Apache Maven

Carmine Spagnuolo
Ph.D Student in Computer Science

spagnuolocarmine@gmail.com
cspagnuolo@unisa.it

April 26, 2015

Indice

Prerequisiti della lezione

- Conoscenza base del linguaggio JAVA.
- Conoscenza base della sintassi e della struttura XML.
- Laptop o workstation (Windows, Linux o Mac OS).

Build automation

- Uno strumento di *Build automation* si occupa di automatizzare il processo di **build** di un prodotto software.
- Sviluppare un prodotto software richiede di scrivere il codice (parte divertente :-D) ma anche di:
  - compiling computer source code into binary code;
  - packaging binary code;
  - running automated tests;
  - deploying to production systems;
  - creating documentation and/or release notes (!!!).

Build automation – Tools

- *Ant*, Java
- ...
- *CMake*, cross-language
- ...
- *Gradle*, cross-language
- ...
- *make*, cross-language
- *Maven*, Java
- ...
- *Visual Build*, cross-language
- ...

— http://en.wikipedia.org/wiki/List_of_build_automation_software.

Apache Maven

Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

— https://maven.apache.org/

Apache Maven

- Prima domanda. Cosa significa Maven?
  - *A maven (also mavin) is a trusted expert in a particular field, who seeks to pass knowledge on to others. The word maven comes from Hebrew, meaning "one who understands", based on an accumulation of knowledge.*

— https://maven.apache.org/what-is-maven.html

— http://en.wikipedia.org/wiki/Maven

Apache Maven

- Maven è uno strumento completo per la gestione di progetti software JAVA:
  - compilazione del codice;
  - distribuzione;
  - verifica (test);
  - documentazione;
  - deployment e relativa configurazione [SCM (Software Configuration Management)];
  - collaborazione del team di sviluppo.
- *L'idea è quella di applicare pattern ben collaudati all'infrastruttura per la build di progetti JAVA.*

Apache Maven – History

- *2002*. Nato per semplificare il processo di sviluppo del progetto Jakarta Turbine (Struts, Tomcat, Velocity,etc.).
- *2003*. Integrato come progetto ufficiale della Apache Software Foundation.
- *2004*. Apache Maven v1.
- *2005*. Apache Maven v2.
- *2010*. Apache Maven v3 (compatibile con la versione 2).

Apache Maven – Obiettivi

- Semplificare il processo di 'build'.
- Offrire un sistema di 'build' standard (POM).
- Offrire informazioni circa la qualità del software (change log automatici, cross referenced sources, mailing list, dependency list, Unit testing).
- Offrire guidelines e best practices per il processo di sviluppo (struttura dei progetti src e test).
- Consentire la migrazione trasparente di nuove caratteristiche (download automatico dipendenze etc.).

Apache Maven – Mindset

Processo di 'build'?

- Set up dependencies
- Compile Source code
- Copy Resource
- Compile and Run Tests
- Package Project
- Deploy Project
- Cleanup

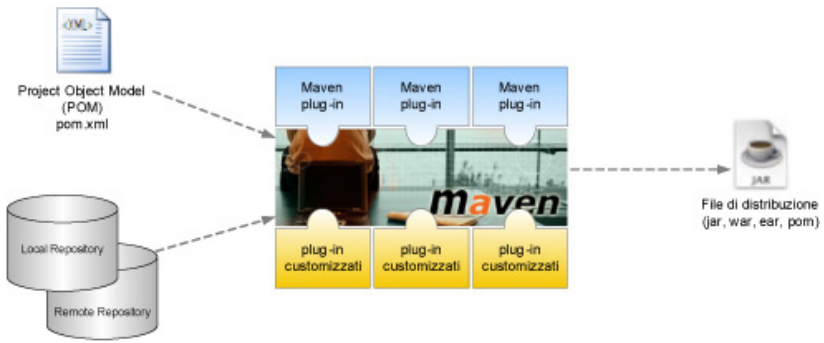
Apache Maven – Altri tool

- Ant (2000)
  - Java Build Tools Il primo di tutti!
  - Scripting in XML
  - Molto flessibile
- Ant+Ivy (2004)
  - Ant e Dependency Management
- Gradle (2008)
  - Maven e Groovy Scripting (linguaggio per JVM)
  - Estensibilità
  - Immaturo

Apache Maven – Vantaggi

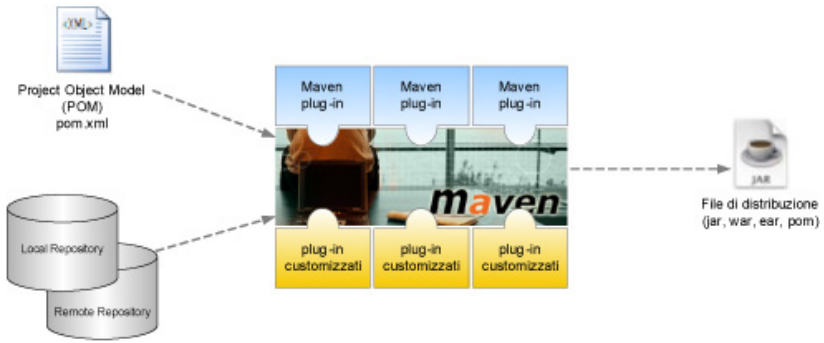
- **Coerenza**: le varie organizzazioni possono standardizzare la gestione dei progetti Java utilizzando l'insieme di best practice alla base di Maven.
- **Riutilizzo**: la business logic è incapsulata in comodi moduli (plug-in);
- **Maggiore agilità**: semplifica il processo di generazione di nuovi componenti, di integrazione tra componenti, di condivisione di file eseguibili.
- **Semplificazione della manutenzione**.

Apache Maven – Architettura



- File di progetto, pom.xml (POM, Project Object Model).
- Goal: è l'equivalente Maven dei task Ant. Funzione eseguibile che agisce su un progetto. I goal possono essere sia specifici per il progetto dove sono inclusi, sia riusabili.

Apache Maven – Architettura



- Jelly script: è utilizzato per descrivere i goal.

  — Jelly is a tool for turning XML into executable code. So Jelly is a Java and XML based scripting and processing engine.
- Plug-in: si tratta di goal riutilizzabili e cross-project.
- Repository: si tratta di un meccanismo che permette di memorizzare file di distribuzione.

Apache Maven – Archetype

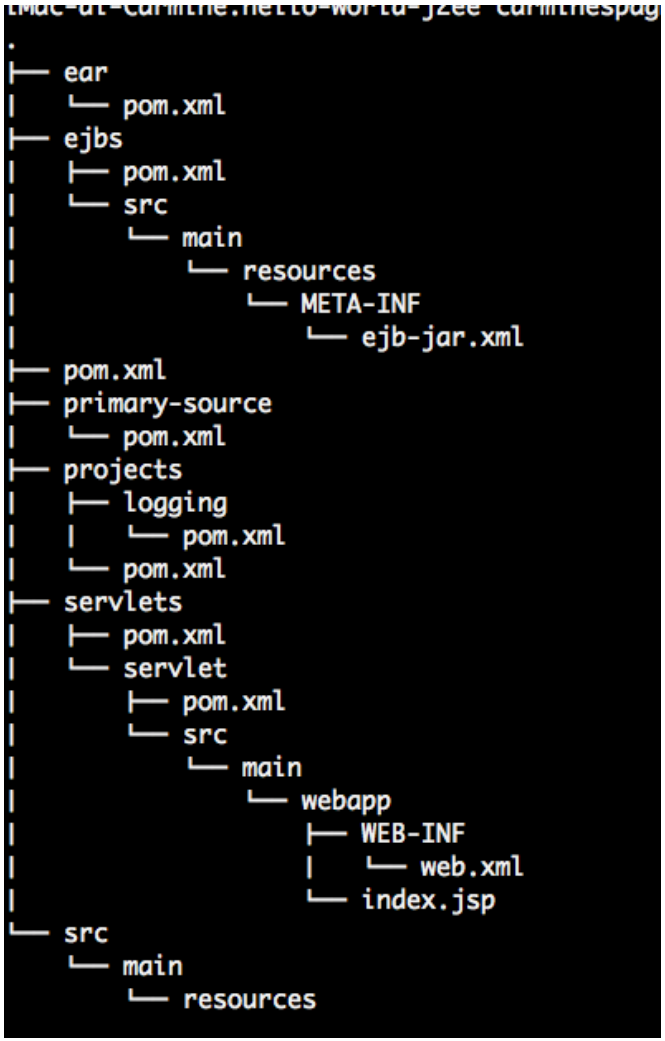
Il principale standard utilizzato in Maven è l'archetipo: struttura delle directory del progetto.

- E' possibile uniformare i progetti in un'organizzazione.
- Conformarsi e/o uniformarsi a standard (es. progetti JAVA EE).

Primo comando MAVEN 3.0: **mvn archetype:generate -DgroupId=it.isislab.test -DartifactId=hello-world -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false**

Apache Maven – Archetype

Apache Maven – Archetype

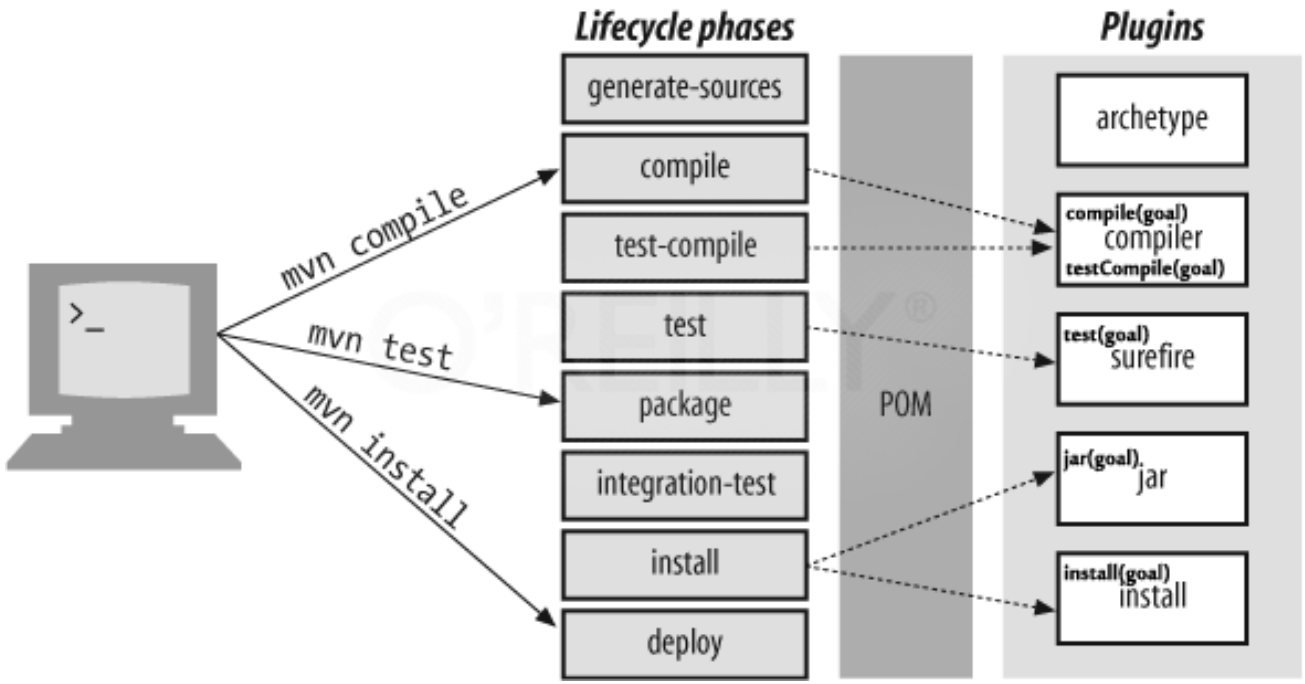


mvn archetype:generate
-DgroupId=it.isislab.test
-DartifactId=hello-world
-DarchetypeArtifactId=maven-archetype-j2ee-simple
-DinteractiveMode=false

Apache Maven – Comandi base

1. *mvn* **test-compile**, compila ma non esegue lo JUnit test (oppure utilizzare il parametro *-DskipTests=true*).
2. *mvn* **test**, compila ed esegue JUnit.
3. *mvn* **package**, genera il file di distribuzione (default è jar).
4. *mvn* **install**, permette di installare il file di distribuzione nel repository locale in modo che i progetti dipendenti/aggregati possano utilizzarlo.
5. *mvn* **clean**, rimuove dalla directory target i file.
6. *mvn* {**idea,eclipse**}:{**idea,eclipse**}, genera i file descrittori per le IDE IntelliJ e/o eclipse.
7. *mvn* **netbeans-freeform:generate-netbeans-project**, genera i file descrittori per Netbeans.

Apache Maven – Goal to Phase Bindings



Apache Maven – POM

- POM (Porject Object Model)
- Descrive il progetto in formato XML:
  - Nome e versione;
  - Artifact Type;
  - Source Code Locations;
  - Dipendenze;
  - Maven goal;
  - Maven plugins;
  - Maven profiles (Alternate build configurations);
  - Developers;
  - Mailing list.

Apache Maven – POM example

```
<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.companyname.project-group</groupId>
<artifactId>gtproject</artifactId>
<version>1.0</version>

</project>
```

Apache Maven – POM example

- *groupId*, identificativo univoco del gruppo di progetto, es. package, consente la gestione di gerarchie di progetti.
- *artifactId*, identificativo univoco del progetto.
- *version*, versione del progetto.

Campi necessari in ogni POM.

Apache Maven – POM struttura (definizioni)

```
<project>
    <modelVersion>4.0.0</modelVersion>
    <!-- POM Relationships -->
    <groupId>...</groupId>
    <artifactId>...</artifactId>
    <version>...</version>
    <packing>...</ packing>
    <parent>...</parent>
    <dependencyManagement>...</dependencyManagement>
    <dependencies>...</dependencies>
    <modules>...</modules>
     ....
```

Apache Maven – POM struttura (informazioni)

```
<!-- Project Information -->
    <name>...</name>
    <description>...</description>
    <url>...</url>
    <inceptionYear>...</inceptionYear>
    <licenses>...</licenses>
    <developers>...</developers>
    <contributors>...</contributors>
    <organization>...</organization>
    .....
```

Apache Maven – POM struttura (impostazioni di build)

```
<!-- Build Settings -->
    <packaging>...</packaging>
    <properties>...</properties>
    <build>...</build>
    <reporting>...</reporting>
    .....
```

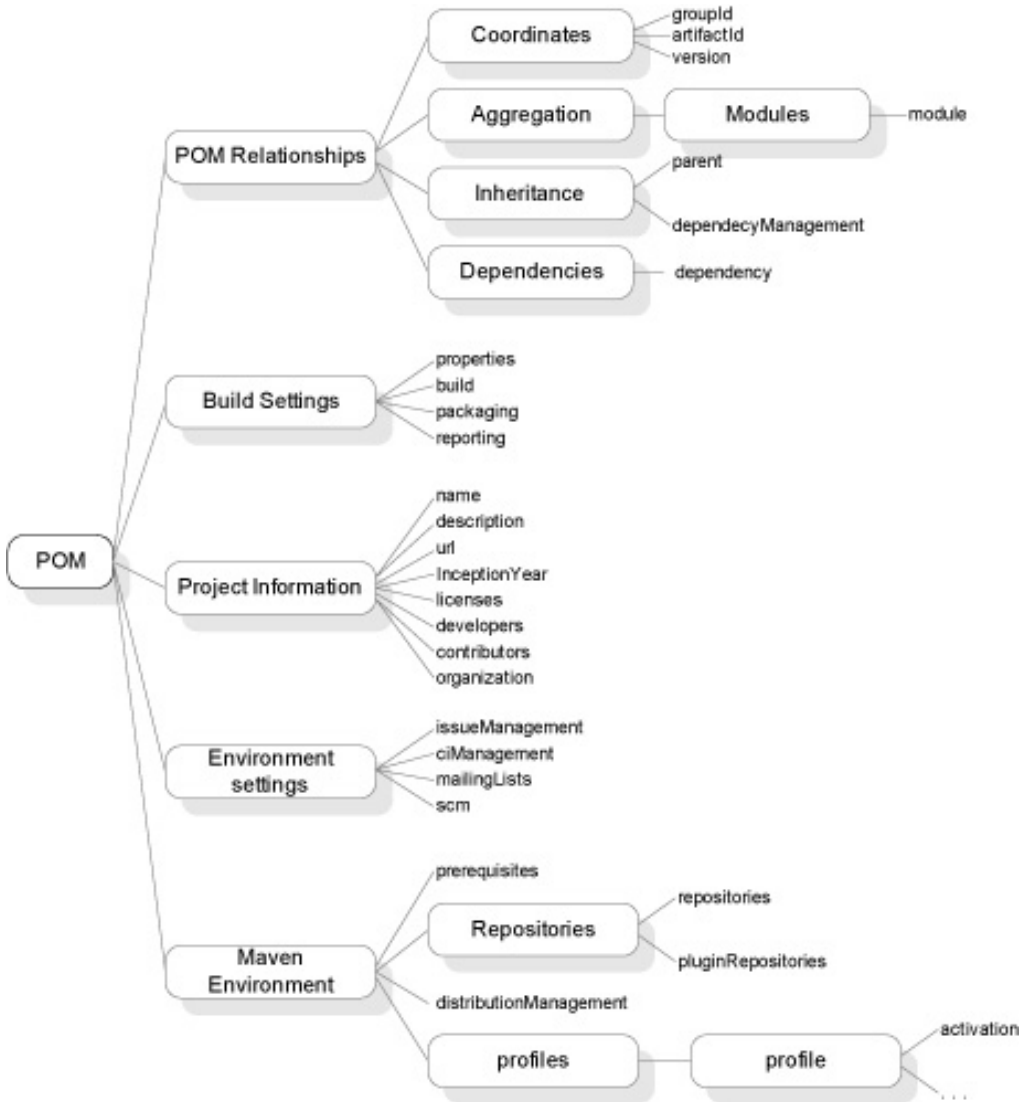
Apache Maven – POM struttura (ambiente di build)

```
<!-- Build Environment -->
<!-- Environment Information -->
<issueManagement>...</issueManagement>
<ciManagement>...</ciManagement>
<mailingLists>...</mailingLists>
<scm>...</scm>
.....
```

Apache Maven – POM struttura (ambiente MAVEN)

```
    <!-- Maven Environment -->
    <prerequisites>...</prerequisites>
    <repositories>...</repositories>
    <pluginRepositories>...</pluginRepositories>
    <distributionManagement>...</distributionManagement>
    <profiles>...</profiles>
</project>
```

Apache Maven – POM struttura logica



Apache Maven – POM struttura logica

Differenze con Ant:

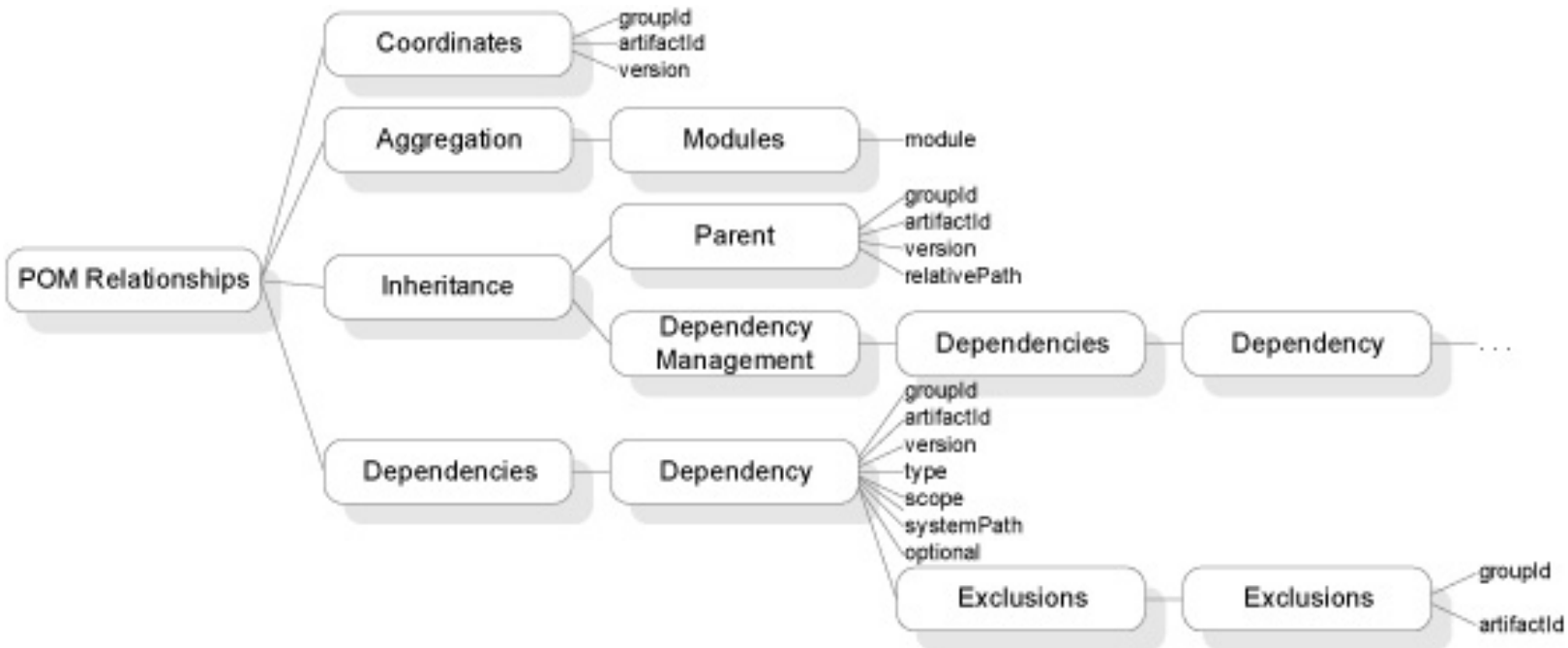
- *il file POM contiene la dichiarazione del progetto e non le azioni da eseguire nel processo di build.*
- ma.....
  - è possibile inserire nei file POM la dichiarazione di eseguire dei task Ant (si utilizza il plugin **maven-antrun-plugin**).
- Filosoficamente i file POM definiscono "chi", "cosa" e "dove", mentre i file di build Ant si limitano al "quando" e al "come".

Apache Maven – POM *Relationships*

POM relationships:

- permette di organizzare i progetti attraverso una serie di file POM opportunamente relazionati. Le relazioni possibili sono:
  - dipendenza;
  - ereditarietà;
  - aggregazione.

Apache Maven – POM *Relationships*



Apache Maven – POM *Relationships, Coordinates*

Coordinates:

- *groupId*, identificativo univoco del gruppo di progetto, consente la gestione di gerarchie di progetti (può essere ereditato).
  - *es. org.apache.Maven, la notazione a struttura package col punto non è necessaria ma è un'ottima convenzione*.
  - in realtà il punto verrà sostituito (" /" in UNIX) per la costruzione del percorso relativo all'interno del repository Maven locale.
- *artifactId*, identificativo univoco del progetto (può essere ereditato).
  - in combinazione con groupId genera l'identificativo univoco per il progetto.
- *version*, versione del progetto (non può essere ereditato).

Apache Maven – POM *Relationships, Dependencies*

Dependencies:

- Consente la definizione delle dipendenze del progetto (librerie e sotto-progetti, anche da terze parti).
- Maven è in grado di eseguire il download automatico dei manufatti richiesti e di risolvere automaticamente eventuali dipendenze transitive (dipendenze dei sotto-progetti).

```
<dependencies>
     <dependency>
      <groupId>log4j</groupId>
      <artifactId>log4j</artifactId>
      <version>1.2.6</version>
      <type>jar</type>
     </dependency>
</dependencies>
```

Apache Maven – POM *Relationships, Dependencies*

Dependencies:

- Il campo scope prevede:
  - *compile*: valore di default, prevede che il file incluso sia disponibile nel percorso base;
  - *provided*: è molto simile al precedente, ma il file è fornito da un'altra entità (JDK o un app. container);
  - *runtime*: indica che la dipendenza non è richiesta a tempo di compilazione, ma solo in esecuzione. Pertanto è disponibile solo nel percorso di runtime e/o di test.

Apache Maven – POM *Relationships, Dependencies*

Dependencies:

- Il campo scope prevede:
  - *test*: specifica che la dipendenza è richiesta solo nella fase di test;
  - *system*: è molto simile a provided con l'eccezione che è necessario fornire esplicitamente il jar che contiene la dipendenza, specificando il path locale (sconsigliato, meglio creare un repository e renderlo disponibile sul web es. GitHub).

Apache Maven – POM *Relationships, Inheritance*

- Un POM può ereditare da un altro POM (Super POM).
- Per verificare la configurazione reale del file POM in caso di ereditarietà è possibile eseguire il comando: **mvn help:effective-pom**.

```
<project>
    <modelVersion>4.0.2</modelVersion>
    <parent>
        <groupId>com.company.finance</groupId>
        <artifactId>frontoffice</artifactId>
        <version>2</version>
    </parent>
    <artifactId>PriceBlotter</artifactId>
</project>
```

Apache Maven – POM *Relationships, Inheritance*

Gli elementi ereditabili dai discendenti che si possono specificare in un POM genitore sono:

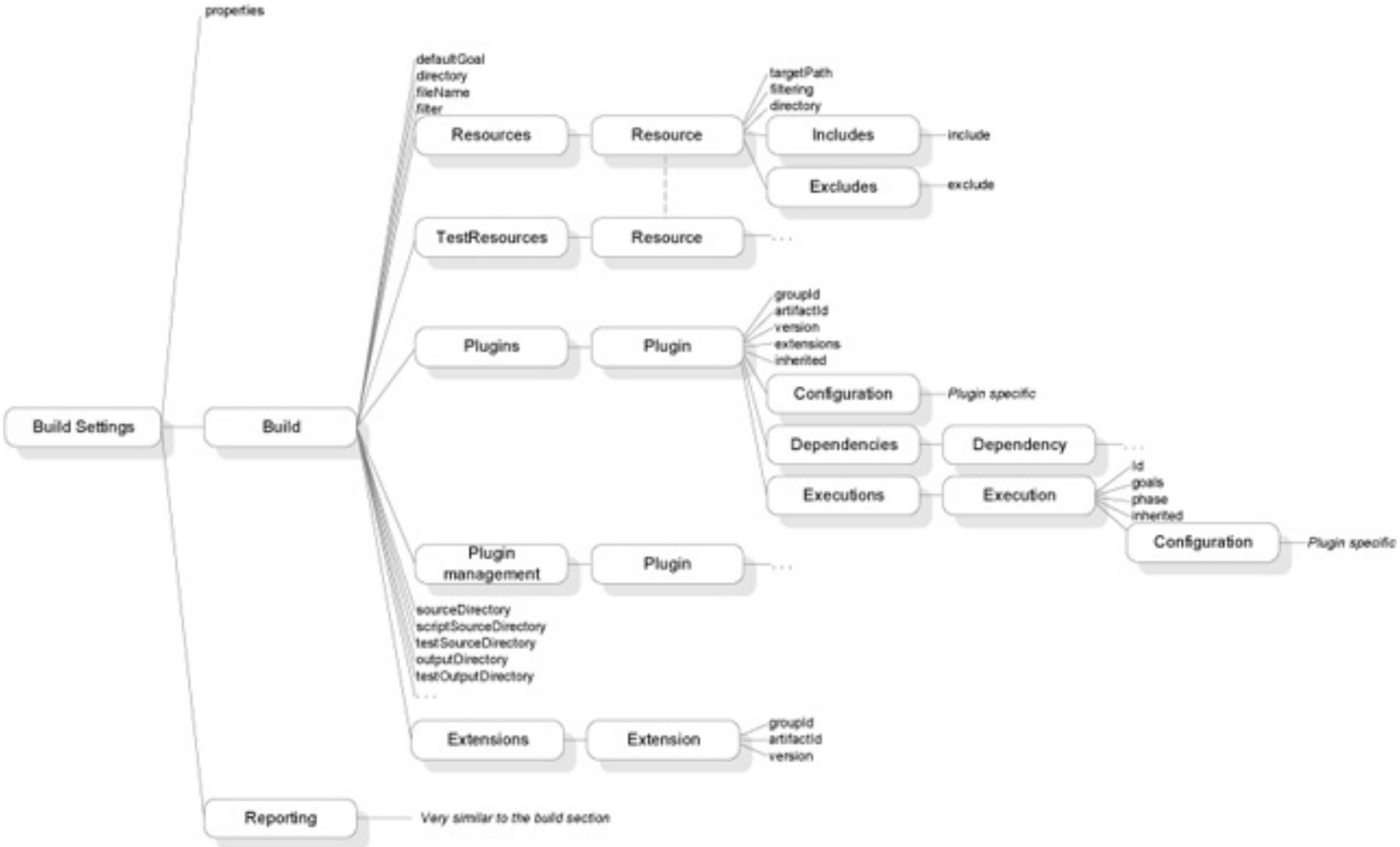
- dependencies (dipendenze);
- developers and contributors (lista degli sviluppatori e contributori);
- plugin lists (elenchi dei plugin);
- reports lists (elenchi dei reports);
- plugin executions with matching id (esecuzioni dei plugin con identificatori corrispondenti);
- plugin configuration (configurazione dei plugin).

Apache Maven – POM *Relationships, Aggregation*

Un progetto Maven ha la possibilità di aggregare diversi moduli; in tal caso prende il nome di progetto multi-modulo (multimodule) o aggregatore (aggregator).

```
<modules>
    <module>client-system-endpoint</module>
    <module>admin-dto</module>
    <module>admin-gui</module>
    <module>client-api</module>
    <module>server</module>
</modules>
```

Apache Maven – POM *Build Settings*



Apache Maven – POM *Build Settings, Properties*

Maven mette a disposizione una serie di proprietà, molto simili a quanto avviene con Ant:

- una una sorta di variabili per i file POM;
- ${}, ${prefisso.<proprietà>}:
  - env.<proprietà>: il prefisso env fa riferimento alle variabili di ambiente, come per esempio il percorso (${env.PATH});
  - project.<proprietà>: in questo caso il suffisso project permette di accedere agli elementi del file POM;
  - settings.<proprietà>: permette di accedere agli elementi all'interno di un file di configurazione di Maven;
  - java.<proprietà>: permette di accedere alle proprietà di sistema messe a disposizione dalla API java.lang.System.getProprietes. Per esempio, ${java.home};
  - <proprietà>: fa riferimento a valori specificati nella struttura <properties> ... </properties>.

Apache Maven – POM *Relationships, Properties*

User-defined Properties in a POM.

```
<project>
    ...
    <properties>
        <arbitrary.property.a>
              This is some text
        </arbitrary.property.a>
        <hibernate.version>
              3.3.0.ga
        </hibernate.version>
        ..
        ...
    </properties>
    ...
```

Apache Maven – POM *Relationships, Properties*

User-defined Properties in a POM.

```
    ...
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate</artifactId>
            <version>${hibernate.version}</version>
        </dependency>
    </dependencies>
    ...
</project>
```

Apache Maven – POM *Relationships, Properties*

User-defined Properties in a Profile in a POM.

```
<project>
    ...
    <profiles>
        <profile>
            <id>some-profile</id>
            <properties>
                <arbitrary.property>
                This is some text
                </arbitrary.property>
            </properties>
        </profile>
    </profiles>
    ...
</project>
```

Apache Maven – POM *Build Settings, baseBuild*

Contiene informazioni relative al processo di build (la directory dei file sorgente, la configurazione dei vari plug-in, e così via).

```
<build>
    <defaultGoal>install</defaultGoal>
    <directory>${basedir}/target</directory>
    <finalName>${artifactId}-${version}</finalName>
    <filters>
        <filter>filters/filter1.properties</filter>
    </filters>
    ...
</build>
```

Apache Maven – POM *Build Settings, Resources*

Consente di definire tutti i file che non vanno compilati ma fanno parte del processo di distribuzione:

- *resources*: contiene la lista delle varie risorse. Ognuna specifica i file da includere e relativa allocazione.
- *targetPath*: specifica la struttura della directory dove ubicare il file delle risorse. Il default è la directory base, tuttavia la convenzione adottata nei file JAR è quella di includere le risorse a partire dalla directory META-INF.
- *filtering*: si tratta di un elemento booleano che indica se abilitare o meno il filtraggio della risorsa durante la copia nella directory resources.

Apache Maven – POM *Build Settings, Resources*

Consente di definire tutti i file che non vanno compilati ma fanno parte del processo di distribuzione:

- *direcotry*: specifica la directory resources. Il valore di default è: $basedir/src/main/resources.
- *includes*: consente di definire quali file includere.
- *excludes*: speculare di includes.
- *testResources*: questa sezione è dedicata a risorse utilizzate solo nella fase di test.

Apache Maven – POM *Build Settings, Resources*

```
<build>
    ...
    <resources>
        <resource>
            <targetPath>META-INF/project</targetPath>
            <filtering> false</filtering>
            <directory>
                ${basedir}/src/main/project
            </directory>
            <includes>
                <include>configuration.xml</include>
            </includes>
..............
```

Apache Maven – POM *Build Settings, Resources*

```
.......
            <excludes>
                <exclude>**/*.properties</exclude>
            </excludes>
        </resource>
    </resources>
    <testResources>
        ...
    </testResources>
    ...
</build>
```

Apache Maven – POM *Build Settings, Plug-in*

Consente di definire tutti i plugin utilizzati nel processo di build, configurazione base:

```
<plugins>
    <plugin>
        <groupId>org.apache.Maven.plugins</groupId>
        <artifactId>Maven-compiler-plugin</artifactId>
        <configuration>
            <source>1.5</source>
            <target>1.5</target>
        </configuration>
    </plugin>
</plugins>
```

Apache Maven – POM *Build Settings, Plug-in*

Configurazione avanzata:

- *extensions*: specifica se caricare, o meno le estensioni del plug-in. Il valore di default è false.
- *inherited*: indica se eventuali file POM, ereditanti da quello corrente ereditino la configurazione del presente plug-in.
- *configuration*: consente di specificare una qualsiasi proprietà che una classe MOJO potrebbe richiedere. MOJO è un gioco di parole utilizzato per definire i Maven POJO (Plain Old Java Object), ossia classi Java (simili a JavaBeans).
- *dependencies*: dipendenze a livello di plug-in.
- *executions*: permette di configurare il goal di un plug-in. Ogni plug-in può definire diversi goal e ciascuno può richiedere una diversa configurazione.

Apache Maven – POM *Build Settings, Plug-in*

```
<build>
    <plugins>
        <plugin>
            <artifactId>
                Maven-antrun-plugin
            </artifactId>
            <executions>
                <execution>
                    <id>echodir</id>
                    <goals>
                      <goal>run</goal>
                    </goals>
                    <phase>verify</phase>
                    <inherited>false</inherited>
```

..

Apache Maven – POM *Build Settings, Plug-in*

```
..
..
                    <configuration>
                        <tasks>
                            <echo>
                              Build Dir:
                              ${project.build.
                              directory}
                             <echo>
                        </tasks>
                    </configuration>
                </execution>
            </executions>
        </plugin>
    </plugins>
</build>
```

Apache Maven – POM *Build Settings, Directories*

MAVEN suddivide i file in due grandi categorie: sorgenti e compilati.

```
<project>
    <build>
        <sourceDirectory>
                ${basedir}/src/main/java
        </sourceDirectory>
        <scriptSourceDirectory>
                ${basedir}/src/main/scripts
        </scriptSourceDirectory>
        <testSourceDirectory>
                ${basedir}/src/test/java
        </testSourceDirectory>
..
..
```

Apache Maven – POM *Build Settings, Directories*

MAVEN suddivide i file in due grandi categorie: sorgenti e compilati.

```
..
..
        <testSourceDirectory>
                ${basedir}/src/test/java
        </testSourceDirectory>
        <outputDirectory>
                ${basedir}/target/classes
         </outputDirectory>
        <testOutputDirectory>
                ${basedir}/target/test-classes
        </testOutputDirectory>
        ...
    </build>
</project>
```

Apache Maven – POM

Le sezioni:

1. *Project Information*, ha un ruolo descrittivo del progetto.
2. *Environment Settings*, è dedicata alle informazioni relative a vari elementi dell'ambiente in cui Maven è installato (stema di gestione delle problematiche, change management, liste di distribuzioni etc.).
3. *Maven environment*, informazioni relative ai repository, alla gestione delle distribuzioni e ai profili.

Esempio Maven – Hello World

http://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

1. *Installazione*
2. *Creazione progetto Java*
3. *Modifica del file POM*
4. *Build del progetto*
5. *Esecuzione*

Esempio Maven – Hello World (Installazione)

1. *Dowload Maven:* http://maven.apache.org/download.cgi.
2. *Install Maven:* http://maven.apache.org/download.cgi#Installation:
   - *Windows*
   - *Unix-Based OS*

Esempio Maven – Hello World (Installazione)

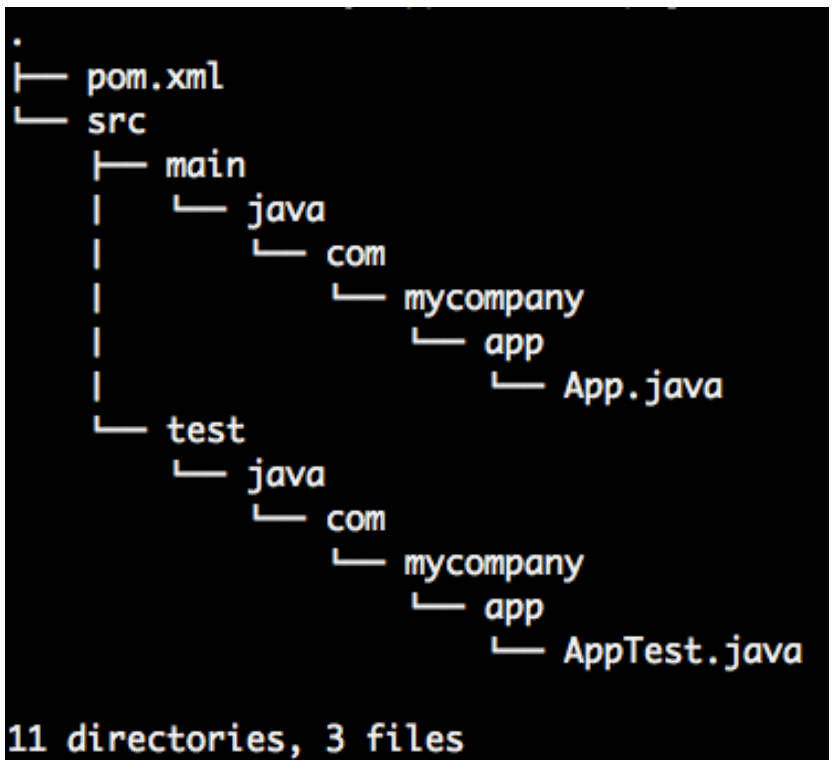
1. *Windows:*
   - Unzip the distribution archive in
     `C:\Program Files\Apache Software Foundation`
   - Add bin path to "Environment Variables" e.g.
     `C:\Program Files\Apache Software Foundation`
     `\apache-maven-3.3.1\bin`
   - Check if the JAVA_HOME is setted e.g.
     `C:\Program Files\Java\jdk1.7.0_51`
   - Check installation run command **mvn –version**

Esempio Maven – Hello World (Installazione)

1. *Unik-Based OS:*
   - Unzip the distribution archive in `/usr/local/apache-maven`
   - Add bin path to "Environment Variables" e.g.
     `export PATH=$PATH:/usr/local/apache-maven/apache-maven-3.3.1/bin`
   - Check if the JAVA_HOME is setted e.g.
     `export JAVA_HOME=/usr/java/jdk1.7.0_51`
   - Check installation run command **mvn –version**

Esempio Maven – Hello World (Nuovo progetto)

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



Esempio Maven – Hello World (Modifica del file POM)

Ricerca della dependency nel repository Maven centrale http://search.maven.org/.

1. Aggiunta di una dipendenza al file POM nel tag $<$ *dependencies* $>$ (Apache ActiveMQ (open source messaging and Integration Patterns server)).

```
<dependency>
    <groupId>org.apache.activemq</groupId>
    <artifactId>apache-activemq</artifactId>
    <version>6.0.0</version>
</dependency>
```

Esempio Maven – Hello World (Build)

1. **mvn package**

```
 ...
[INFO] ------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------
[INFO] Total time: 1.103 s
[INFO] Finished at: 2015-04-22T13:00:33+02:00
[INFO] Final Memory: 10M/245M
[INFO] ------------------------------------------
```

Esempio Maven – Hello World (Esecuzione)

1. **java -cp target/my-app-1.0-SNAPSHOT.jar com.mycompany.app.App**

   ```
   Hello World!
   ```

Riferimenti

Introduzione a Apache Maven Project,
http://maven.apache.org/what-is-maven.html,
ultimo accesso Aprile 2015.

Apache Maven – POM,
http://maven.apache.org/pom.html,
ultimo accesso Aprile 2015.

Maven: guides,
https://maven.apache.org/guides,
ultimo accesso Aprile 2015.

Riferimenti

Maven: The Complete Reference,
http://books.sonatype.com/mvnref-book/reference/,
ultimo accesso Aprile 2015.

Building Web Applications with Maven 2,
https://today.java.net/pub/a/today/2007/03/01/building-web-applications-with-maven-2.html,
ultimo accesso Aprile 2015.

Building J2EE Projects with Maven,
http://www.onjava.com/pub/a/onjava/2005/09/07/maven.html,
ultimo accesso Aprile 2015.