Maven

What is maven?

Maven is a build automation tool mainly used in java projects to manage project dependencies, run tests, compile code and put applications in packages.

What are dependencies?

External libraries that contain elements the project needs to work properly, They add additional third-party functionalities.

Dependency management

When you start a project with maven, it creates a pom.xml file with the details of the project, to preserve and share the metadata.

Maven standardizes the version of the necessary dependencies in the project for all the team members to work with no problem, its plugins keep these dependencies updated to avoid conflicts.

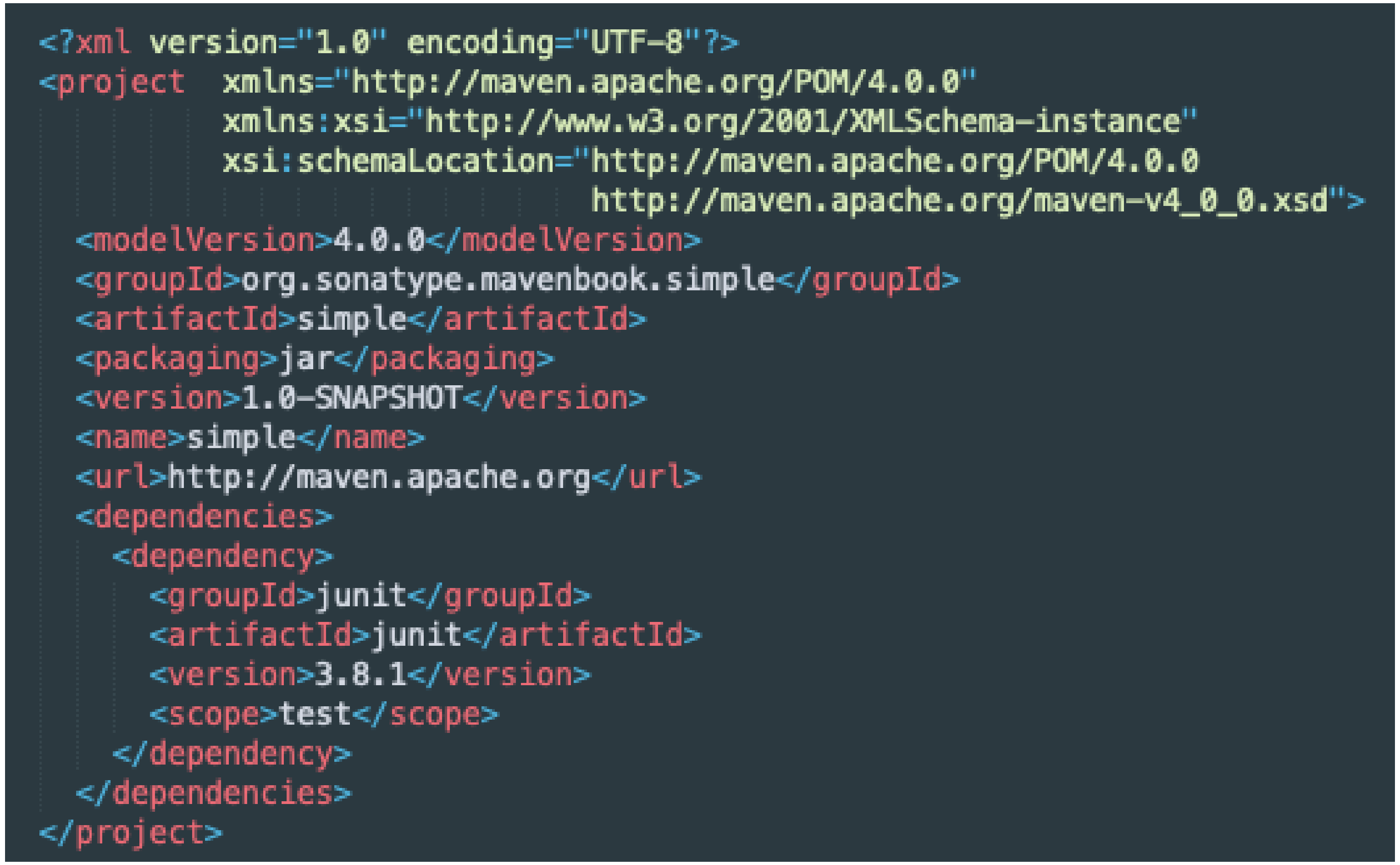
Maven also search for the dependencies of dependencies to make them work properly.

Pom.xml

A pom file contains the data to be used to generate a build, it contains the project information like Version, Team id, project Id, version of project and type of packaging.

The dependencies tag lists the external libraries needed, the build tag contains the configuration of plugins and directories, the repositories tag defines repositories that are used to search for dependencies, the properties tag lets you define custom properties for the file, the profiles tag defines configurations that are activated under certain conditions and the modules tag is used for multimodule projects.

Pom.xml

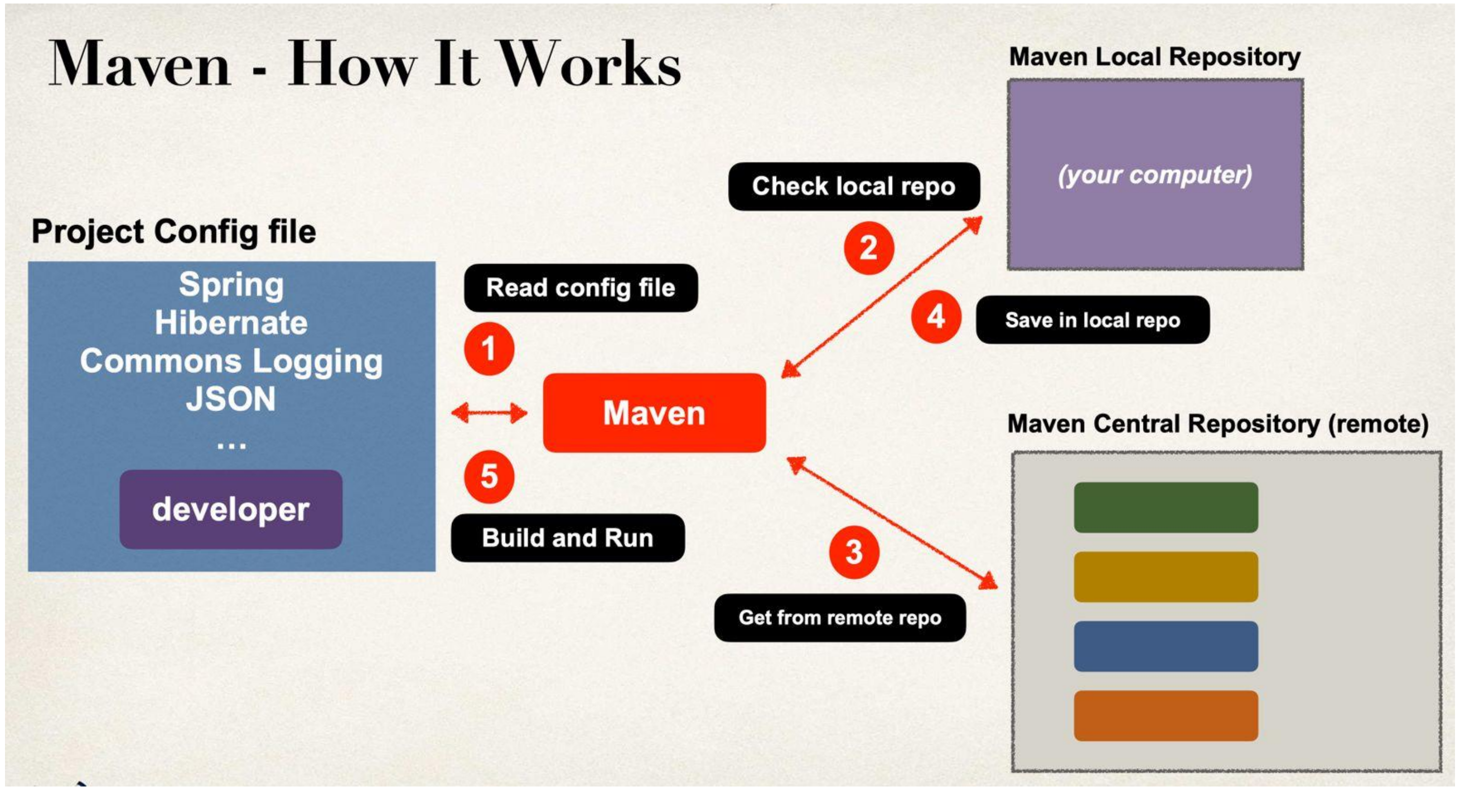


```xml
<?xml version="1.0" encoding="UTF-8"?>
<project  xmlns="http://maven.apache.org/POM/4.0.0"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                    http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>org.sonatype.mavenbook.simple</groupId>
  <artifactId>simple</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>simple</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
</project>
```

Steps to create a build

Maven takes the following steps to create the build of the project:

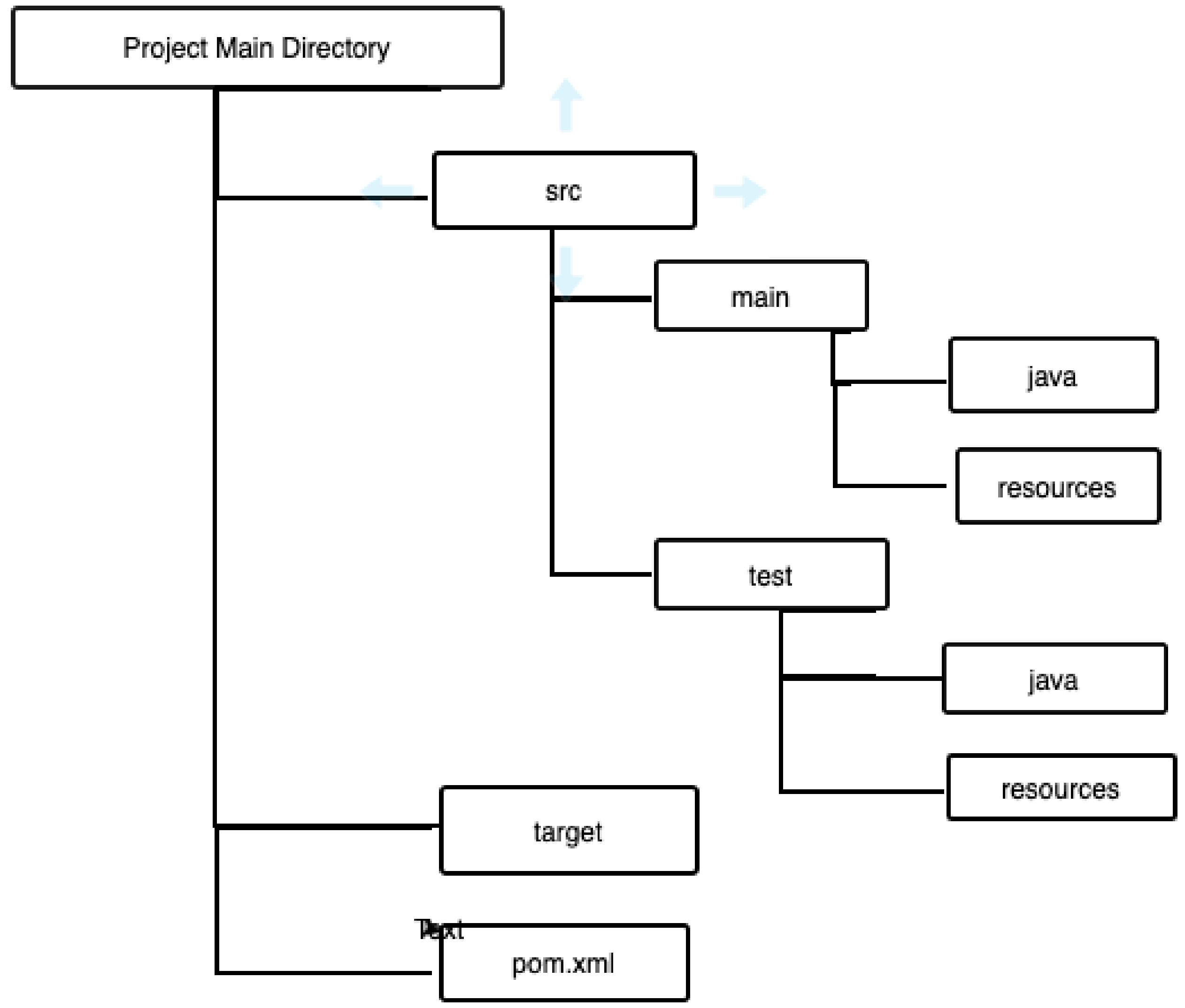
1. Read the configuration file to know what to look for.
2. Check for the dependencies on the local repository.
3. Check for any missing dependencies in remote repositories, also store them in the local repository.
4. Build and run the project.



Standard maven directory

- Pom.xml with the project information.
- Target directory to store the compiled build of the project.
- Src directory which contains:
  - Main directory which contains:
    - Java directory with the source code of the project.
    - Resources directory for the project resources.
  - Test directory which contains:
    - Java directory with the source code of the tests.
    - Resources directory for the test resources.

Standard maven directory



Maven commands

- mvn archetype:generate : Creates a new project from the archetype template, the groupId, artifactId and version of the project need to be specified.
- mvn compile: Compiles the source code and stores the resulting files in target.
- mvn test: Runs the test files in the test directory.
- mvn package: Compiles the project an creates a JAR file with the compiled code and resources in the target directory.
- mvn clean: Removes the files in the target directory.
- mvn dependency:tree: Displays a tree of dependencies showing the libraries the project uses.