Outline

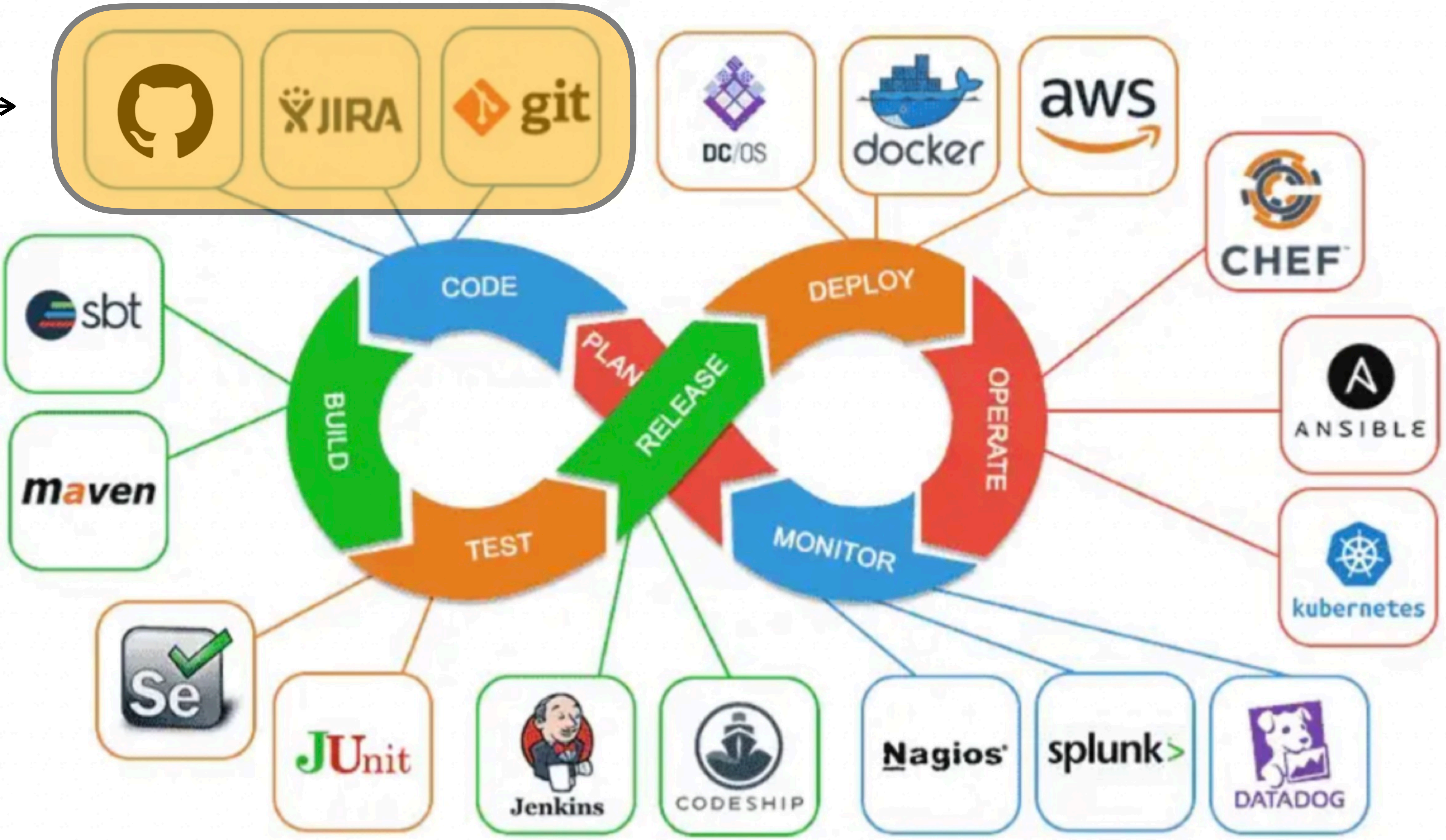
Planned topics for this lesson:

- Introduction to Spring Boot for building Java applications
  What are build tools?
- What is DevOps? Importance of automation
  Deliver software faster with higher quality
- Introduction to CI/CD
  Automation concepts, stages, and workflows

CI/CD Pipeline

Example of a continuous software development system:

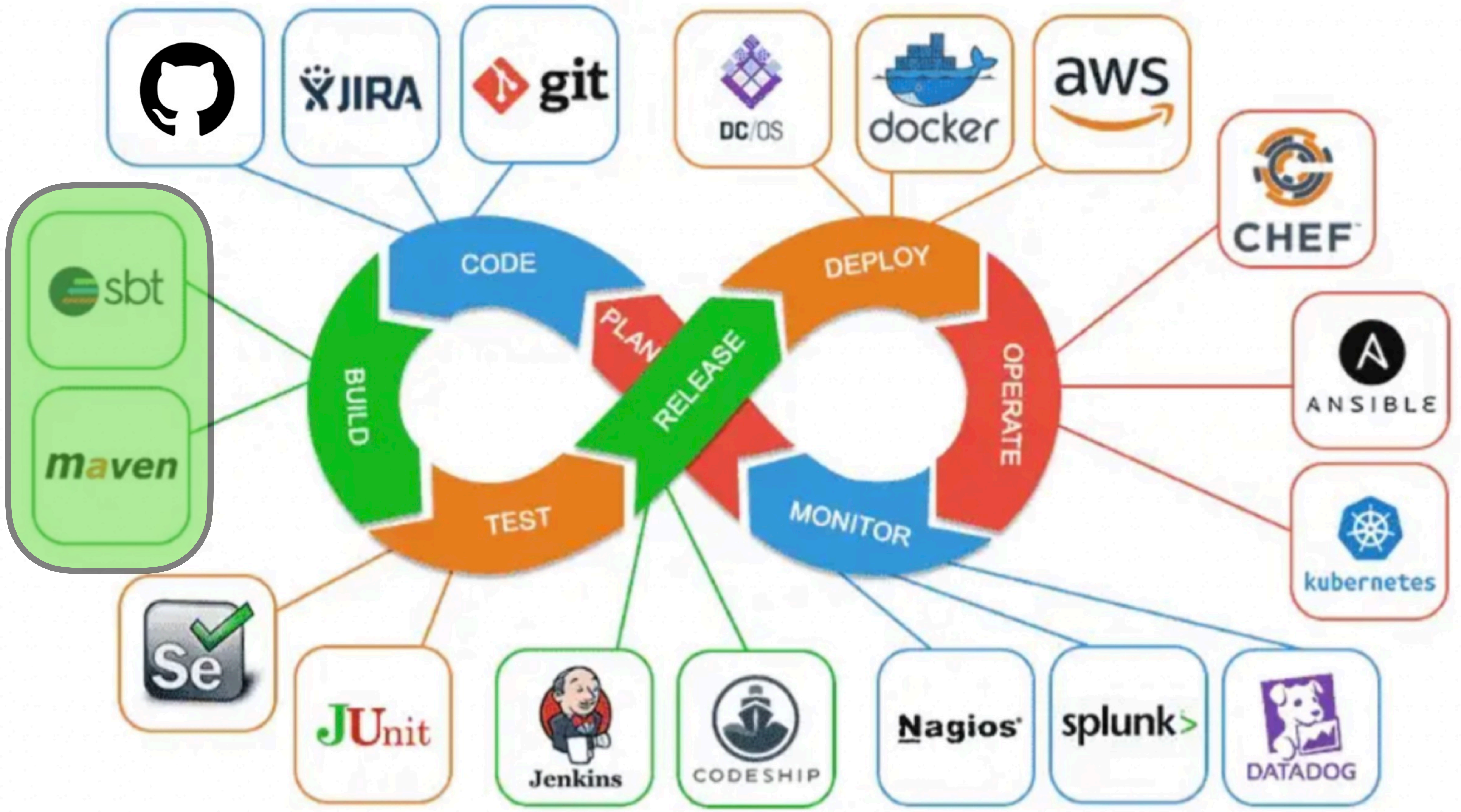
WEEK 1 : VERSION CONTROL →



CI/CD Pipeline

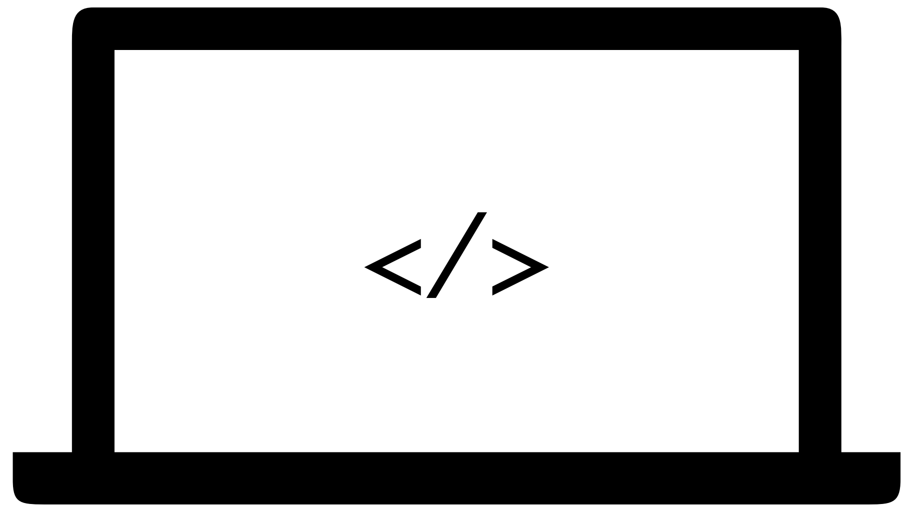
Example of a continuous software development system:

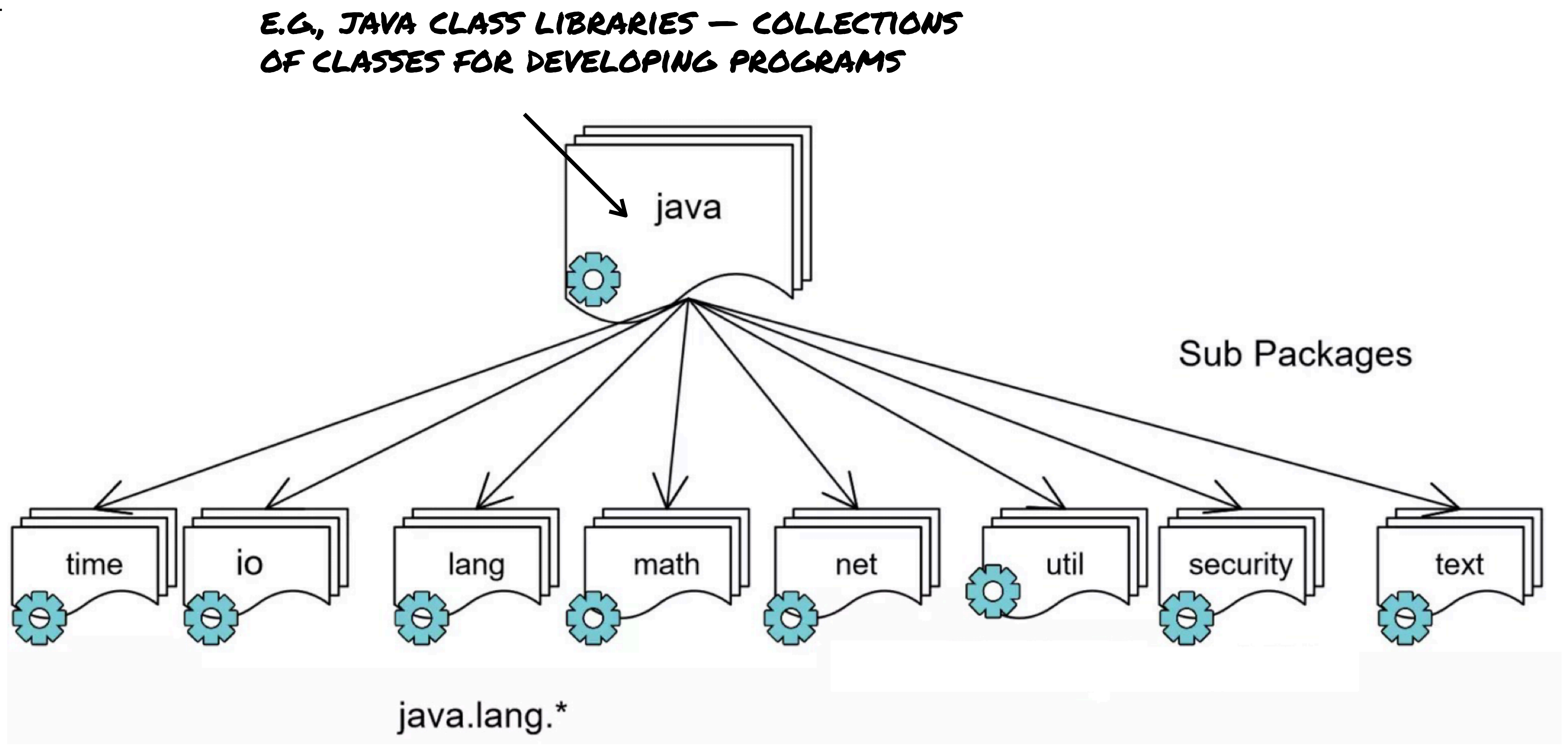
THIS WEEK : BUILD TOOLS →



Code Libraries

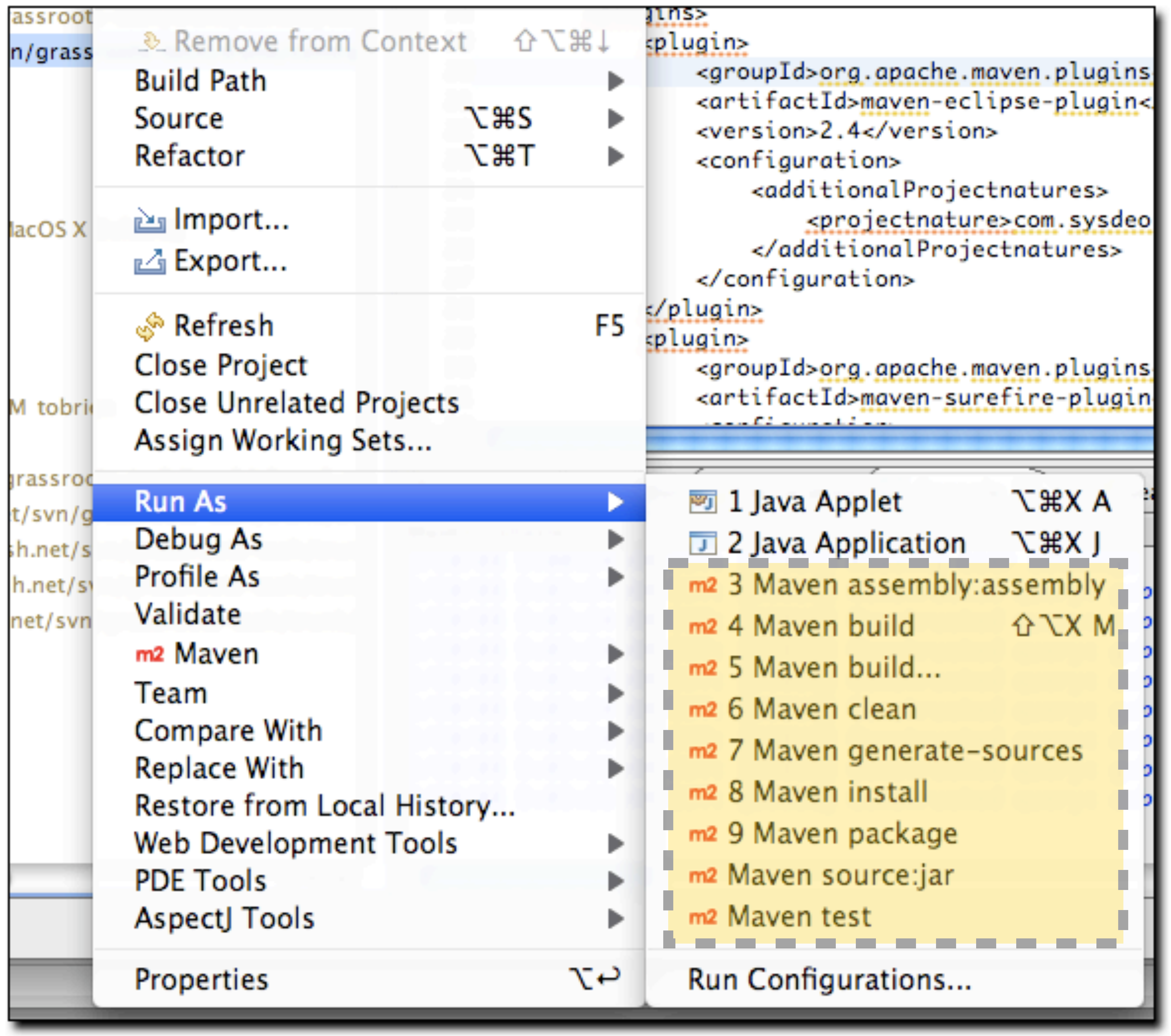
- Code libraries are convenient ways to package functionality and reuse components
  - `JAR` files in Java
  - `.DLL` files Windows / .NET



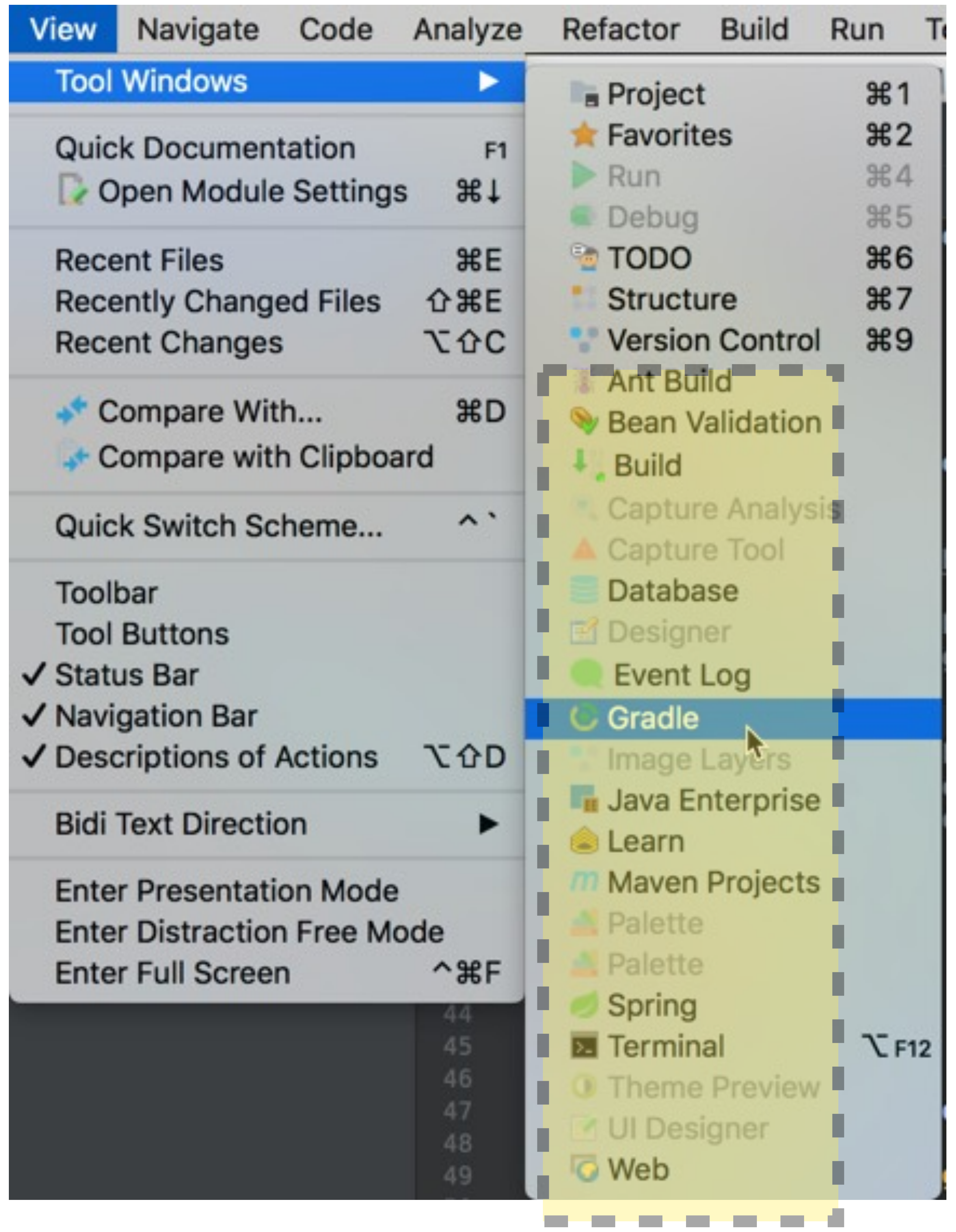

E.G., JAVA CLASS LIBRARIES — COLLECTIONS OF CLASSES FOR DEVELOPING PROGRAMS
java
Sub Packages
time
io
lang
math
net
util
security
text
java.lang.*

Build + Compile + Clean + Run

- IDE has many options, i.e., run-as, build, etc.

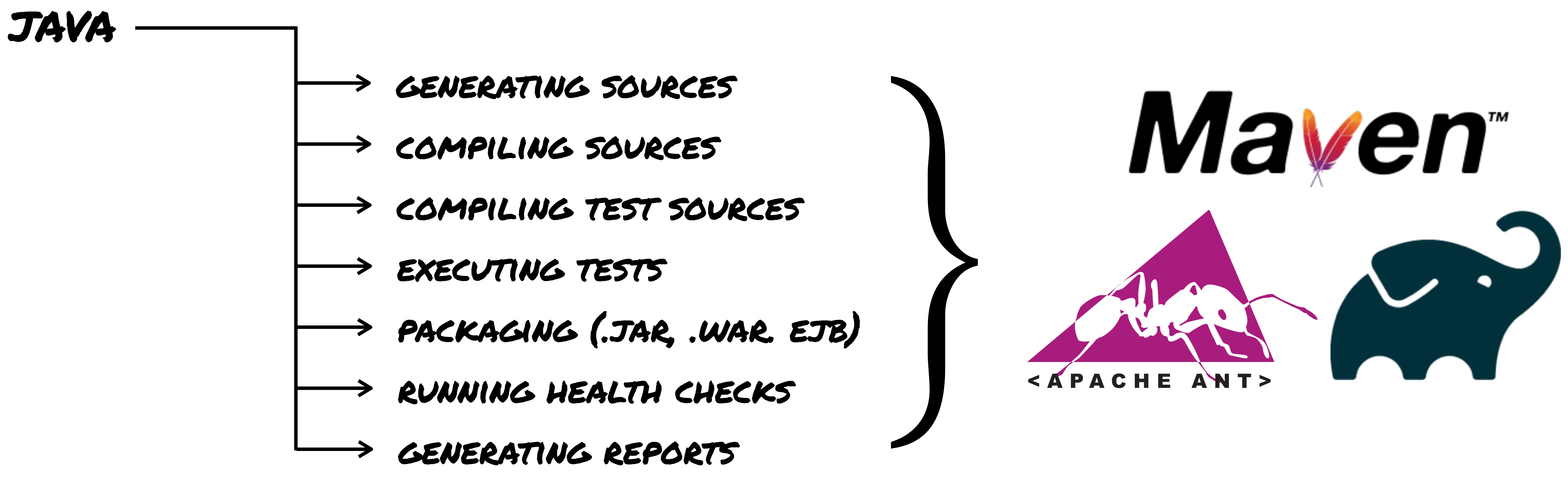


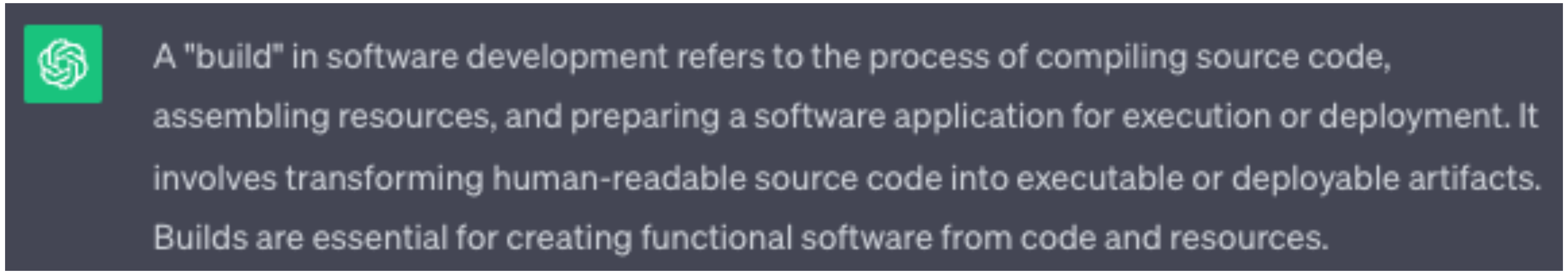
```
ins>
<plugin>
  <groupId>org.apache.maven.plugins
  <artifactId>maven-eclipse-plugin<
  <version>2.4</version>
  <configuration>
    <additionalProjectnatures>
      <projectnature>com.sysdeo
    </additionalProjectnatures>
  </configuration>
</plugin>
<plugin>
  <groupId>org.apache.maven.plugins
  <artifactId>maven-surefire-plugin
```



What is a build?

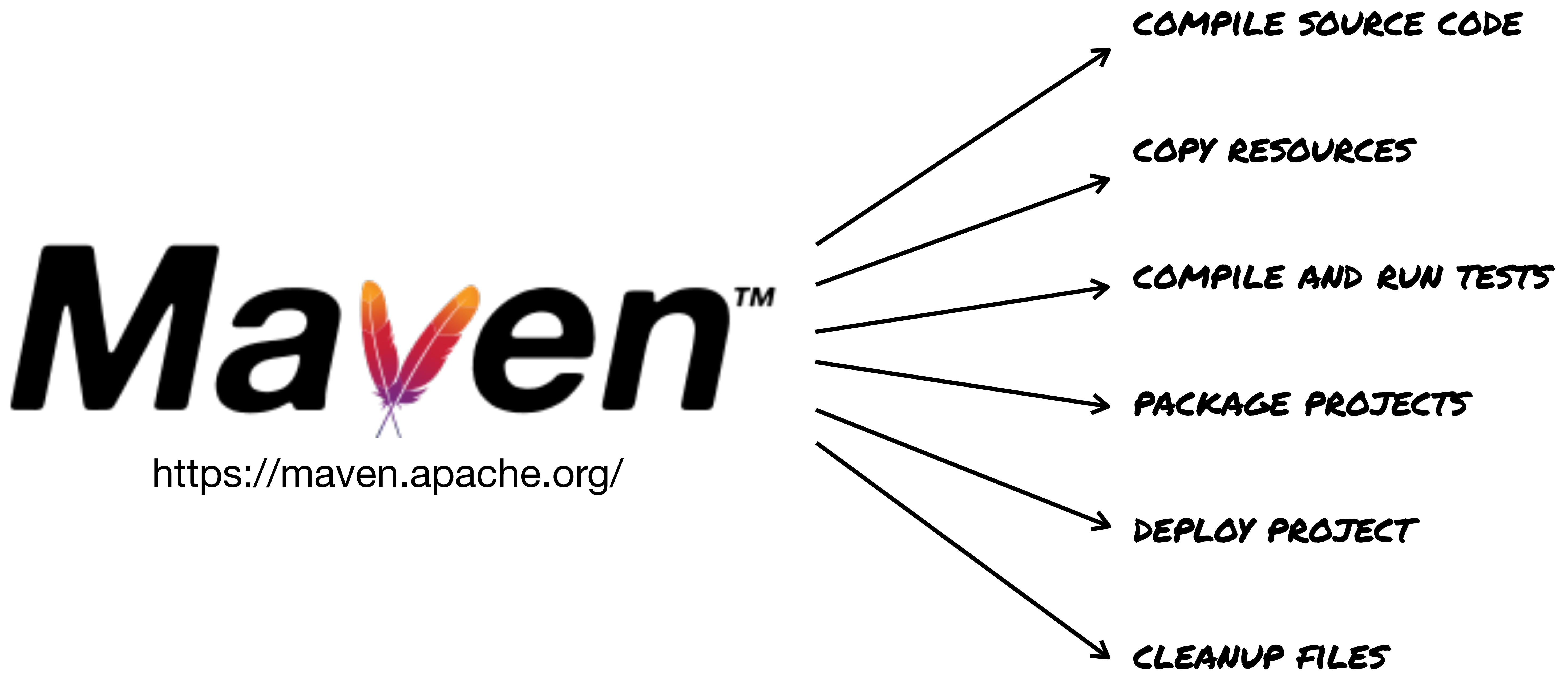
- The build is a process which covers all the steps required to create a deliverable of your software




A "build" in software development refers to the process of compiling source code, assembling resources, and preparing a software application for execution or deployment. It involves transforming human-readable source code into executable or deployable artifacts. Builds are essential for creating functional software from code and resources.

Maven

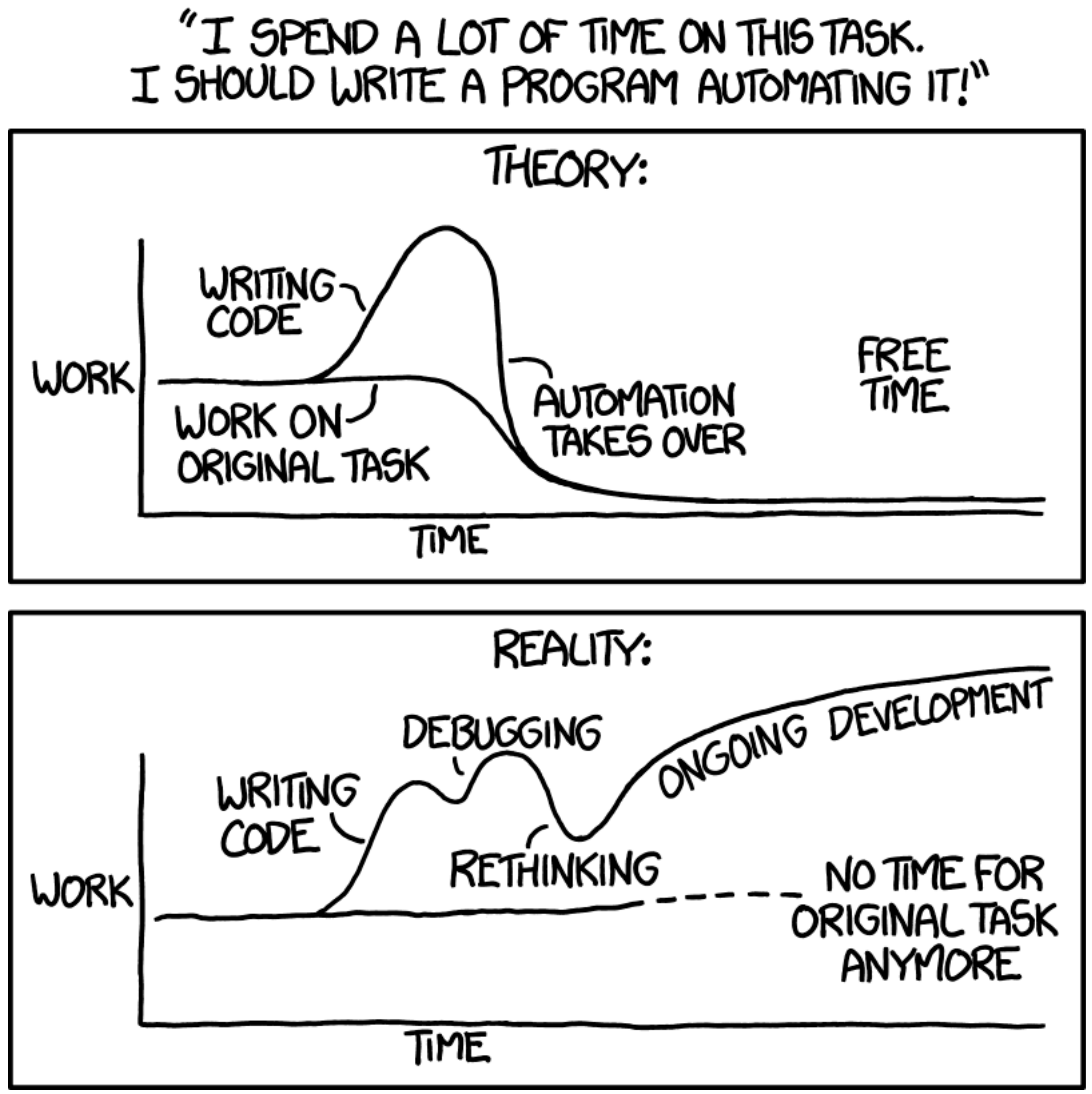
- Software build tool which can manage the project build, reporting and documentation



https://maven.apache.org/

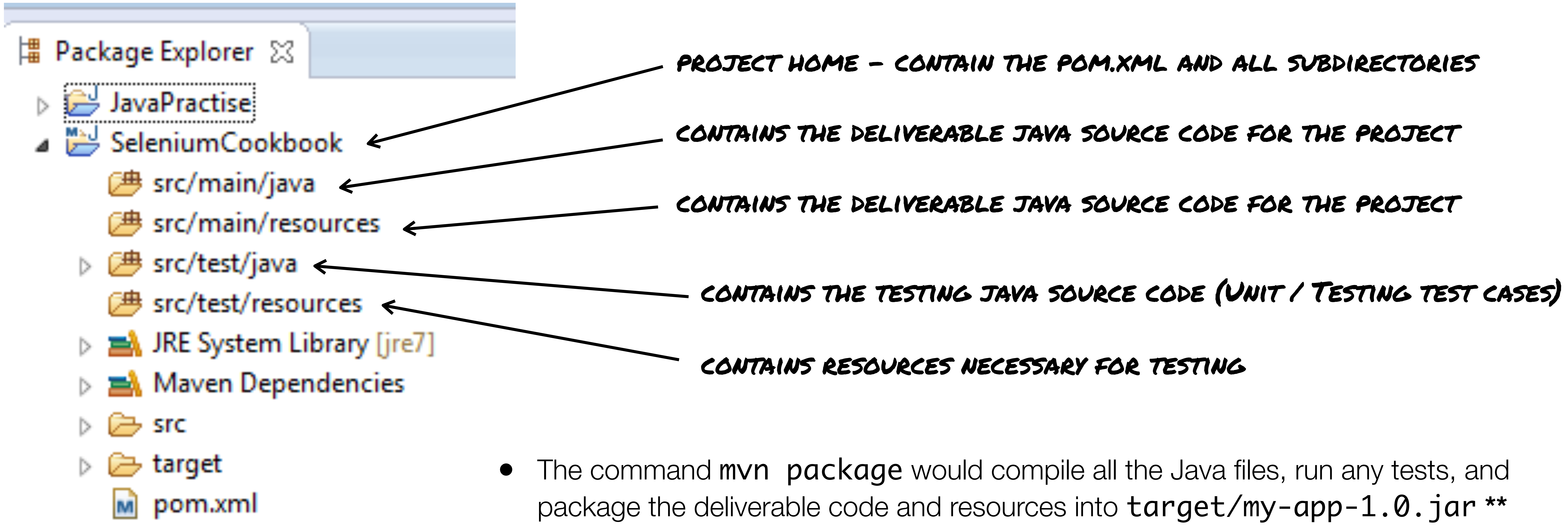
Maven

- Developers wanted:
  - to make the build process easy
  - a standard way to build projects
  - a clear definition of what the project consisted of
  - an easy way to publish project information and a way to share JARs across several projects
- The result is a tool that developers can use to build and manage any Java based project
- It embraces the idea of "convention over configuration"



Maven

Default Directory Structure

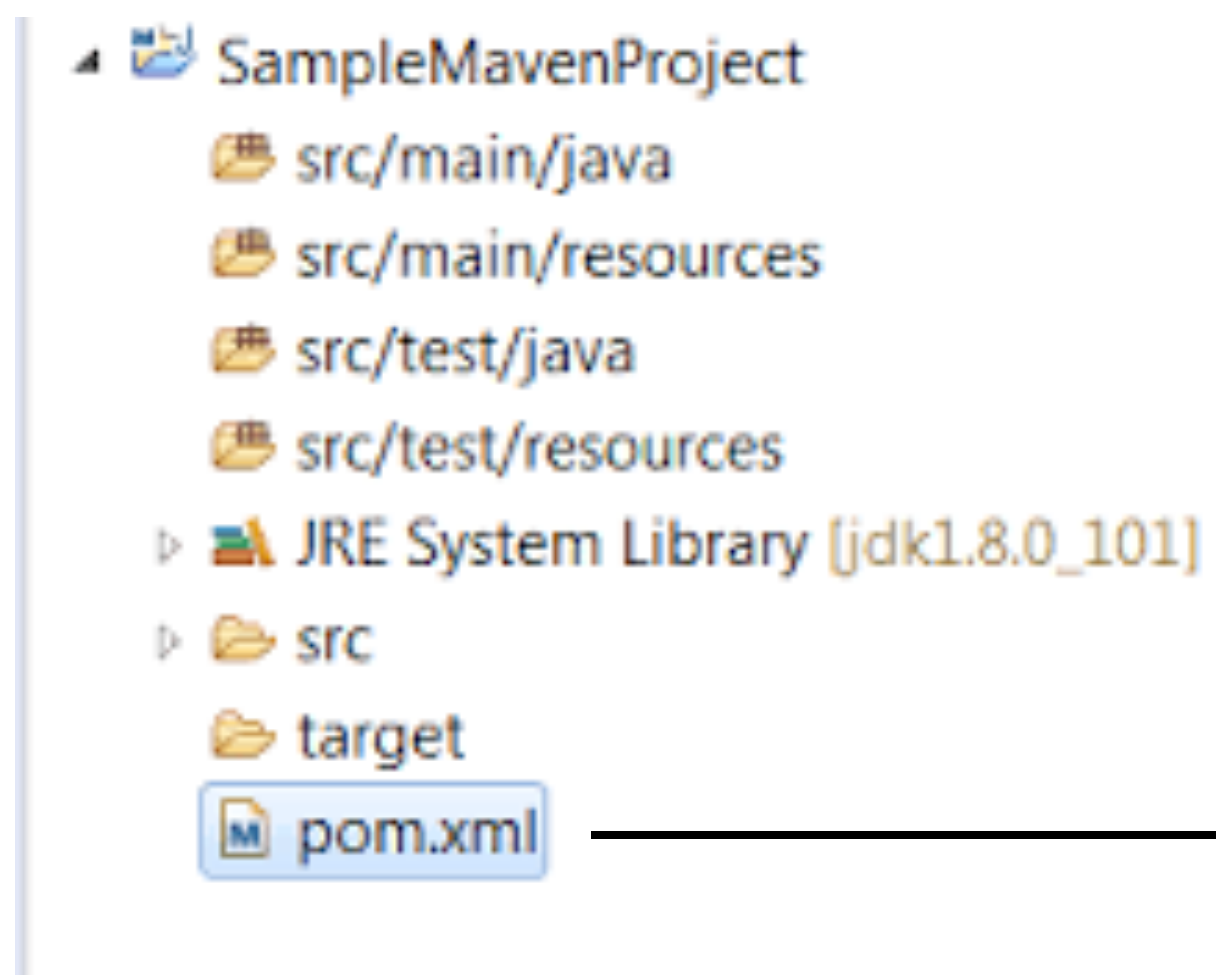

Package Explorer
JavaPractise
SeleniumCookbook
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [jre7]
Maven Dependencies
src
target
pom.xml
PROJECT HOME – CONTAIN THE POM.XML AND ALL SUBDIRECTORIES
CONTAINS THE DELIVERABLE JAVA SOURCE CODE FOR THE PROJECT
CONTAINS THE DELIVERABLE JAVA SOURCE CODE FOR THE PROJECT
CONTAINS THE TESTING JAVA SOURCE CODE (UNIT / TESTING TEST CASES)
CONTAINS RESOURCES NECESSARY FOR TESTING

- The command `mvn package` would compile all the Java files, run any tests, and package the deliverable code and resources into `target/my-app-1.0.jar` **

** `artifactId` is my-app, and the version is 1.0

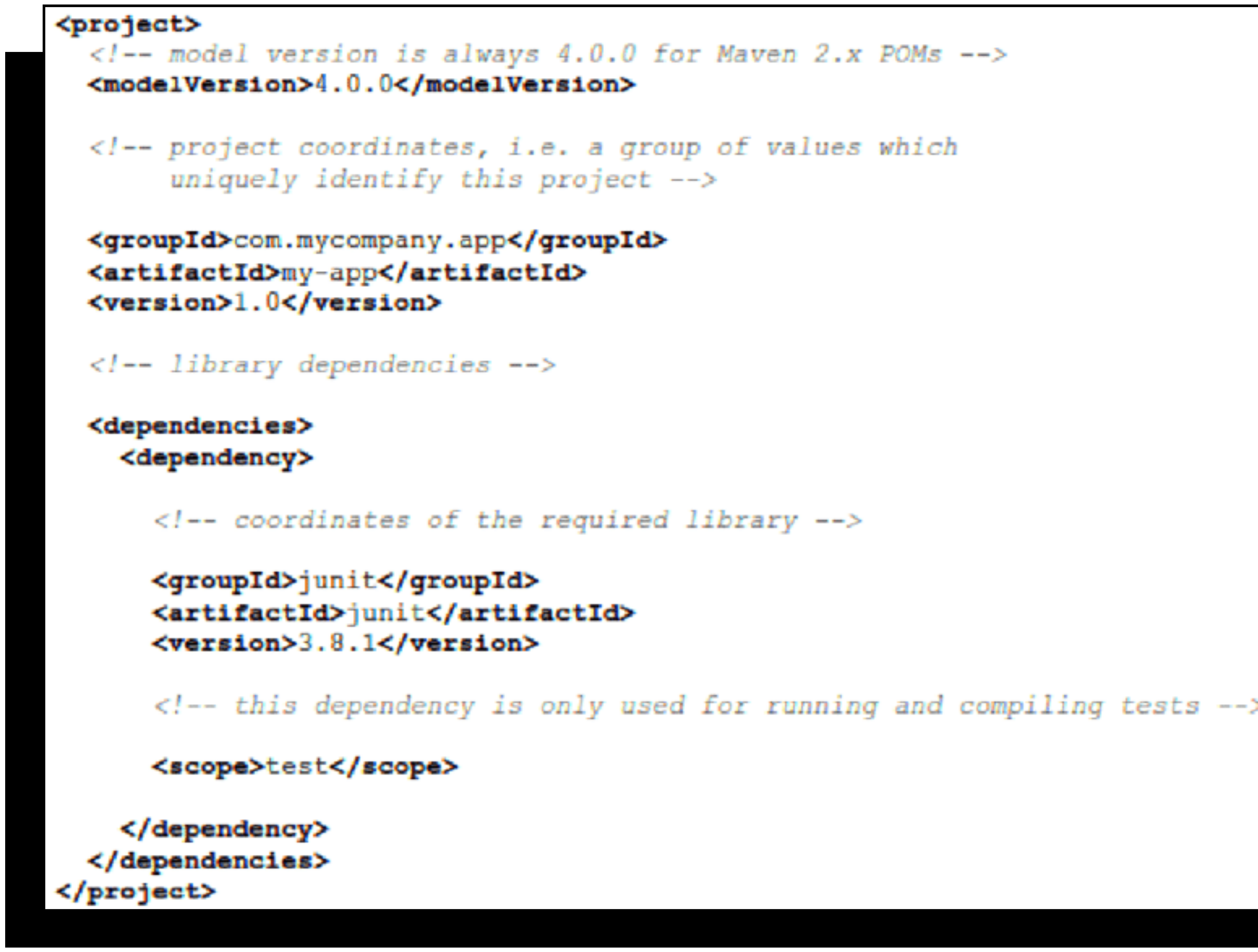
Maven

Project Object Model (POM)



```
SampleMavenProject
    src/main/java
    src/main/resources
    src/test/java
    src/test/resources
    JRE System Library [jdk1.8.0_101]
    src
    target
    pom.xml
```

- All modern IDEs support Maven
- It has a pom.xml as its root — its an CML document
- It contains all the information that Maven requires to automate a build of your software
- It's automatically updated on demand, but can be manually configured as well

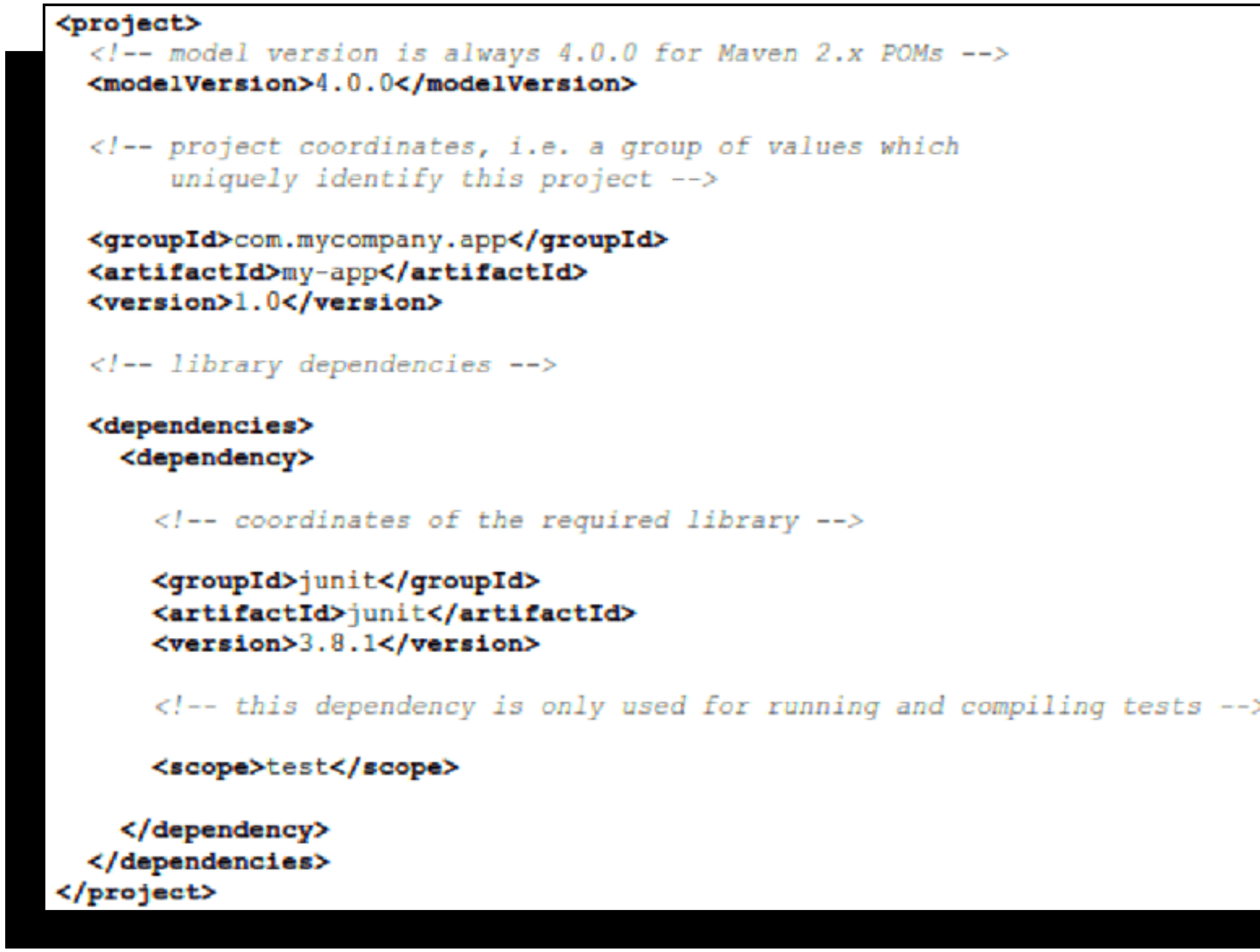


```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- 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

Project Object Model (POM)

- POM provides all the configuration for a single project
  - general configuration covers the project's name, its owner, and its dependencies on other projects
  - One can also configure individual phases of the build process, which are implemented as plugins (e.g., one can configure the compiler-plugin to use Java 1.5 for compilation, or specify packaging the project even if some unit tests fail)
- Larger projects should be divided into several modules, or sub-projects each with its own POM
- All root POM can compile all the modules with a single command
- POMs can also inherit configuration from other POMs

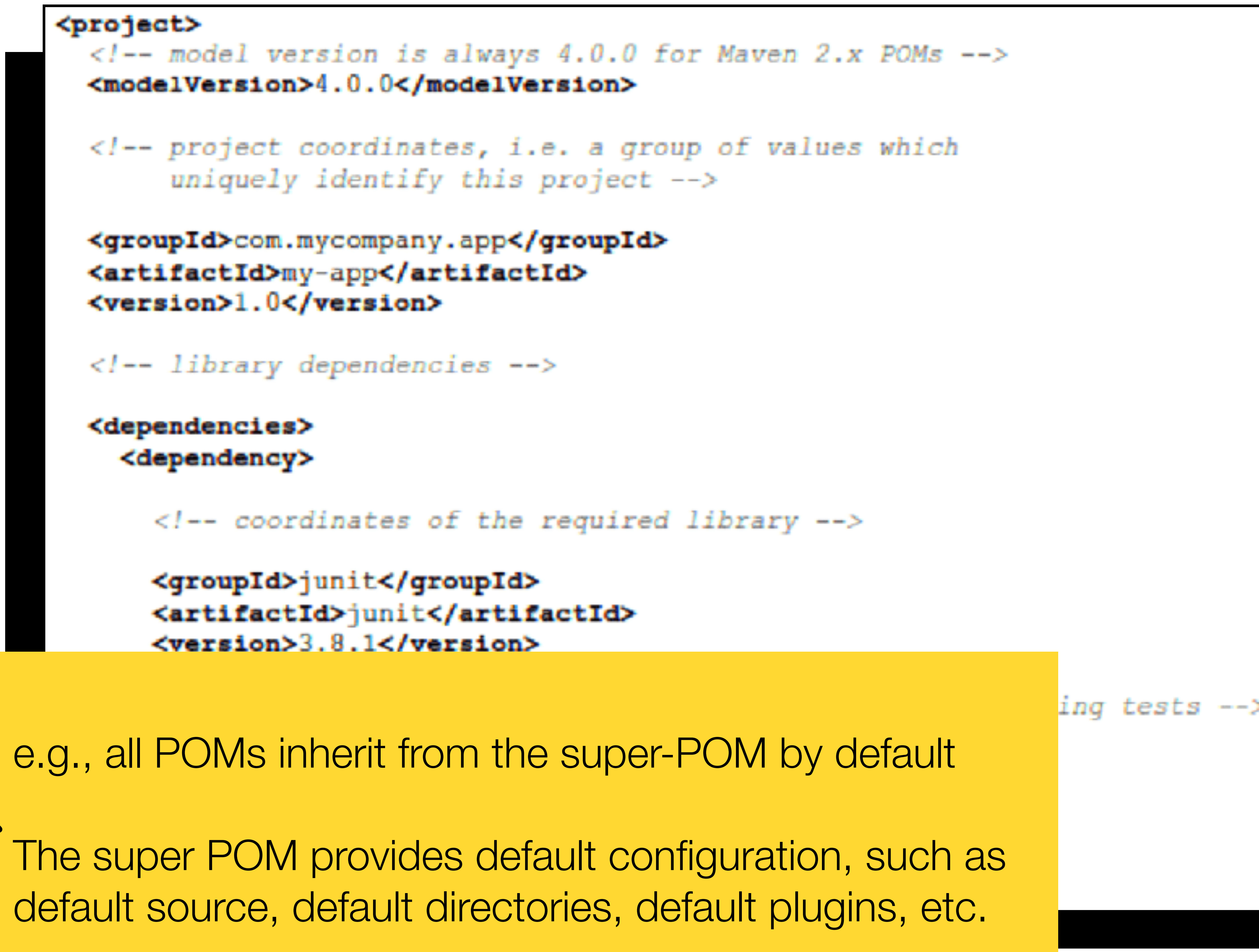


```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- 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

Project Object Model (POM)

- POM provides all the configuration for a single project
  - general configuration covers the project's name, its owner, and its dependencies on other projects
  - One can also configure individual phases of the build process, which are implemented as plugins (e.g., one can configure the compiler-plugin to use Java 1.5 for compilation, or specify packaging the project even if some unit tests fail)
- Larger projects should be divided into several modules, or sub-projects each with its own POM
- All root POM can compile all the modules with a single command
- POMs can also inherit configuration from other POMs



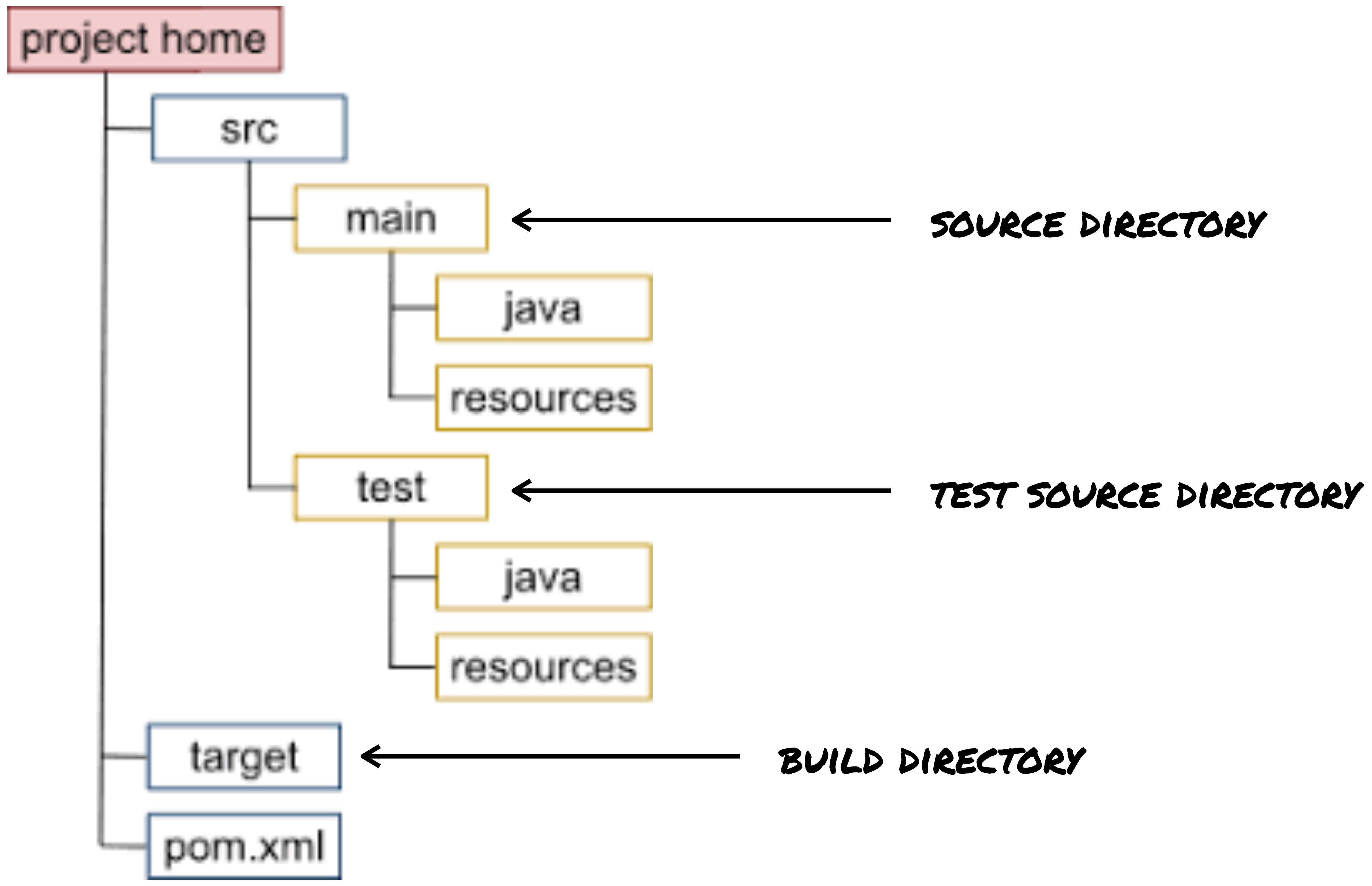
```
<project>
  <!-- model version is always 4.0.0 for Maven 2.x POMs -->
  <modelVersion>4.0.0</modelVersion>

  <!-- 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>
                                                    ing tests -->
```

e.g., all POMs inherit from the super-POM by default

The super POM provides default configuration, such as default source, default directories, default plugins, etc.

Maven

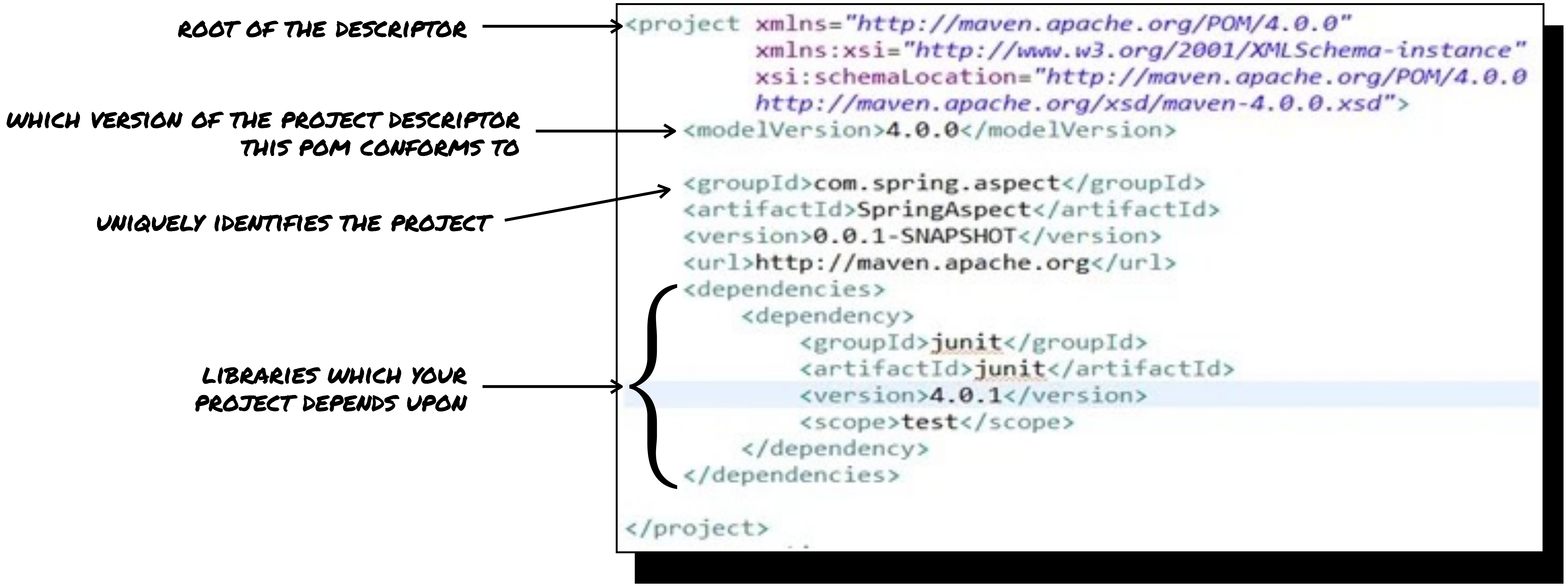
The convention



- Maven builds projects based on convention
  - It expects files to be in a certain place
  - This is very useful when developing in teams !

Maven

pom.xml

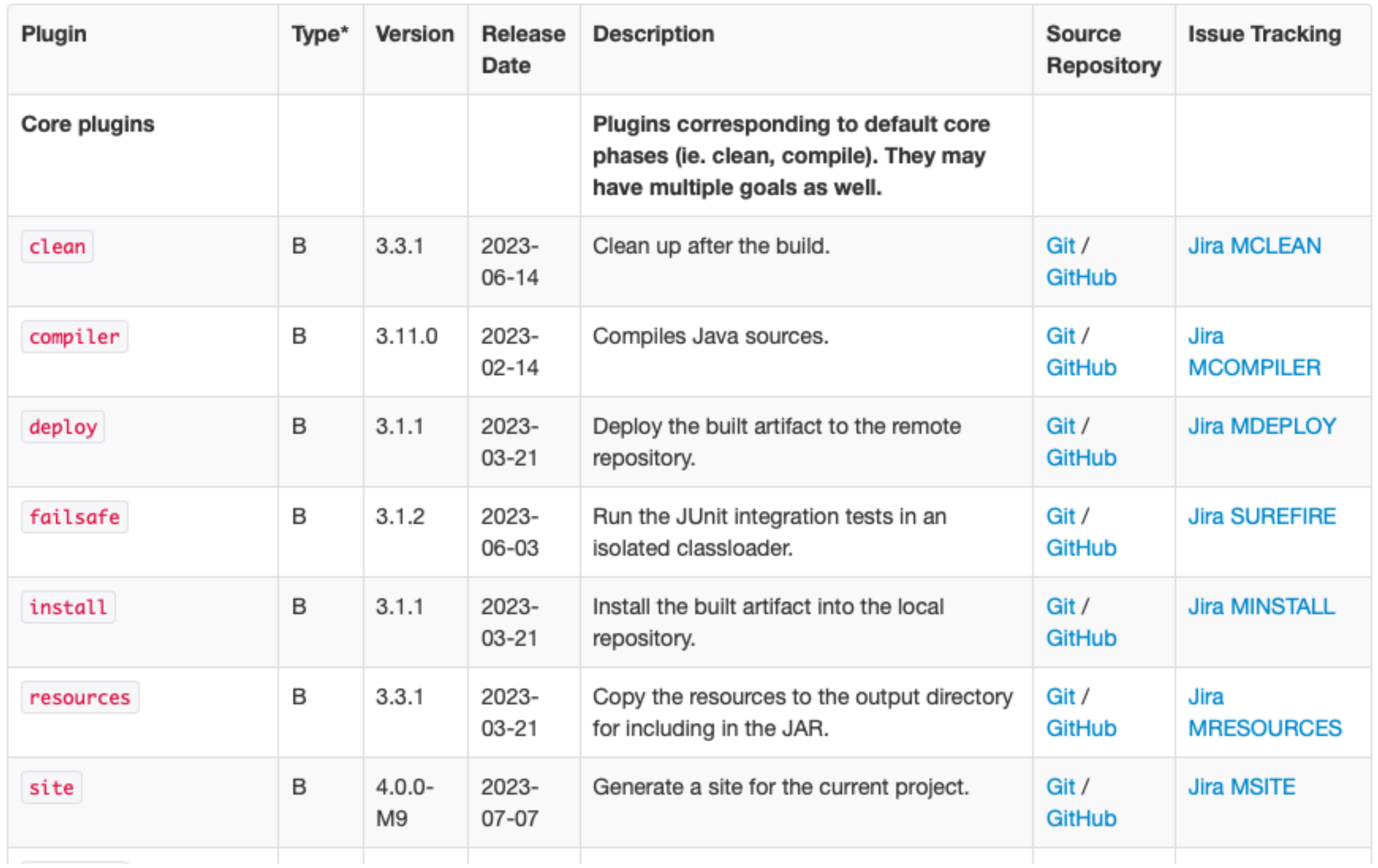

ROOT OF THE DESCRIPTOR
WHICH VERSION OF THE PROJECT DESCRIPTOR THIS POM CONFORMS TO
UNIQUELY IDENTIFIES THE PROJECT
LIBRARIES WHICH YOUR PROJECT DEPENDS UPON

```
<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.spring.aspect</groupId>
    <artifactId>SpringAspect</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <url>http://maven.apache.org</url>
    <dependencies>
        <dependency>
            <groupId>junit</groupId>
            <artifactId>junit</artifactId>
            <version>4.0.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

</project>
```

Maven

pom.xml — plugins

- Maven plugin is an extension or add-on module that enhances the functionality of Apache Maven
- Maven plugins provide additional capabilities and tasks that can be executed during the build process or as part of project lifecycle management.
- These plugins are typically packaged as JAR (Java Archive) files and can be easily added to a Maven project's configuration

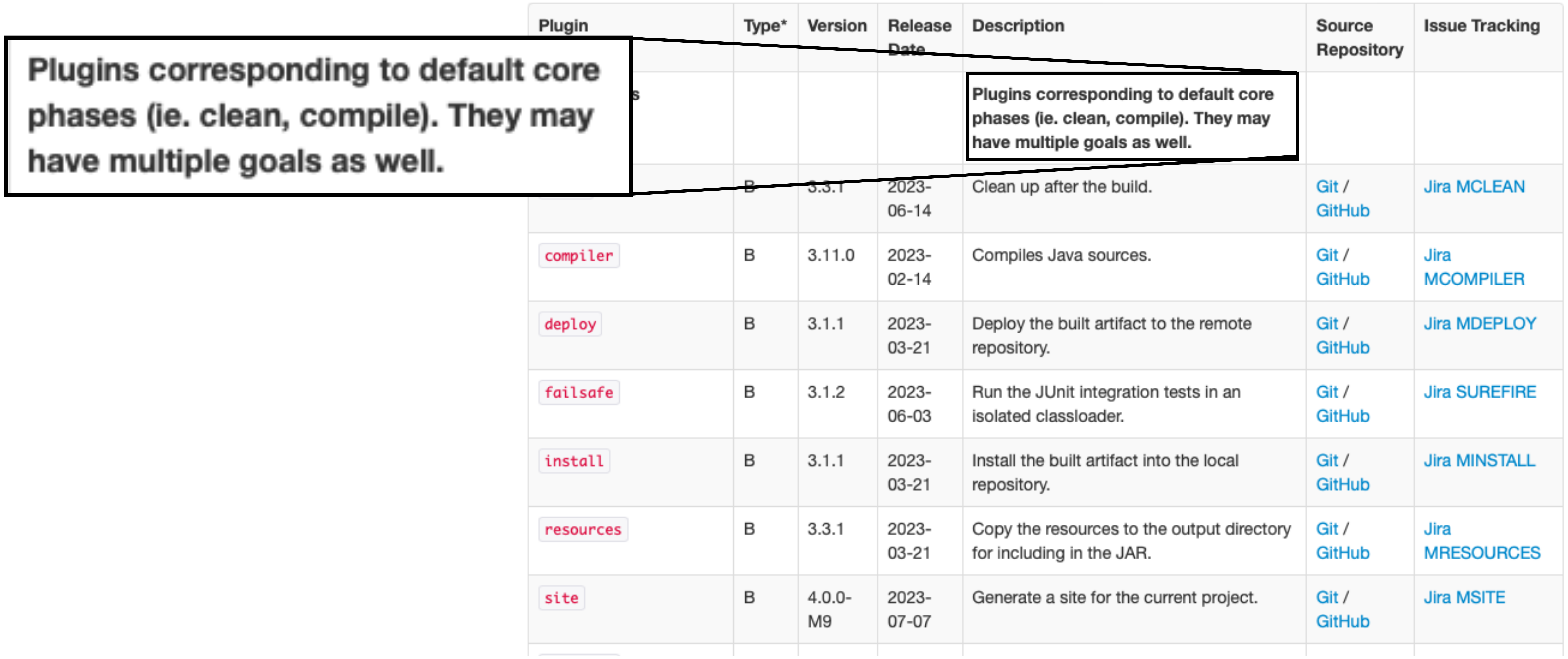


Plugin	Type*	Version	Release Date	Description	Source Repository	Issue Tracking
Core plugins				**Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.**		
`clean`	B	3.3.1	2023-06-14	Clean up after the build.	Git / GitHub	Jira MCLEAN
`compiler`	B	3.11.0	2023-02-14	Compiles Java sources.	Git / GitHub	Jira MCOMPILER
`deploy`	B	3.1.1	2023-03-21	Deploy the built artifact to the remote repository.	Git / GitHub	Jira MDEPLOY
`failsafe`	B	3.1.2	2023-06-03	Run the JUnit integration tests in an isolated classloader.	Git / GitHub	Jira SUREFIRE
`install`	B	3.1.1	2023-03-21	Install the built artifact into the local repository.	Git / GitHub	Jira MINSTALL
`resources`	B	3.3.1	2023-03-21	Copy the resources to the output directory for including in the JAR.	Git / GitHub	Jira MRESOURCES
`site`	B	4.0.0-M9	2023-07-07	Generate a site for the current project.	Git / GitHub	Jira MSITE

https://maven.apache.org/plugins/index.html

Maven

pom.xml — plugins



Plugin	Type*	Version	Release Date	Description	Source Repository	Issue Tracking
s				Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.		
	B	3.3.1	2023-06-14	Clean up after the build.	Git / GitHub	Jira MCLEAN
`compiler`	B	3.11.0	2023-02-14	Compiles Java sources.	Git / GitHub	Jira MCOMPILER
`deploy`	B	3.1.1	2023-03-21	Deploy the built artifact to the remote repository.	Git / GitHub	Jira MDEPLOY
`failsafe`	B	3.1.2	2023-06-03	Run the JUnit integration tests in an isolated classloader.	Git / GitHub	Jira SUREFIRE
`install`	B	3.1.1	2023-03-21	Install the built artifact into the local repository.	Git / GitHub	Jira MINSTALL
`resources`	B	3.3.1	2023-03-21	Copy the resources to the output directory for including in the JAR.	Git / GitHub	Jira MRESOURCES
`site`	B	4.0.0-M9	2023-07-07	Generate a site for the current project.	Git / GitHub	Jira MSITE

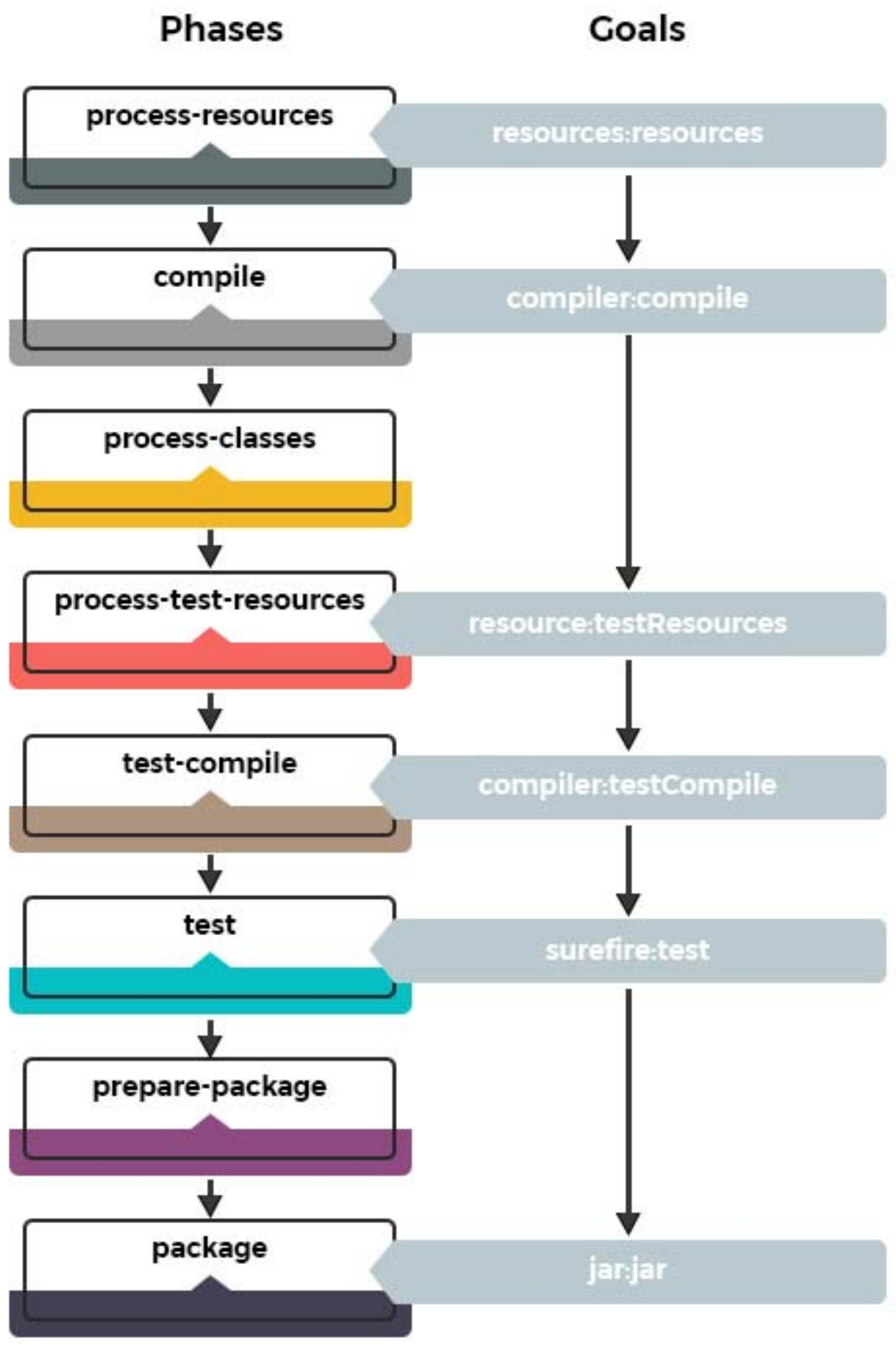
Plugins corresponding to default core phases (ie. clean, compile). They may have multiple goals as well.

https://maven.apache.org/plugins/index.html

Maven

pom.xml — build lifecycle

- Process for building and distributing a particular project is clearly defined
- It comprises of a list of named phases that can be used to give order to goal execution
- Goals provided by plugins can be associated with different phases of the lifecycle
- e.g., by default, the goal compiler:compile is associated with the compile phase, while the goal surefire:test is associated with the test phase
- e.g., man test will cause Maven to run all goals associated with each of the phases up to and including the test phase



Maven

Dependency management

- Central feature in Maven
- The dependency mechanism is organised around a coordinate system identifying individual artefacts such as software libraries or modules (e.g., JUnit)
- If your project depends on a JAR file, Maven will automatically retrieve it for you, and store them in the user'a local repository
- If the JAR file depends on other libraries, Maven will ensure these are also included
  - These are know as transitive dependencies
  - This wasn't always part of Maven, so its huge benefit

- Dependency features supported:
  - Management: you can specify library versions that transitive dependencies should use
  - Scope: include dependencies appropriate for the current stage of the build, i.e., compile, test, run, etc
  - Exclude dependencies: If project X depends on Project Y, and Project Y depends on Project Z, you can choose to exclude Project Z from your build

Maven

Local and remote repository

https://mvnrepository.com/

- Contains libraries for almost everything
  - Cloud computing
  - Date and Time utilities
  - HTML Parsers
  - Mail clients
  - etc
- Once you specify the correct details in you pom file, Maven will automatically get it for you

- Local repository
  - Windows `C:\Users\<username>\.m2`
  - Linux / MacOS `~/.m2`
- Maven will search the local repository first and then move to third party repositories
- You can create your own repository and share it within you company