Maven

Bernardo Cuteri

What is Maven?

- Project management tool for Java
- Based on the concept of a **Project Object Model (POM)**
- Handles the building process of a project (from source to program)
- Natively support code reuse and cross-project integration

Maven build life-cycle

Maven is a building tool. Maven adopts a building lifecycle which splits the building process in several phases, the most common are:

- **validate**: validates that the project is correct and all information are available
- **compile**: compiles project sources (into Bytecode for Java sources)
- **test**: tests compiled sources using a unit testing framework (e. g. with JUnit)
- **package**: packages the compiled source code into a distributable package (e.g. as a JAR file)
- **install**: installs the package into the local repository so to be used in other local projects
- **deploy**: copies the package into a remote repository, allowing to share the project to other developers and other external projects

Maven build life-cycle (cont.)

Some other important phases are:

- **site**: for generating the documentation
- **clean**: for cleaning outputs

Maven build life-cycle (cont.)

- The build life-cycle is a chain of phases
- When a phase is executed all phases that precede it in the life-cycle will be executed: for example, the package phase implies the execution of validate, compile and test

Maven keywords

- POM
- Maven Coordinates
- Dependencies
- Archetypes
- Goals e Plugins

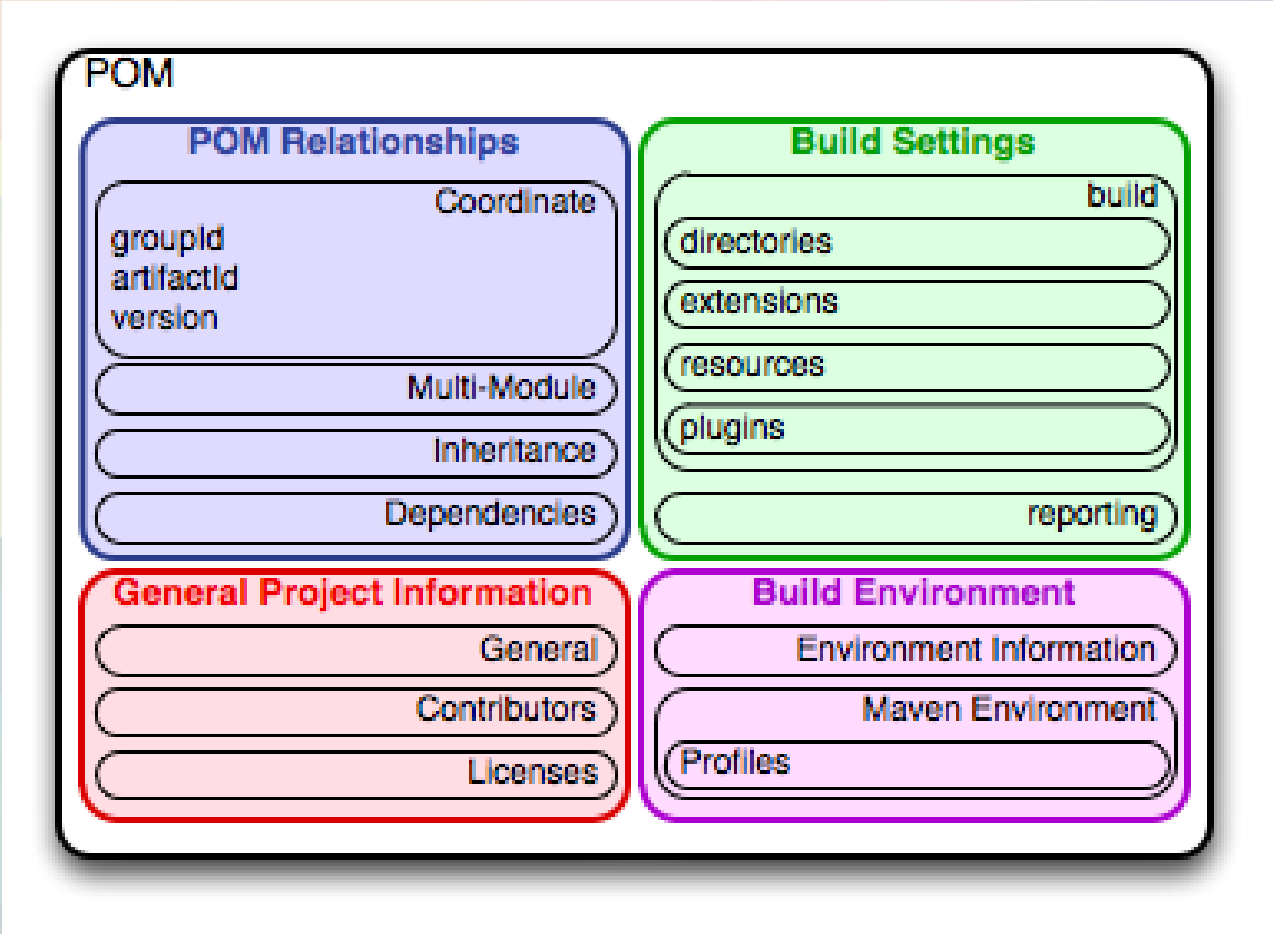
POM

All information related to a Maven project are centralized in the pom.xml file

```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.)

A pom is split in several parts



MAVEN COORDINATES

A Maven project is identified by a triple of values:
<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>
```

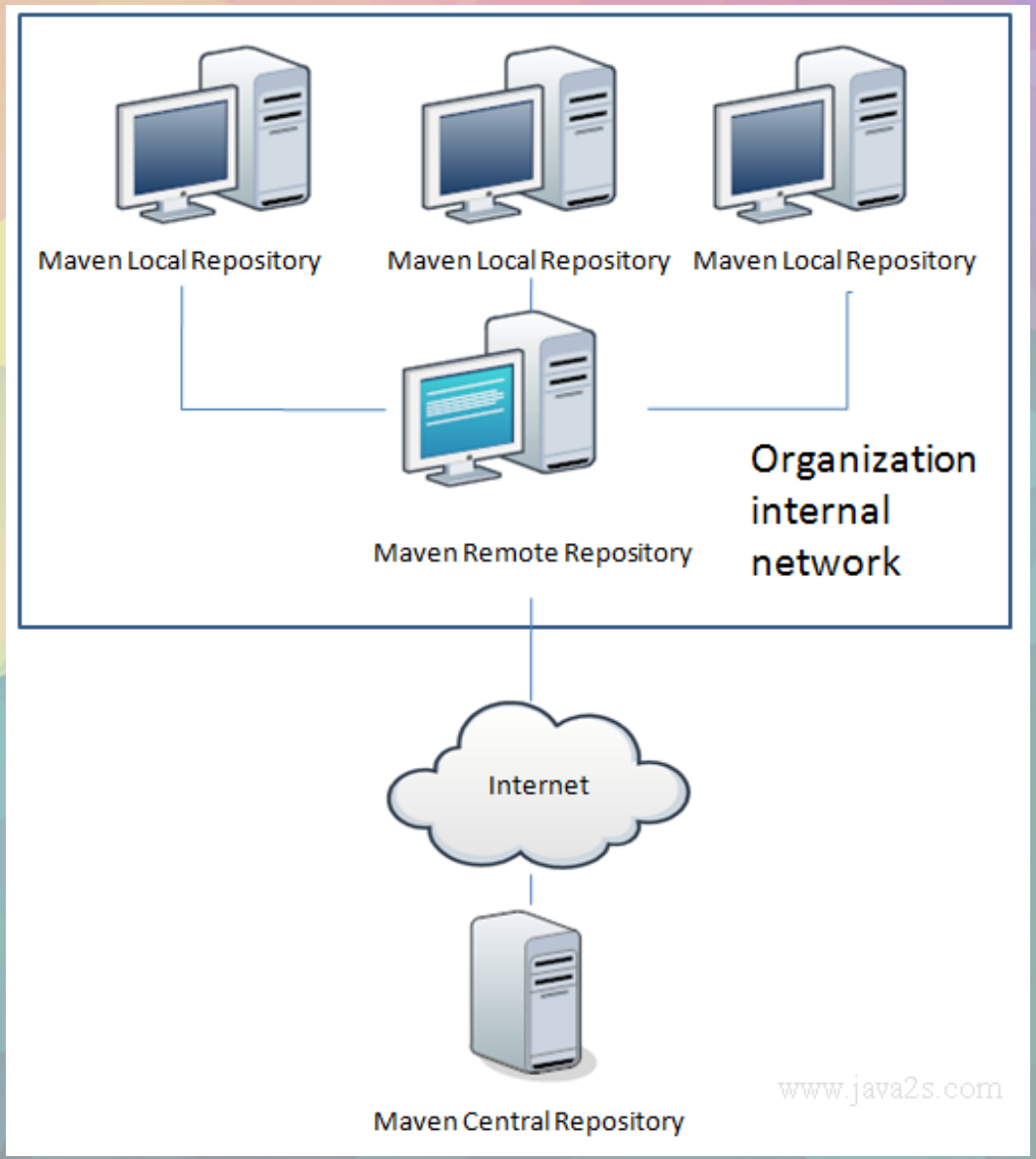
Dependencies

- Maven allows to declare project dependencies declaratively in the pom.xml file
- It is not needed to manually download JAR files and include them in the project
- Maven adopts a repository system
- A repository can be local (.m2 folder), remote or central

Dependencies (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>
```

Maven repositories architecture

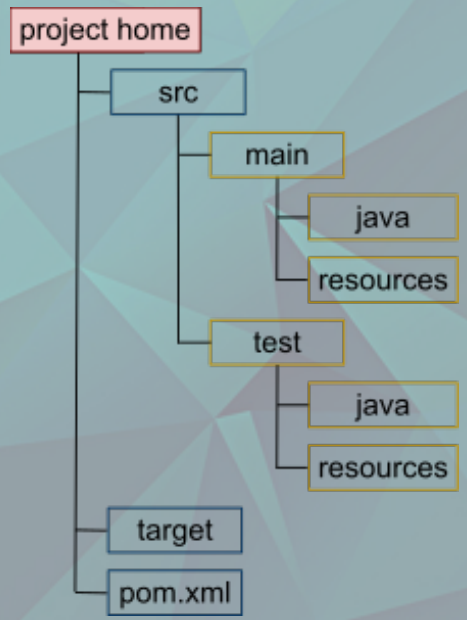


Archetypes

- Archetypes are projects templates from which a programmer can start when creating a new Maven project.
- Command: mvn archetype:generate
- It creates a folder structure and a POM file according to the chosen archetype

Convention Over Configuration

- Maven adopts the so called *Convention Over Configuration* principle
- A convention is a set of default behaviors. Using conventions allows to write standard project with minimal configuration: (i.e. default folder structure is used, naming conventions are assumed and so on)
- Yet, it is still possible to change configuration when needed
- An example of convention is the Maven projects folder structure



Goals e Plugins

- Goals are executable actions in Maven
- Maven build phases are goals: package, test, install etc..
- Goals are provided by Maven artifacts called **Plugins**
- Maven comes with some default plugins that are always included and provide (among other things) the build life-cycle goals
- Other plugins can be added to execute specific goals

Goals e Plugins (cont.)

There are two types of plugins:

- **build plugins**: used during project build phases (compilation, packaging, etc.)
- **reporting plugins**: used for reporting (e.g. during documentation generation phase)