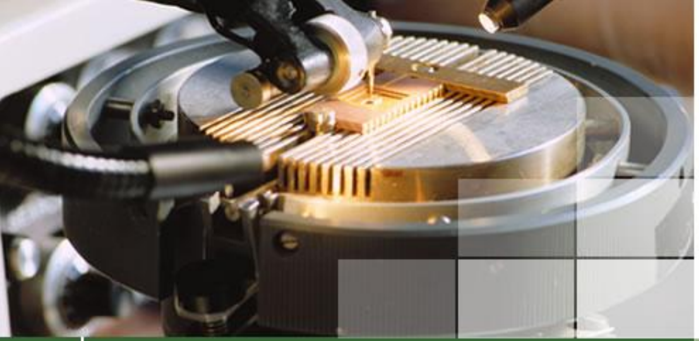


POLITECNICO DI MILANO

Dipartimento di
Elettronica e Informazione

Planning and Managing Software Projects 2011-12

Maven

Emanuele Della Valle, Lecturer: Daniele Dell'Aglio
http://emanueledellavalle.org

What is Maven?

- Maven is a project management tool
  - Build automation tool
  - Deployment tool
  - Documentation generator
  - Report generator
  - ...
- Even if the focus in on Java projects, other programming languages are supported (C#, Ruby, etc.)

Convention

- Maven exploits the «Convention over configuration» approach
  - Assumption about the location of source code, test, resources
  - Assumption about the location of the output
  - Etc.
- In general the conventions can be customized by users
- Example: folder layout
  - src/main/java
  - src/main/resources
  - src/test/java
  - src/test/resources

(http://maven.apache.org/guides/introduction/introduction-to-the-standard-directory-layout.html)

Project Object Model

- The key element is the Project Object Model (POM)
- It is a xml file (pom.xml) located in the root of the project folder
- It contains the description of the project and all the information required by Maven
- POM contains:
  - Project coordinates
  - Dependencies
  - Plugins
  - Repositories
- A POM can be easily generated through

```
mvn archetype:generate
```

Project coordinates

- It is the “passport” of the artifact
- The required information are:
  - Model version: the version of the POM
  - Group id: unique identificator of the group/organization that creates the artifact
  - Artifact id: unique identificator of the artifact
  - Version: the version of the artifact. If the artifact is a non-stable/under development component, -SNAPSHOT should be added after the version number

- Another important information is the packaging
  - It influences the build lifecycles
  - Core packaging: `jar` (default), `pom`, `maven-plugin`, `ejb`, `war`, `ear`, `rar`, `par`

Project coordinates

```
<project>
   <modelVersion>4.0.0</modelVersion>
   <groupId>pmsp.maven</groupId>
   <artifactId>firstexample</artifactId>
   <version>0.1</version>
   <packaging>jar</packaging>
</project>
```

Dependencies

- POM contains the description of the artifacts required by the project
- Each dependency contains
  - The required artifact (identified by its coordinates)
  - The scope: controls the inclusion in the application and the availability in the classpath
    - `compile` (default): required for compilation, in the classpath and packaged
    - `provided`: required for compilation, in the classpath but not packaged (e.g. servlet.jar)
    - `runtime`: required for the execution but not for the compilation (e.g. JDBC, SLF4J bridges)
    - `test`: required only for testing, not packaged (e.g. JUnit)
    - `system`: like provided but with an explicit folder location

Dependencies

- In general the dependency is transitive
  - It depends on the scope! http://maven.apache.org/guides/introduction/introduction-to-dependency-mechanism.html#Dependency_Scope
- If the project requires artifact A, and artifact A requires artifact B, then also artifact B will be included in the project
- Users can disable the transitivity through the exclusion command

Dependencies

```
…
<dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.10</version>
      <scope>test</scope>
    </dependency>
    …
  </dependencies>
…
```

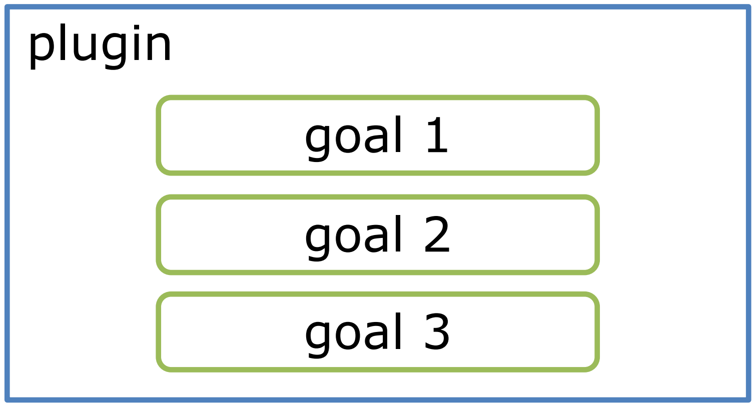
Plugins and Goals

- The Maven core doesn't contain the logic to compile, test, package the software
- Those features are provided through plugins
  - Examples: archetype, jar, compiler, surefire
- As dependencies, projects should declare what are the required plugins

```
...
<plugins>
   <plugin>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.10</version>
   </plugin>
</plugins>
...
```

Goals

- Each plugin offers a set of goals
  - A goal is an executable task



- A goal can be invoked in the following way:

mvn *plugin*:*goal*

- Example:

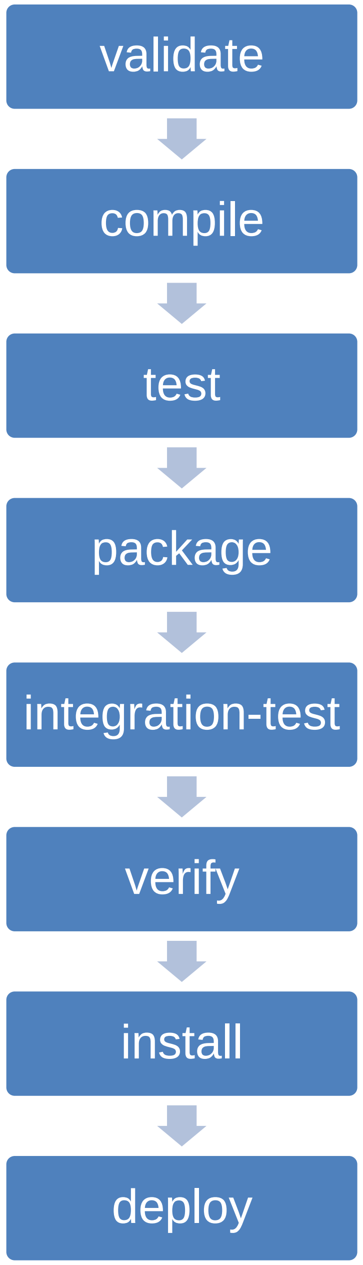
```
mvn archetype:generate
```

Phases and lifecycles

- A phase is a step in the build lifecycle, which is an ordered sequence of phases. [Apache Maven manual]
- **Lifecycles** can be used to execute mutliple operations
  - A lifecycle models a sequence of operations
- Each step of a lifecycle is a **phase**
- When a lifecycle is executed, plugin goals are bound to the phases
  - The goal bindings depends by the packaging!
- Built-in lifecycle
  - `default`
  - `clean`
  - `site`

(http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html)

Lifecycles: default (simplified)



Lifecycles: default (simplified) - packaging jar

validate	
compile	• compiler:compile
test	• surefire:test
package	• jar:jar
integration-test	
verify	
install	• install:install
deploy	• deploy:deploy

Lifecycles: default (simplified) - packaging pom

validate	
compile	
test	
package	• site:attach-descriptor
integration-test	
verify	
install	• install:install
deploy	• deploy:deploy

Repositories

- Repositories are the places where dependencies and plugins are located
  - They are declared into the POM
- When Maven runs, it connects to the repositories in order to retrieve the dependencies and the plugins
- There is a special repository that is built by Maven on the local machine
  - It is located in these locations:
    - Unix: `~/.m2/repository`
    - Win: `C:\%USER%\.m2`
  - Its purpose is twofold:
    - It is a **cache**
    - Artifacts can be **installed** there (`install:install`) and be available for the other artifacts

Repositories

```
<repositories>
    <repository>
        <id>central</id>
        <name>Maven Repository Switchboard</name>
        <layout>default</layout>
        <url>http://repo1.maven.org/maven2</url>
        <snapshots>
            <enabled>false</enabled>
        </snapshots>
    </repository>
    ...
</repositories>
<pluginRepositories>
    <pluginRepository>
        ...
    </pluginRepository>
</pluginRepositories>
```

References

- Maven http://maven.apache.org/guides/
- Apache Maven 2 (Dzone RefCard) http://refcardz.dzone.com/refcardz/apache-maven-2
- Maven: the complete reference (Sonatype) http://www.sonatype.com/books/mvnref-book/reference/
- Maven – The definitive Guide (O’Reilly) http://www.amazon.com/Maven-Definitive-Guide-Sonatype-Company/dp/0596517335