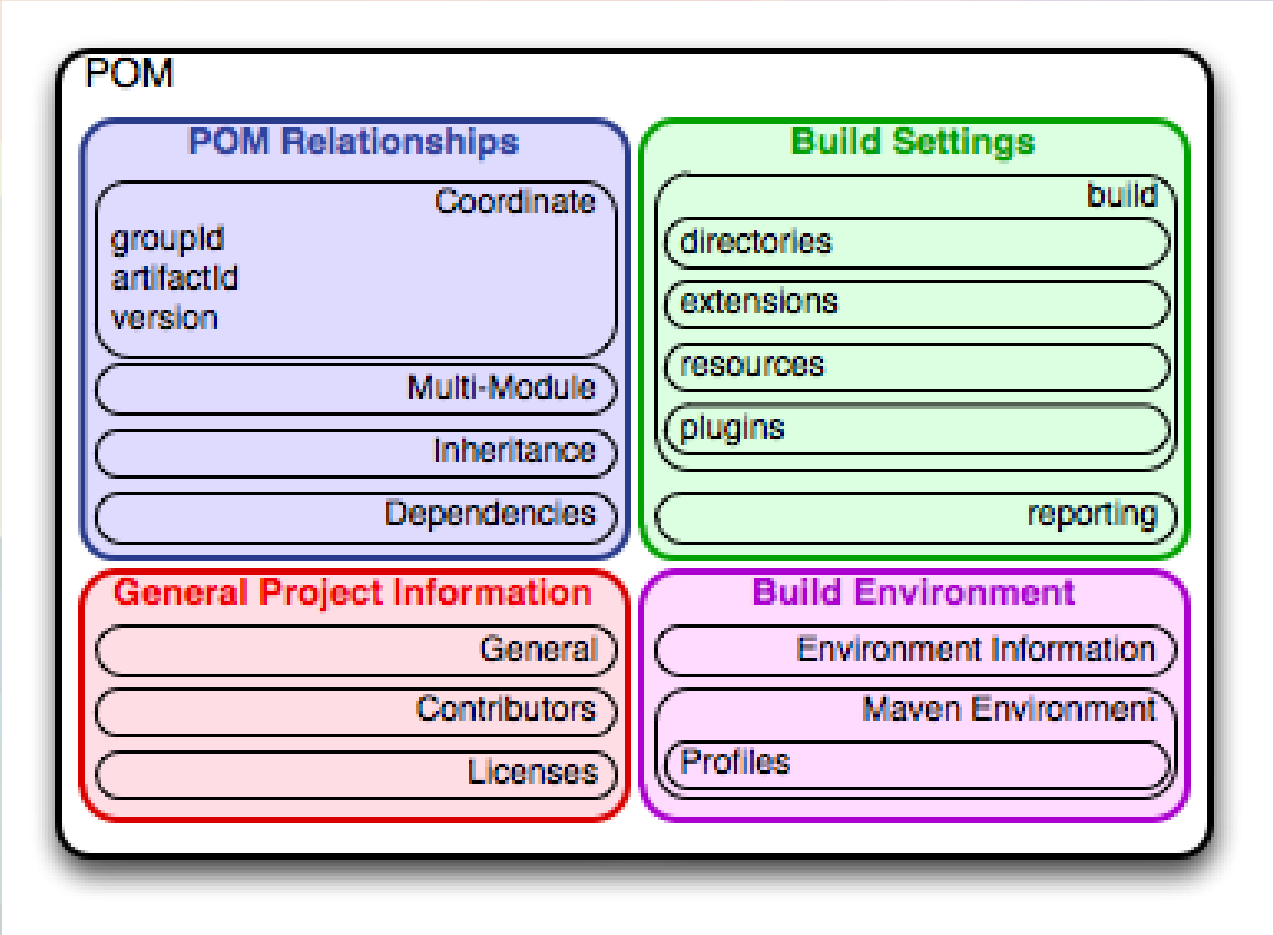
POM

Le informazioni riguardati un progetto Maven sono centralizzate nel file pom.xml

```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

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

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

POM (CONT.)

Un pom è suddiviso in diverse parti



Coordinate Maven

Un progetto Maven è identificato da una tripla:
<groupId, artifactId, version>

```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

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

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

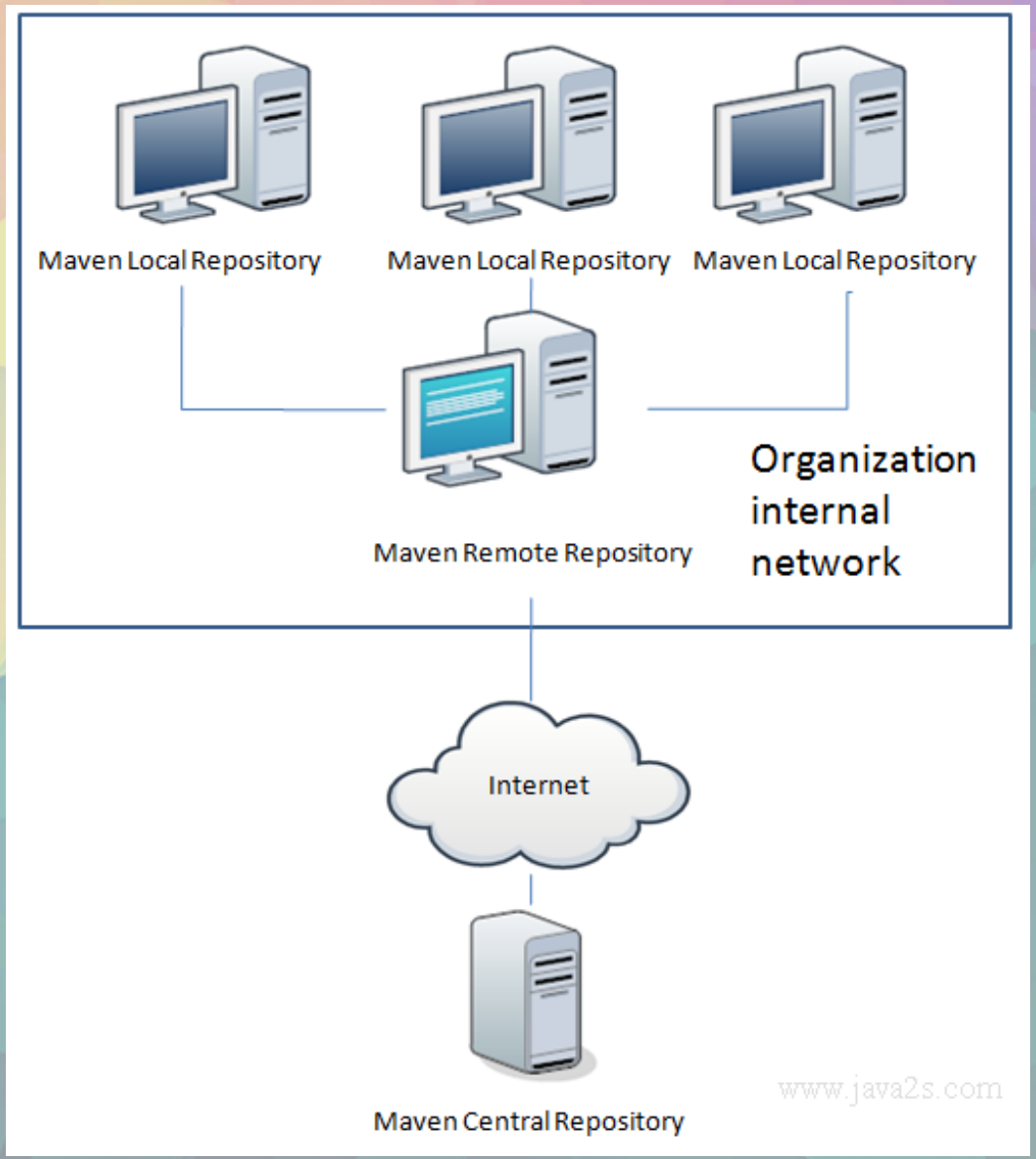
Dipendenze

- Maven permette di dichiarare le dipendenze del progetto dentro il file pom.xml
- Non è necessario scaricare manualmente i JAR e includerli nel progetto
- Maven utilizza un sistema a repository
- Repository locale (cartella .m2e), remoto e centrale

Dipendenze (cont.)

```xml
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- project coordinates, i.e. a group of values which
       uniquely identify this project -->

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

  <!-- library dependencies -->

  <dependencies>
    <dependency>

      <!-- coordinates of the required library -->

      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>

      <!-- this dependency is only used for running and compiling tests -->

      <scope>test</scope>

    </dependency>
  </dependencies>
</project>
```

ARCHITETTURA DEI REPOSITORY

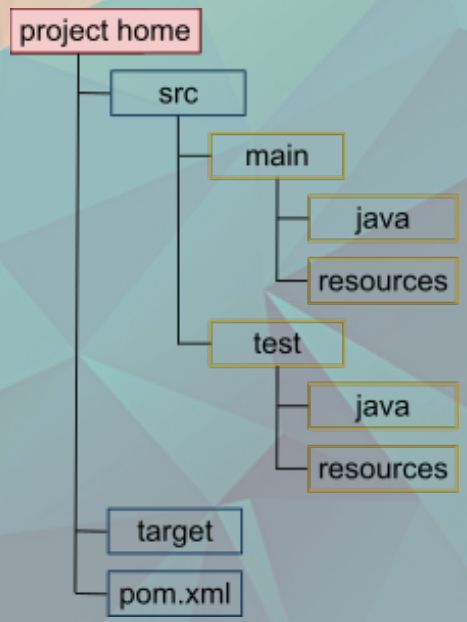


Archetype

- Gli archetipi (archetypes) sono dei template di progetto dai quali partire quando si crea un progetto Maven.
- Comando: mvn archetype:generate
- Crea una struttura di cartelle e un file POM adeguati rispetto all'archetipo scelto

Convention Over Configuration

- Maven segue il principio *Convention Over Configuration*
- Vengono utilizzate delle convenzioni di default (es. posizione dei file, nomi) e questo rende la configurazione minimale
- E' comunque possibile modificare le configurazioni
- Un esempio di convenzione è la struttura delle cartelle



Goal e Plugin

- I goal sono delle azioni eseguibili da Maven per raggiungere un determinato obiettivo
- Le fasi di Maven sono dei goal: package, test, install ecc..
- I goal sono forniti da artefatti Maven noti come **Plugin**
- Maven ha dei plugin di default che sono sempre inclusi e forniscono (tra l'altro) i goal del build life-cycle
- Altri Plugin possono essere aggiunti per poter eseguire dei goal specifici

GOAL E PLUGIN (CONT.)

Esistono due tipi di plugin:

- **build plugins**: pensati per essere eseguiti durante la fase di build del progetto (compilazione, package..)
- **reporting plugins**: pensati per essere eseguiti durante la generazione della documentazione