Introduzione a maven

Maven è uno strumento nato per gestire i progetti Java. È in grado di occuparsi, se opportunamente configurato, della risoluzione delle dipendenze, della compilazione, dell'esecuzione dei test, della generazione degli artefatti, della reportistica, della documentazione del progetto e molto altro. Il progetto maven dispone di un repository pubblico centralizzato sul quale vengono caricati gli artefatti di diversi progetti (tipo apache-commons, log4j ecc.) che possono essere dichiarati come dipendenze del proprio progetto, facendo in modo che maven risolva la catena di dipendenze e scarichi in autonomia quanto gli è necessario per eseguire la build.

Progetti e file pom.xml

Ogni progetto è identificato dai seguenti 2 parametri:

- groupId
- artifactId

Il primo è un identificativo univoco del produttore del software (per esempio, org.apache per la Apache Software Foundation), mentre l'artifactId identifica il singolo artefatto/progetto del gruppo (per esempio, commons per il progetto Commons della Apache Software Foundation). Ogni artefatto ha poi anche una versione. Se si necessita dell'ultima build disponibile non ancora dotata di un numero di versione (uno snapshot, in pratica), si può usare il tag -SNAPSHOT come versione per indicare che si desidera utilizzare (o creare, nel caso si tratti della versione del proprio progetto) una versione bleeding-edge non dotata ancora di numero di versione stabile. Tutte queste informazioni (e molte altre) vengono immesse nel file **pom.xml** presente nella root directory del progetto, che rappresenta il modello del progetto e conterrà tutti i metadati necessari a maven per risolvere le dipendenze, compilarlo, eseguire i test, ecc.

Build lifecycle e goals

Il concetto di build di un progetto in maven è concepito come un “build lifecycle”, ovvero un processo ben definito e standard per creare gli artefatti del progetto. Il lifecycle **default** comprende, tra gli altri, i seguenti passaggi:

1. validate
2. compile
3. test
4. package
5. integration-test
6. verify
7. install

Essendo il processo già noto e ben definito, il pom.xml non serve ad altro se non a specificare i parametri necessari ad eseguire tutte le varie fasi dei lifecycle. Nel caso in cui ci si appoggi a plugin aggiuntivi per creare gli artefatti del proprio progetto, è probabile che questi presentino dei loro step aggiuntivi che vanno ad aggiungersi ai passi della normale build: questi step prendono il nome di goal e si specificano solitamente nella forma **nome_plugin:nome_goal**. Un esempio può essere il seguente:

```
mvn clean dependency:copy-dependencies package
```

nel quale si indica a maven di eseguire il lifecycle **clean**, seguito dal goal **copy-dependencies** del plugin **dependency**, seguito poi dal lifecycle **package**.

Per ulteriori approfondimenti, consultare la sezione riferimenti esterni in coda alla pagina.

Sonatype Nexus

Nexus è un'applicazione web che consente, tra le altre cose, la creazione e la gestione di un repository maven accessibile tramite web. Oltre alla possibilità di caricare i propri artefatti e renderli disponibili ad altri, può essere configurato anche come cache-server del repository centrale di maven, rendendo più efficienti i processi di build in ambiti di reti locali.

Un'istanza della versione Open (OSS) di Nexus è stata installata sul server ecatonchiro.bo.priv, ed è raggiungibile alla seguente URL .

Configurazione di maven

Per potersi appoggiare al nostro server Nexus, è necessario configurare maven in modo da istruirlo su dove reperire gli artefatti. Per farlo è sufficiente modificare (o creare, se non già esistente) il file ${HOME}/.m2/settings.xml (%USERPROFILE%/.m2/settings.xml su Windows) col seguente contenuto:

```
<?xml version="1.0"?>
<settings>
  <servers>
    <server>
```

```
      <id>deployment</id>
      <username>USERNAME</username>
      <password>PASSWORD</password>
    </server>
  </servers>
  <mirrors>
    <mirror>
      <!--This sends everything else to /public -->
      <id>nexus</id>
      <mirrorOf>*</mirrorOf>
      <url>http://ecatonchiro:8081/nexus/content/groups/public</url>
    </mirror>
    <mirror>
      <id>nexus_releases</id>
      <mirrorOf>*</mirrorOf>
      <url>http://ecatonchiro:8081/nexus/content/repositories/releases/</url>
    </mirror>
    <mirror>
      <id>nexus_snapshots</id>
      <mirrorOf>*</mirrorOf>
      <url>http://ecatonchiro:8081/nexus/content/repositories/snapshots/</url>
    </mirror>
  </mirrors>
  <profiles>
    <profile>
      <id>nexus</id>
      <!--Enable snapshots for the built in central repo to direct -->
      <!--all requests to nexus via the mirror -->
      <repositories>
        <repository>
          <id>central</id>
          <url>http://central</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </repository>
      </repositories>
      <pluginRepositories>
        <pluginRepository>
          <id>central</id>
          <url>http://central</url>
          <releases>
            <enabled>true</enabled>
          </releases>
          <snapshots>
            <enabled>true</enabled>
          </snapshots>
        </pluginRepository>
      </pluginRepositories>
    </profile>
  </profiles>
  <activeProfiles>
    <!--make the profile active all the time -->
    <activeProfile>nexus</activeProfile>
  </activeProfiles>
</settings>
```

Come si può notare, è anche presente una sezione deployment all'inzio del file: immettendo le proprie credenziali LDAP al posto di USERNAME e PASSWORD, sarà possibile eseguire direttamente il deploy degli artefatti sul nostro repository Nexus specificando il lifecycle **deploy** durante la build, e.g:

```
mvn deploy
```

Riferimenti esterni

Repository centrale di maven

Documentazione ufficiale di maven

Introduzione ai lifecycle di maven

Sito web di Sonatype Nexus

Sito del repository Nexus 3DI