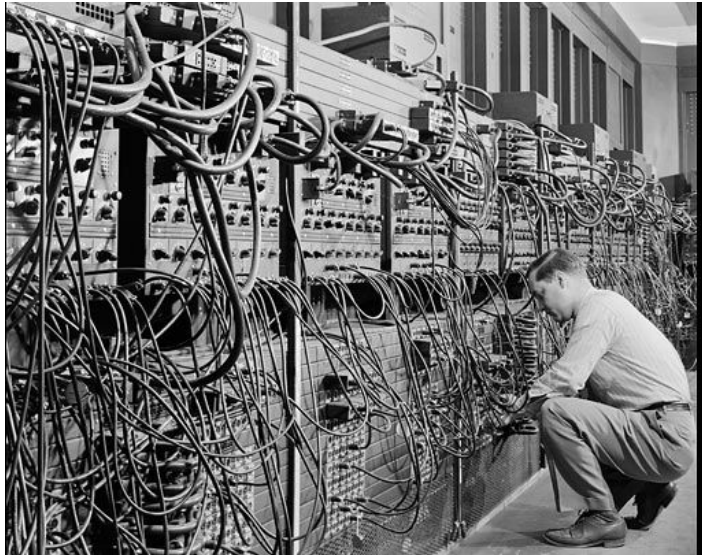
Build tools

History...

of computers...

actually of build tools...



First build tools

GNU Make

<APACHE ANT>

First build tools

```
all: hello

hello: main.o factorial.o hello.o
    g++ main.o factorial.o hello.o -o hello

main.o: main.cpp
    g++ -c main.cpp

factorial.o: factorial.cpp
    g++ -c factorial.cpp

hello.o: hello.cpp
    g++ -c hello.cpp

clean:
    rm *o hello
```

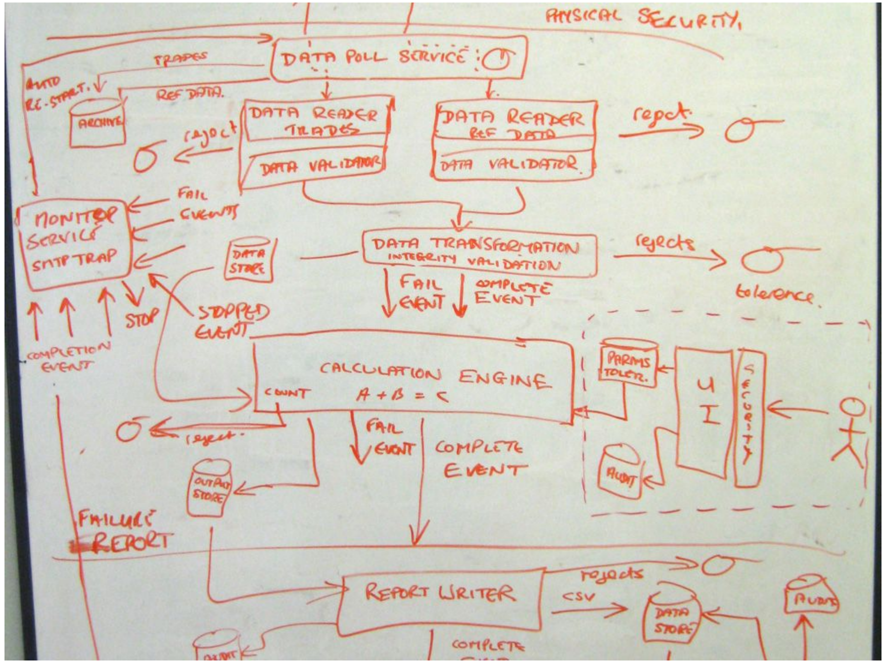
```
<?xml version="1.0"?>
<project name="Hello" default="compile">
    <target name="clean" description="remove intermediate
files">
        <delete dir="classes"/>
    </target>
    <target name="clobber" depends="clean" description="remove
all artifact files">
        <delete file="hello.jar"/>
    </target>
    <target name="compile" description="compile the Java source
code to class files">
        <mkdir dir="classes"/>
        <javac srcdir="." destdir="classes"/>
    </target>
    <target name="jar" depends="compile" description="create a
Jar file for the application">
        <jar destfile="hello.jar">
            <fileset dir="classes" includes="**/*.class"/>
            <manifest>
                <attribute name="Main-Class"
value="HelloProgram"/>
            </manifest>
        </jar>
    </target>
</project>
```

Everyone was happy with <APACHE ANT>

But time passed and …

Present

of software…
and build tools...
and more...



Every environment/build tool is different

- Javascript
  - Gulp, Grunt, Brocolli
- C#/.NET
  - Nant, MSBuild
- Java/JVM
  - Ant, Maven, Gradle, SBT, Leiningen

Java Build Tools

XML Based
+
Flexibility

<APACHE ANT>

2000

XML Based
+
dependency management
+
Reuse

maven

2004

Groovy DSL
+
Hybrid of Ant and Maven

gradle

2012

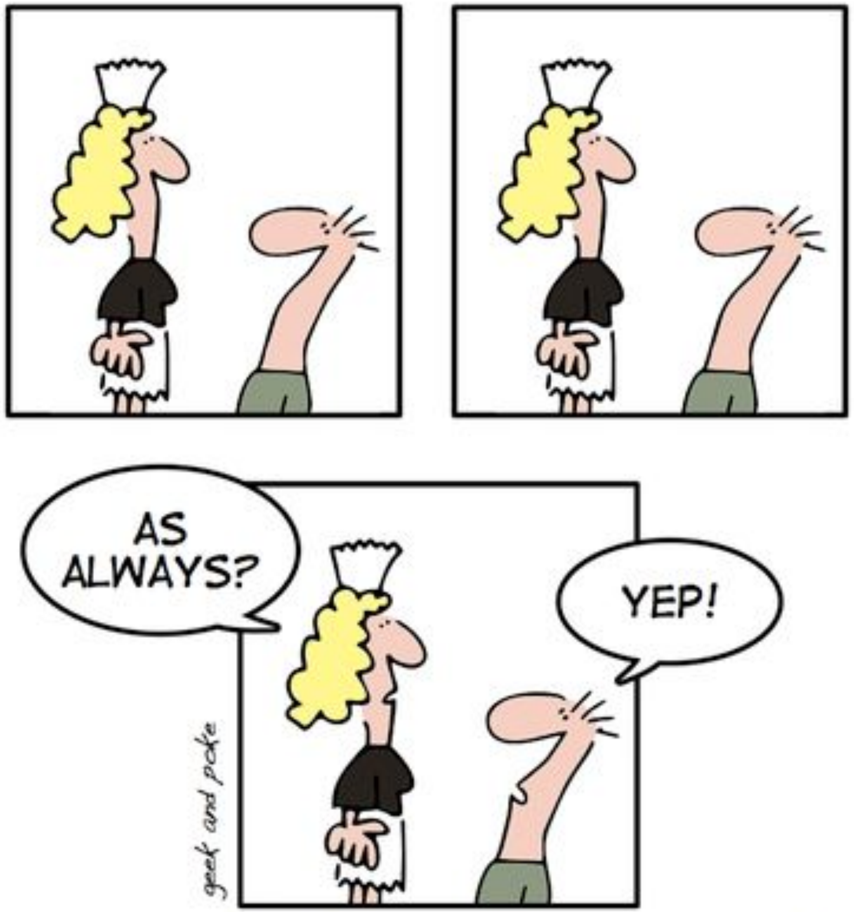
maven

- Convention over configuration
- Dependency management
- Build lifecycle
- Plugins
- Universally reusable
- Common interface
- Standard structure
- Wrapper

convention

---

configuration



SIMPLY EXPLAINED - PART 18:
CONVENTION OVER CONFIGURATION

I love to write a bunch of configuration files before writing actual code

- Said no one ever

```xml
<?xml version="1.0"?>
<project name="simple" default="dist" basedir=".">
  <property name="src" location="src/main/java"/>
  <property name="srcTest" location="src/test/java"/>
  <property name="build" location="build"/>
  <property name="dist" location="${build}/lib"/>
  <property name="version" value="1.0-SNAPSHOT" />
  <path id="classpath.compile">
    <pathelement location="libs/commons-lang-2.5.jar"/
       >
  </path>
  <path id="classpath.test">
    <pathelement location="libs/junit-4.8.2.jar"/>
    <pathelement location="libs/commons-lang-2.5.jar"/
       >
    <pathelement location="${srcTest}"/>
    <pathelement location="${build}/classes"/>
    <pathelement location="${build}/test-classes"/>
  </path>
  <target name="init">
    <mkdir dir="${build}/classes"/>
    <mkdir dir="${build}/test-classes"/>
  </target>
  <target name="compile" depends="init">
    <javac srcdir="${src}" destdir="${build}/classes">
      <classpath refid="classpath.compile"/>
    </javac>
  </target>
  <target name="testCompile" depends="compile">
    <javac srcdir="${srcTest}" destdir="${build}/test-
       classes">
      <classpath refid="classpath.test"/>
    </javac>
  </target>
  <target name="test" depends="testCompile">
    <junit fork="yes" haltonfailure="yes">
      <batchtest fork="yes">
        <fileset dir="${srcTest}">
          <include name="**/*Test.java"/>
        </fileset>
      </batchtest>
      <classpath refid="classpath.test"/>
      <formatter type="plain"/>
    </junit>
  </target>
  <target name="dist" depends="test">
    <mkdir dir="${dist}"/>
    <jar jarfile="${dist}/coc-comparison-$
       {version}.jar" basedir="${build}/classes"/>
  </target>
  <target name="clean">
    <delete dir="${build}"/>
  </target>
</project>
```

<APACHE ANT>

Convention
Over
Configuration

Tomek's Blog
http://kaczanowscy.pl/tomek

```xml
<?xml version="1.0"?>
<project name="simple" default="dist" basedir=".">
  <property name="src" location="src/main/java"/>
  <property name="srcTest" location="src/test/java"/>
  <property name="build" location="build"/>
  <property name="dist" location="${build}/lib"/>
  <property name="version" value="1.0-SNAPSHOT" />
  <path id="classpath.compile">
    <pathelement location="libs/commons-lang-2.5.jar"/
       >
  </path>
  <path id="classpath.test">
    <pathelement location="libs/junit-4.8.2.jar"/>
    <pathelement location="libs/commons-lang-2.5.jar"/
       >
    <pathelement location="${srcTest}"/>
    <pathelement location="${build}/classes"/>
    <pathelement location="${build}/test-classes"/>
  </path>
  <target name="init">
    <mkdir dir="${build}/classes"/>
    <mkdir dir="${build}/test-classes"/>
  </target>
  <target name="compile" depends="init">
    <javac srcdir="${src}" destdir="${build}/classes">
      <classpath refid="classpath.compile"/>
    </javac>
  </target>
  <target name="testCompile" depends="compile">
    <javac srcdir="${srcTest}" destdir="${build}/test-
       classes">
      <classpath refid="classpath.test"/>
    </javac>
  </target>
  <target name="test" depends="testCompile">
    <junit fork="yes" haltonfailure="yes">
      <batchtest fork="yes">
        <fileset dir="${srcTest}">
          <include name="**/*Test.java"/>
        </fileset>
      </batchtest>
      <classpath refid="classpath.test"/>
      <formatter type="plain"/>
    </junit>
  </target>
  <target name="dist" depends="test">
    <mkdir dir="${dist}"/>
    <jar jarfile="${dist}/coc-comparison-$
       {version}.jar" basedir="${build}/classes"/>
  </target>
  <target name="clean">
    <delete dir="${build}"/>
  </target>
</project>
```

<APACHE ANT>

```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>grId</groupId>
  <artifactId>coc-comparison</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
      <version>2.5</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-
       plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

maven

Convention
Over
Configuration

```xml
<?xml version="1.0"?>
<project name="simple" default="dist" basedir=".">
  <property name="src" location="src/main/java"/>
  <property name="srcTest" location="src/test/java"/>
  <property name="build" location="build"/>
  <property name="dist" location="${build}/lib"/>
  <property name="version" value="1.0-SNAPSHOT" />
  <path id="classpath.compile">
    <pathelement location="libs/commons-lang-2.5.jar"/
        >
  </path>
  <path id="classpath.test">
    <pathelement location="libs/junit-4.8.2.jar"/>
    <pathelement location="libs/commons-lang-2.5.jar"/
        >
    <pathelement location="${srcTest}"/>
    <pathelement location="${build}/classes"/>
    <pathelement location="${build}/test-classes"/>
  </path>
  <target name="init">
    <mkdir dir="${build}/classes"/>
    <mkdir dir="${build}/test-classes"/>
  </target>
  <target name="compile" depends="init">
    <javac srcdir="${src}" destdir="${build}/classes">
      <classpath refid="classpath.compile"/>
    </javac>
  </target>
  <target name="testCompile" depends="compile">
    <javac srcdir="${srcTest}" destdir="${build}/test-
        classes">
      <classpath refid="classpath.test"/>
    </javac>
  </target>
  <target name="test" depends="testCompile">
    <junit fork="yes" haltonfailure="yes">
      <batchtest fork="yes">
        <fileset dir="${srcTest}">
          <include name="**/*Test.java"/>
        </fileset>
      </batchtest>
      <classpath refid="classpath.test"/>
      <formatter type="plain"/>
    </junit>
  </target>
  <target name="dist" depends="test">
    <mkdir dir="${dist}"/>
    <jar jarfile="${dist}/coc-comparison-$
        {version}.jar" basedir="${build}/classes"/>
  </target>
  <target name="clean">
    <delete dir="${build}"/>
  </target>
</project>
```

<APACHE ANT>

```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>grId</groupId>
  <artifactId>coc-comparison</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <dependencies>
    <dependency>
      <groupId>commons-lang</groupId>
      <artifactId>commons-lang</artifactId>
      <version>2.5</version>
    </dependency>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.8.1</version>
      <scope>test</scope>
    </dependency>
  </dependencies>
  <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-
       plugin</artifactId>
        <configuration>
          <source>1.5</source>
          <target>1.5</target>
        </configuration>
      </plugin>
    </plugins>
  </build>
</project>
```

maven

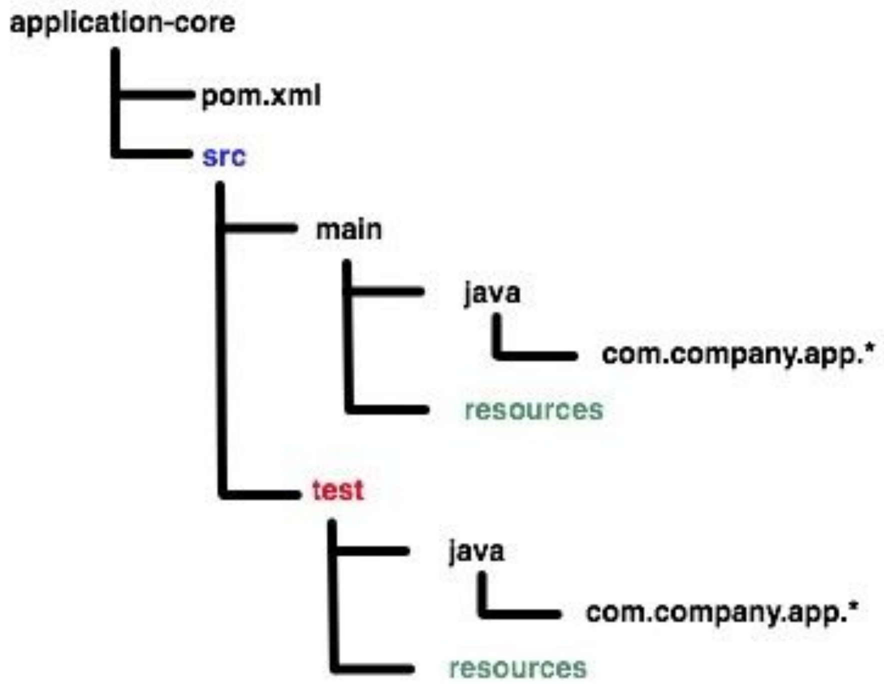
```groovy
apply plugin: 'java'

version="1.0-SNAPSHOT"
group="grId"
archivesBaseName="coc-comparison"

repositories {
    mavenCentral()
}

dependencies {
  compile 'commons-lang:commons-
        lang:2.5'
  testCompile 'junit:junit:4.8.1'
}
```

Gradle

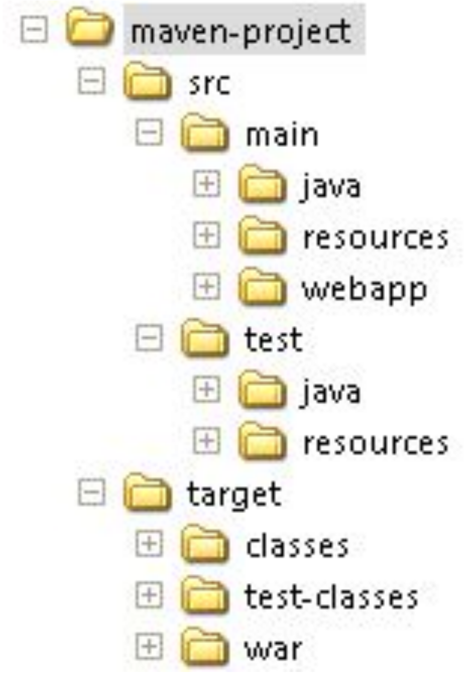
Convention
Over
Configuration

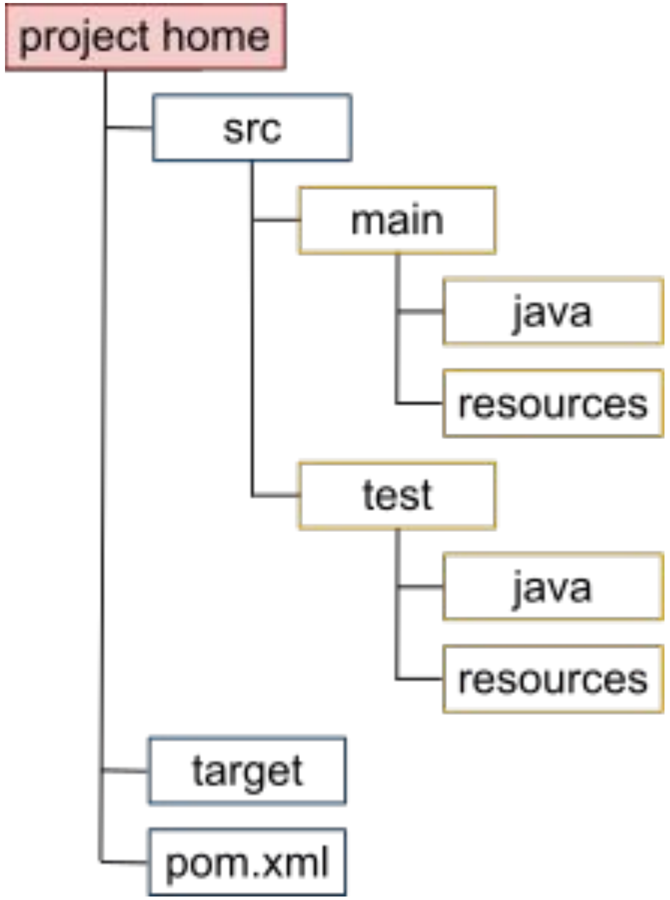
Tomek's Blog
http://kaczanowscy.pl/tomek

Standard Project Structure



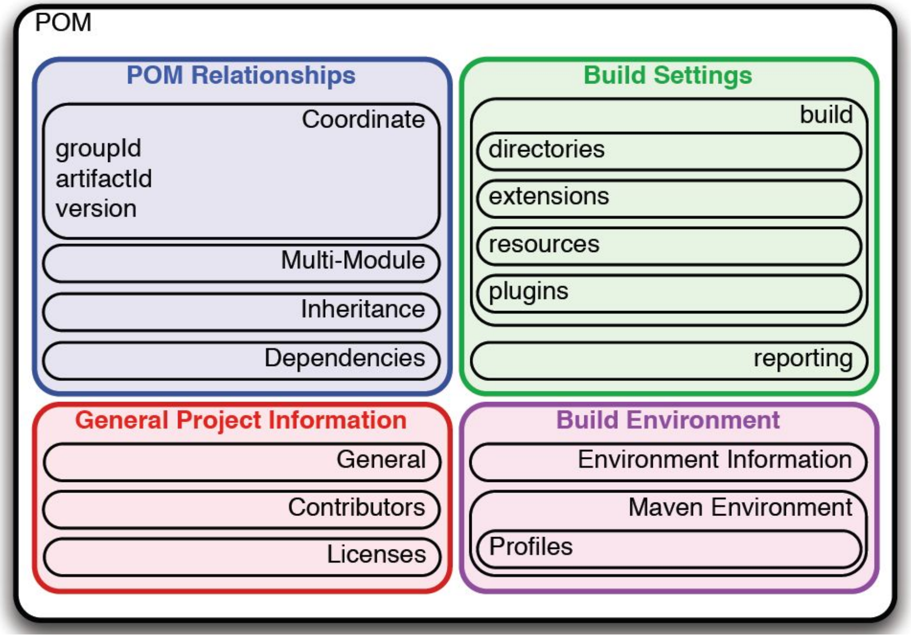
Maven project structure



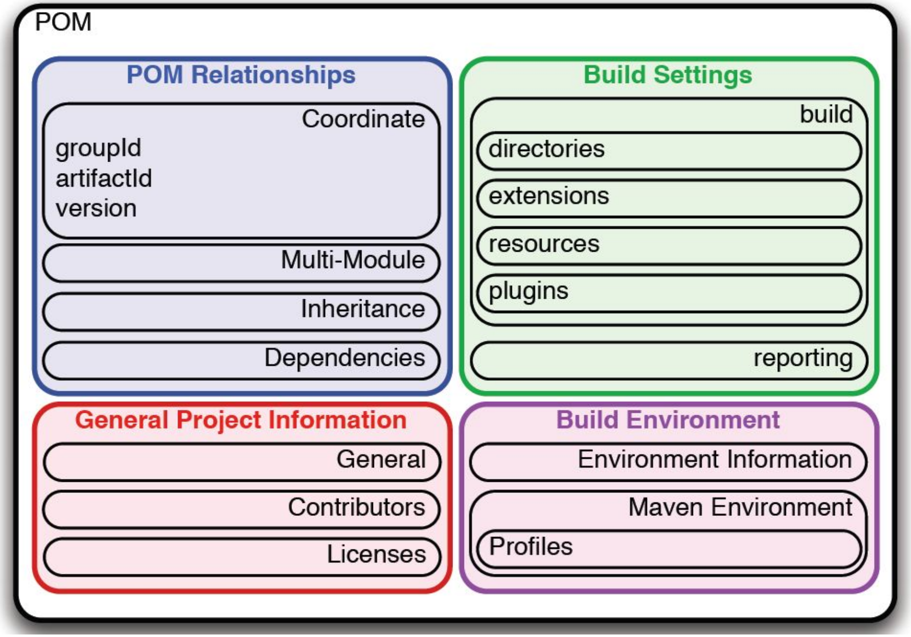


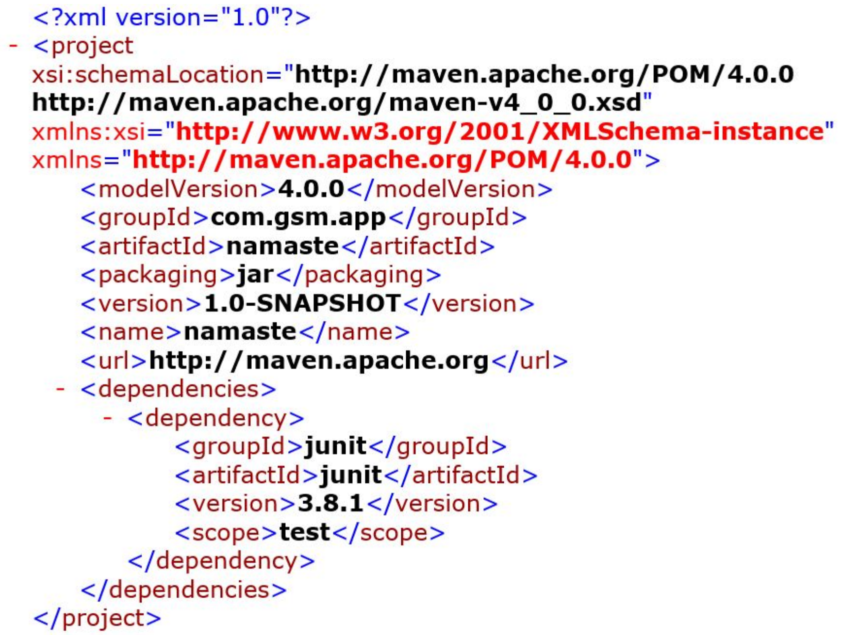
P.O.M

Project Object Model



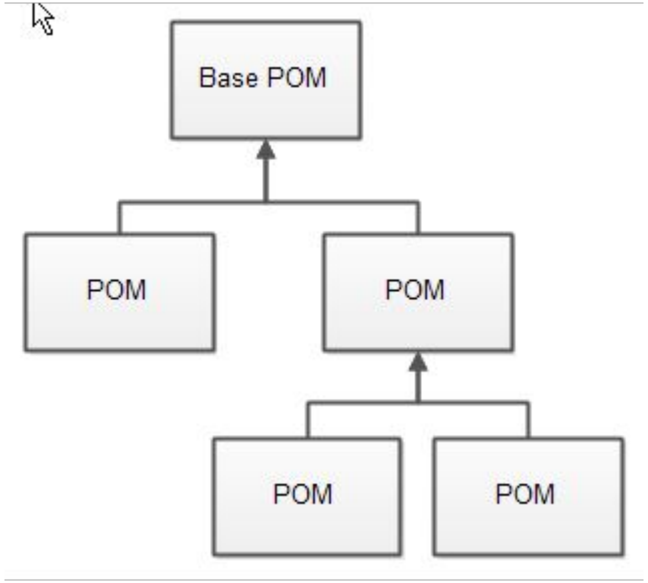
P.O.M.

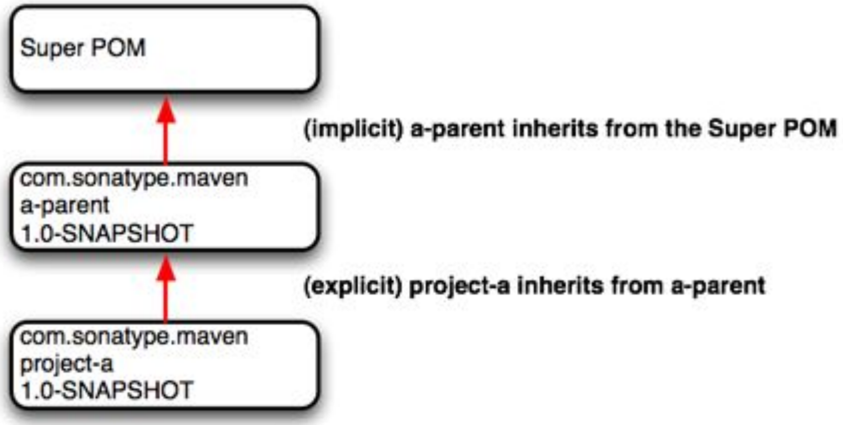




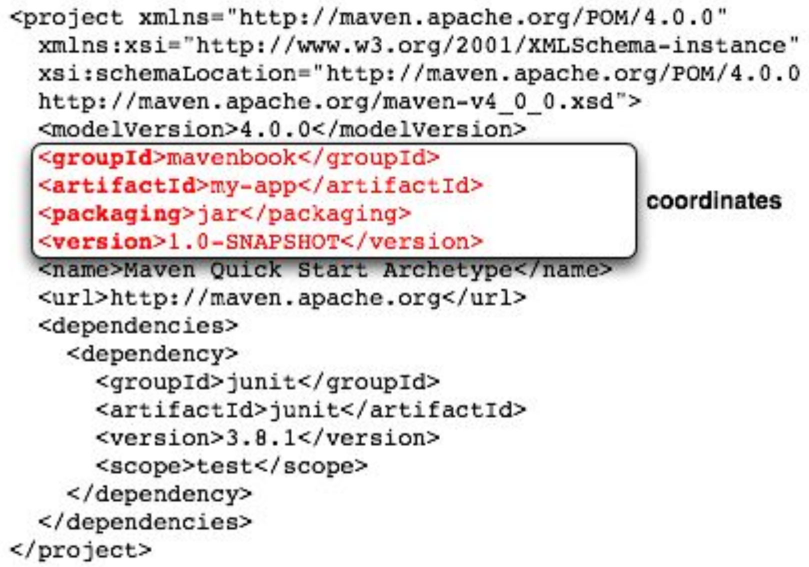
```
  <?xml version="1.0"?>
- <project
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/maven-v4_0_0.xsd"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://maven.apache.org/POM/4.0.0">
     <modelVersion>4.0.0</modelVersion>
     <groupId>com.gsm.app</groupId>
     <artifactId>namaste</artifactId>
     <packaging>jar</packaging>
     <version>1.0-SNAPSHOT</version>
     <name>namaste</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>
```

P.O.M. Inheritance




Super POM
(implicit) a-parent inherits from the Super POM
com.sonatype.maven
a-parent
1.0-SNAPSHOT
(explicit) project-a inherits from a-parent
com.sonatype.maven
project-a
1.0-SNAPSHOT

Dependencies



```
<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>mavenbook</groupId>
  <artifactId>my-app</artifactId>
  <packaging>jar</packaging>
  <version>1.0-SNAPSHOT</version>
  <name>Maven Quick Start Archetype</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>
```

groupId:artifactId:version

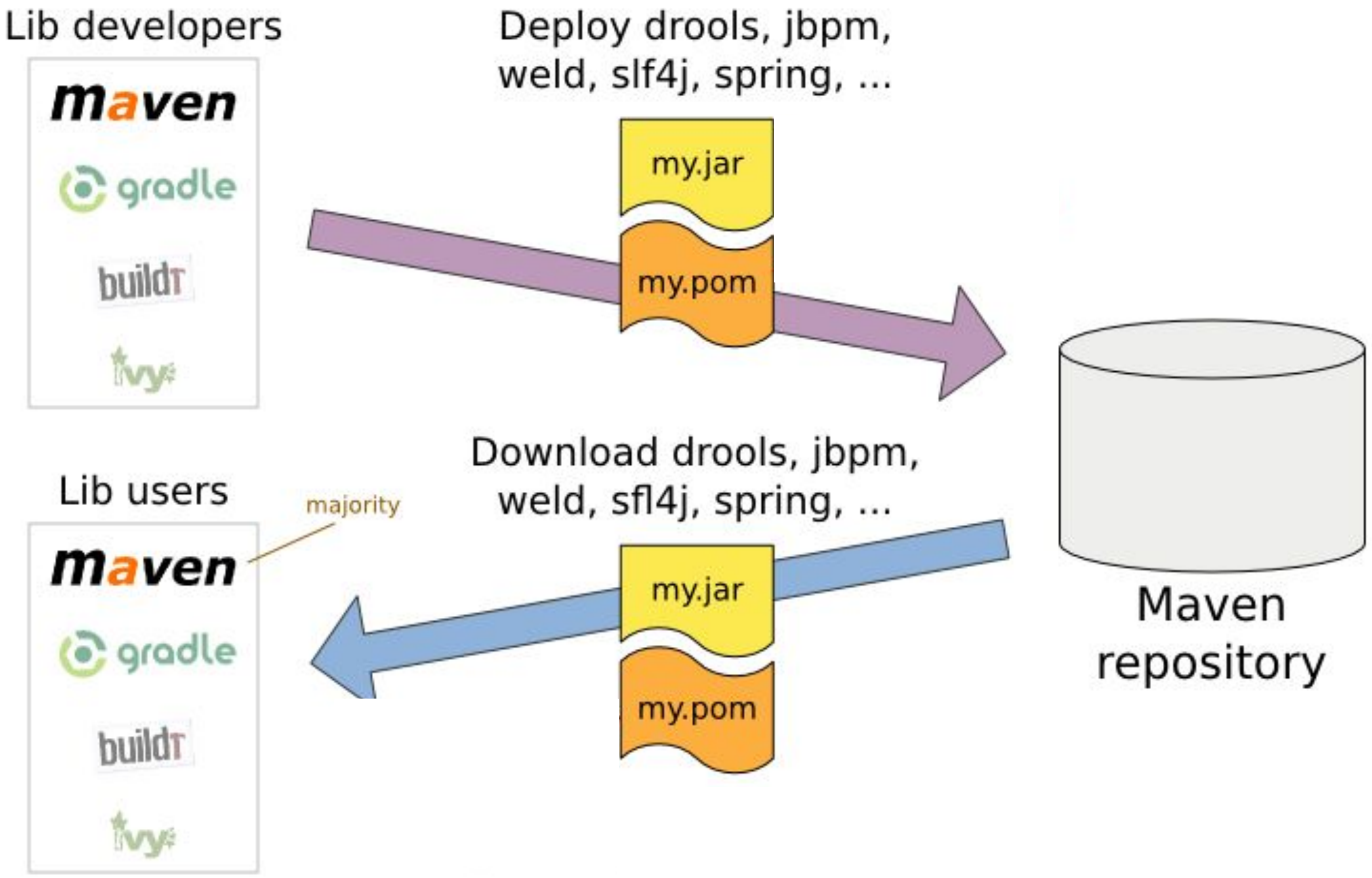
com.example:proj-name:0.0.1-SNAPSHOT

```
<groupId>com.example</groupId>

<artifactId>proj-name</artifactId>

<version>0.0.1-SNAPSHOT</version>
```

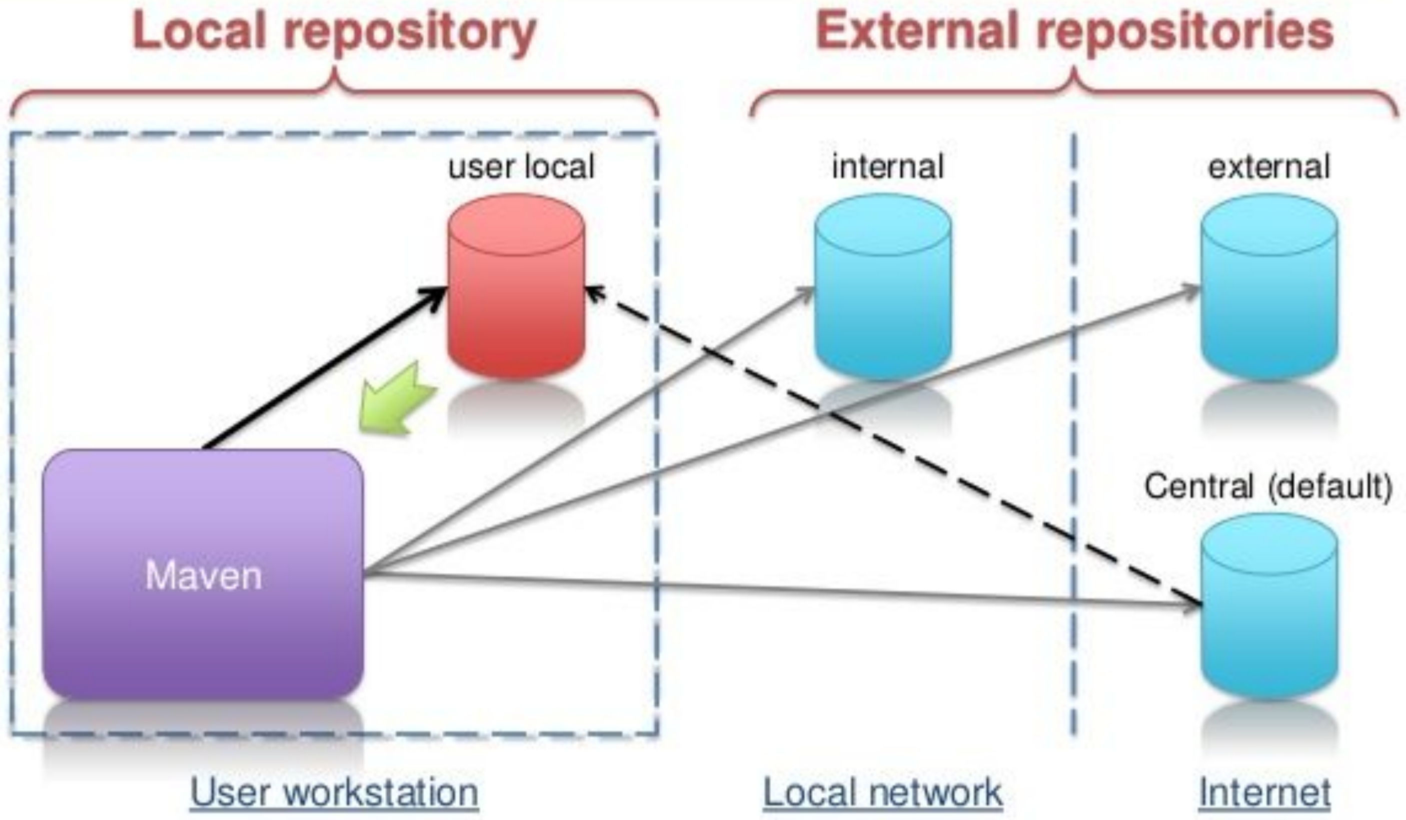
Semver (Semantic Versioning)

Major	Minor	Patch
2	21	2
breaking changes	features	bugfixes and hotfixes

v2.21.2


Lib developers
maven
gradle
buildr
ivy
Deploy drools, jbpm, weld, slf4j, spring, ...
my.jar
my.pom
Lib users
majority
maven
gradle
buildr
ivy
Download drools, jbpm, weld, sfl4j, spring, ...
my.jar
my.pom
Maven repository

Maven Repository



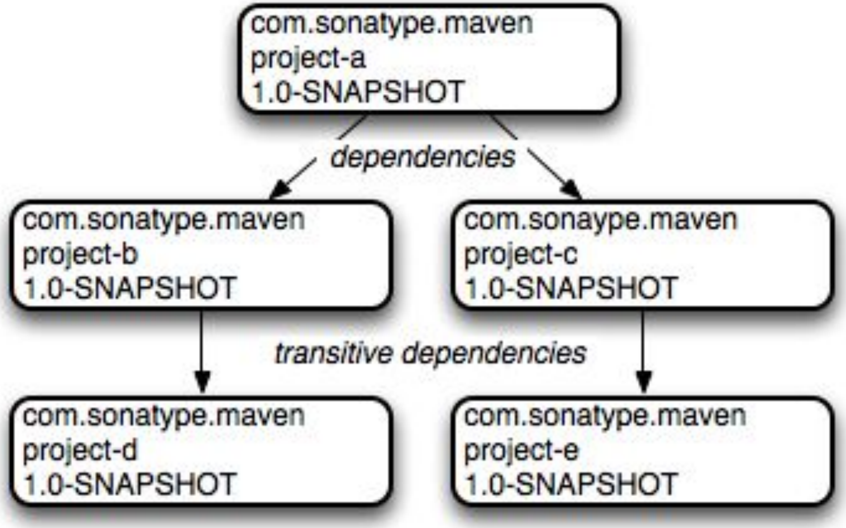
Popular maven Repos

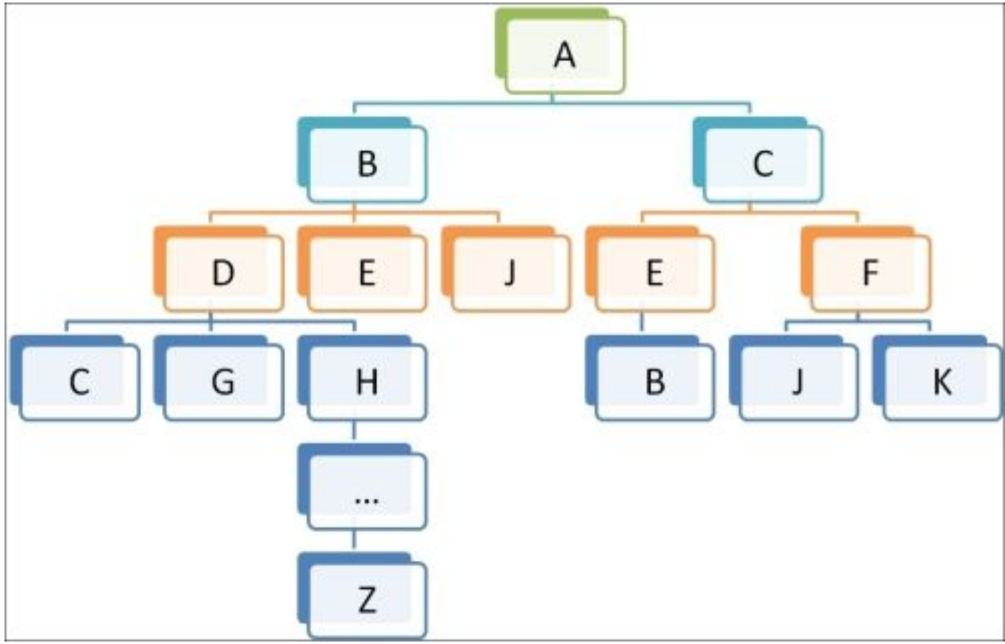
- mavenCentral
- JCenter
- JBoss

- ~/.m2

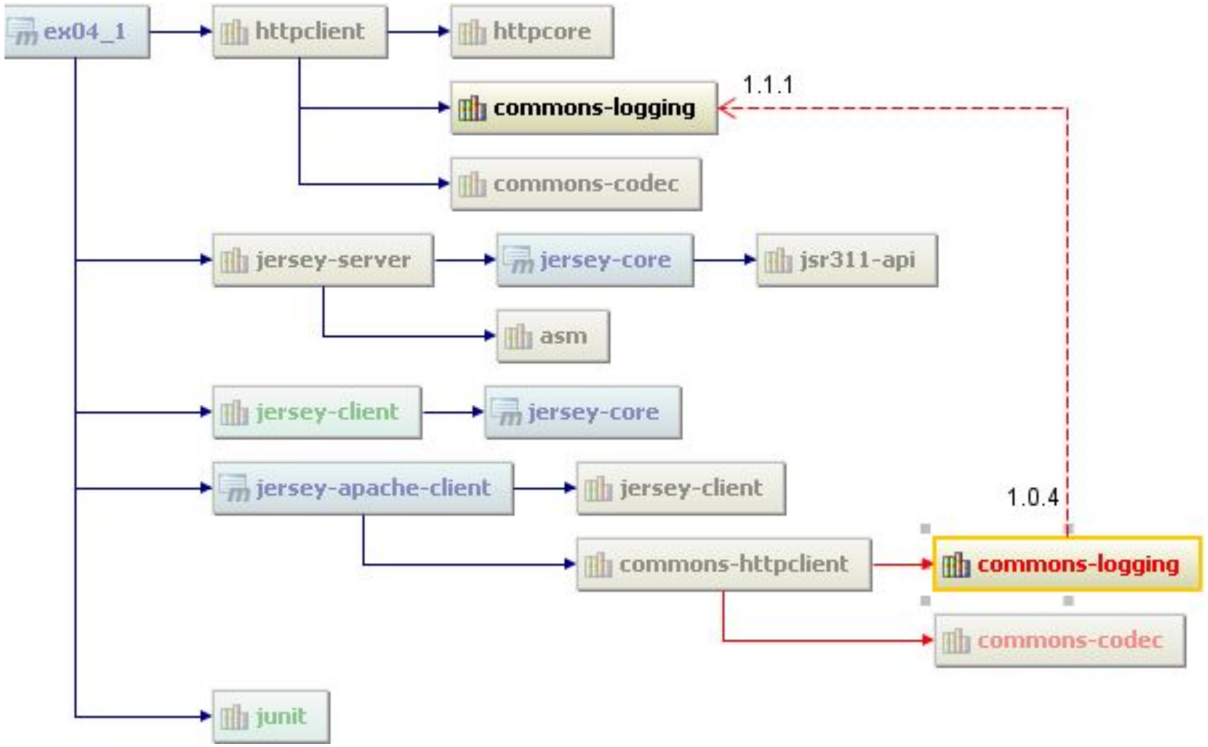
- https://mvnrepository.com/
- https://bintray.com/bintray/jcenter
- http://repository.jboss.com/maven2/
- C:\Users\{your_username}\.m2
- ~/.m2

Transitive dependencies

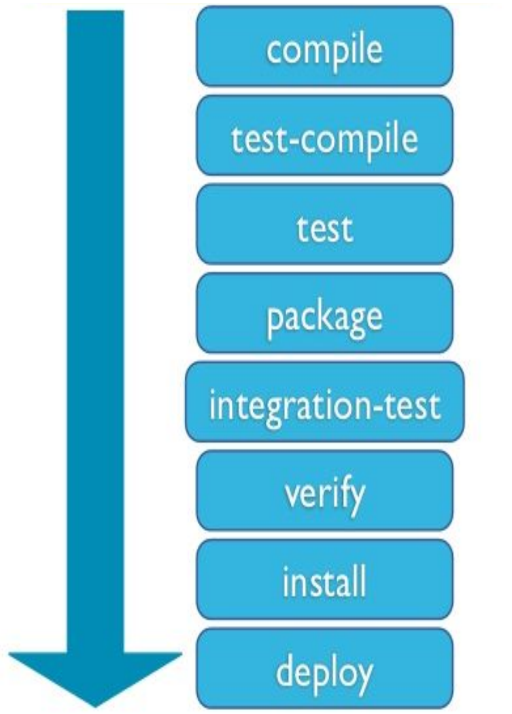


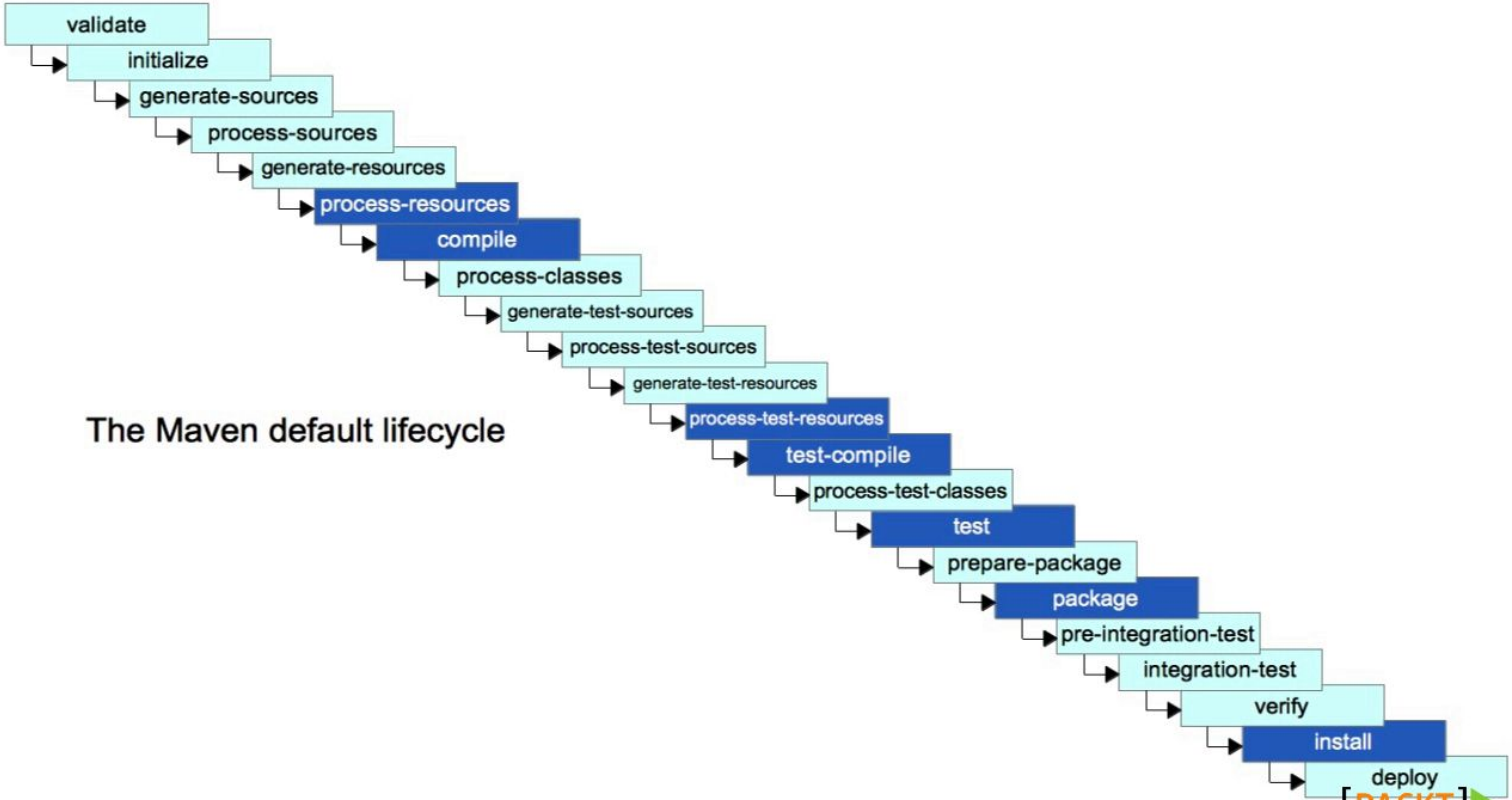


Dependency conflicts



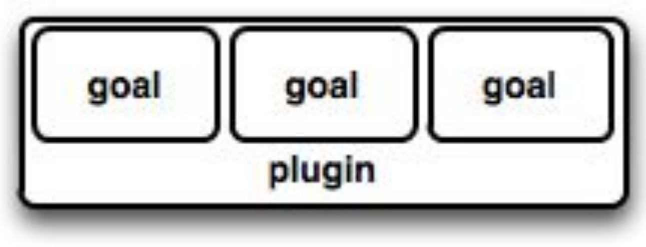
Lifecycle


compile
test-compile
test
package
integration-test
verify
install
deploy



The Maven default lifecycle

Plugins & Goals

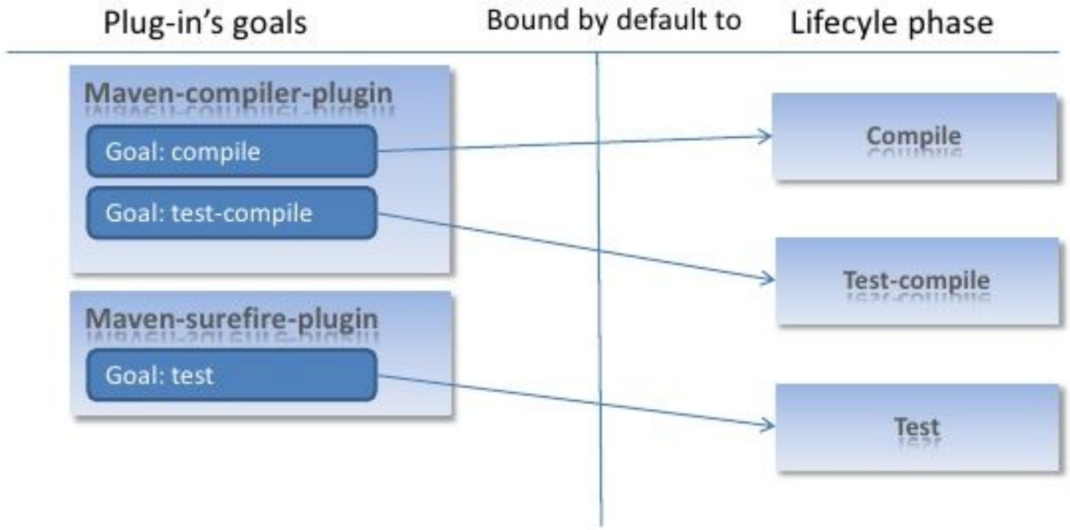


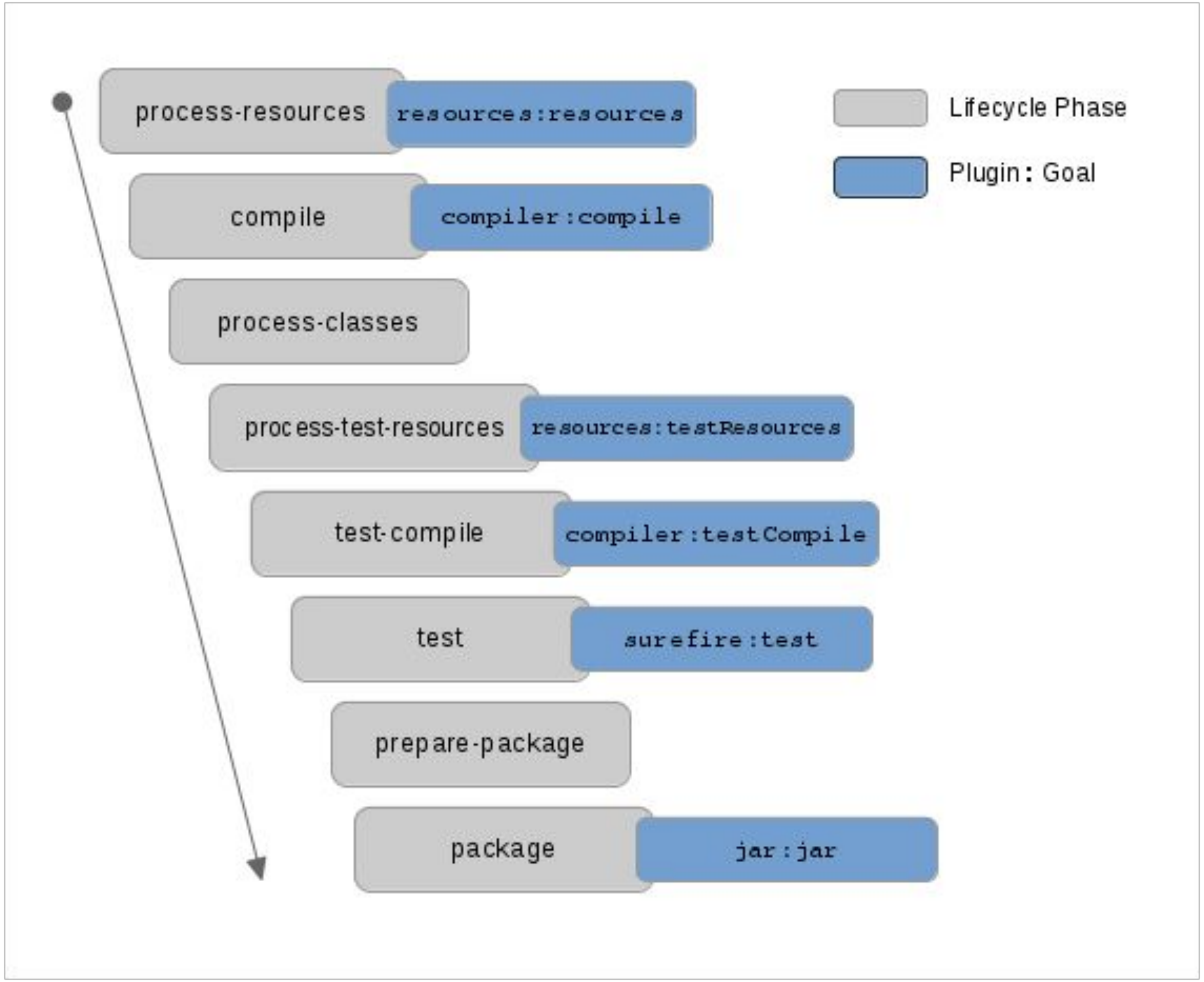
```
<plugin_name>:<goal_name>
```

```
compiler:compile
```

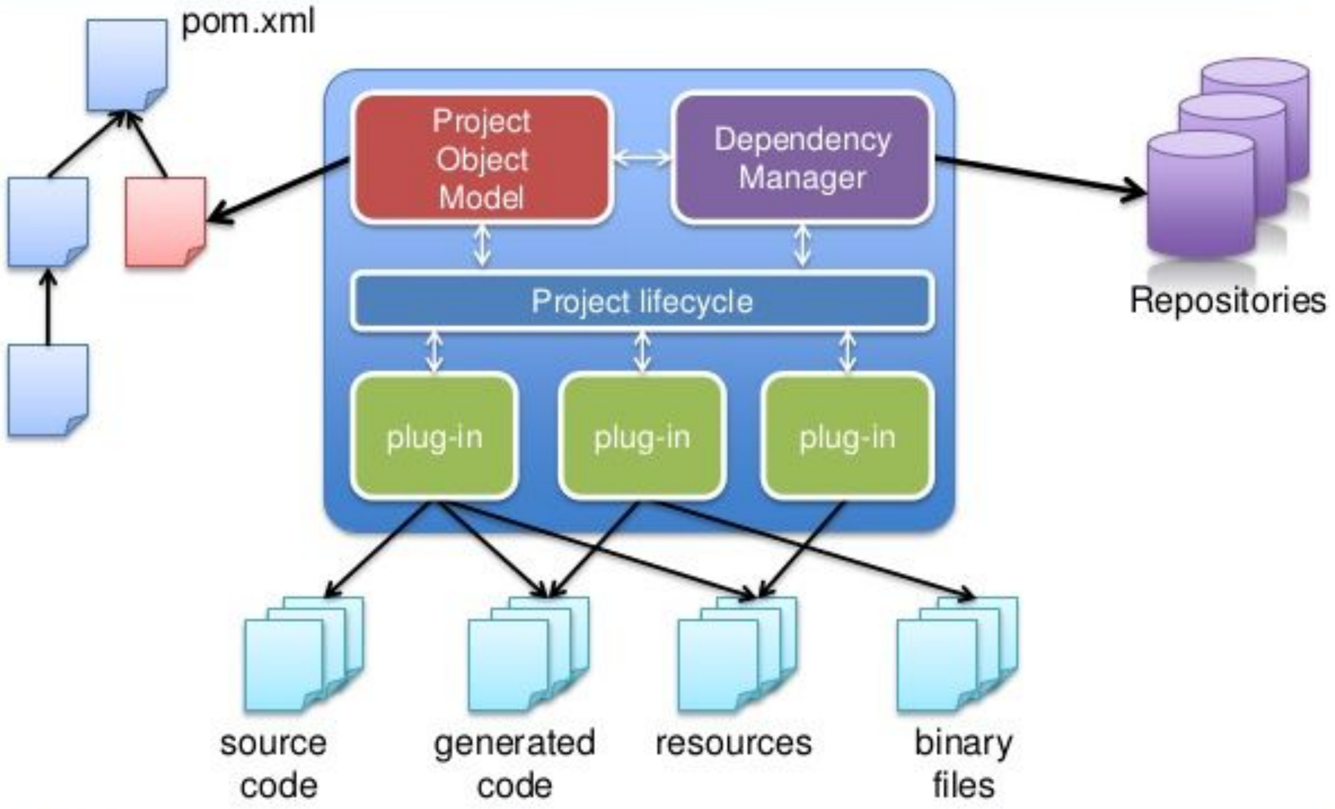
Maven Plug-ins and Goals

- Goals can be bound to build-lifecycle phases. Example:




process-resources
resources:resources
compile
compiler:compile
process-classes
process-test-resources
resources:testResources
test-compile
compiler:testCompile
test
surefire:test
prepare-package
package
jar:jar
Lifecycle Phase
Plugin: Goal

Maven project



Gradle

- Build by convention
- **Supports all Maven conventions**
- DSL
- Groovy
- Gradle Wrapper
- Easy Migration

gradle

Groovy / Kotlin

based DSL
(Domain Specific Language)

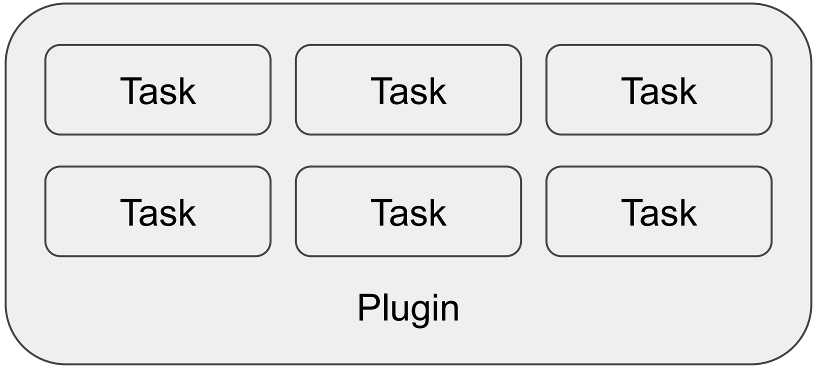
```
apply plugin: 'java'
apply plugin: 'eclipse'
apply plugin: 'idea'
apply plugin: 'org.springframework.boot'

group = 'devex.project'
version = '0.0.1-SNAPSHOT'

sourceCompatibility = 1.8
targetCompatibility = 1.8

repositories {
   mavenCentral()

   maven{
       url "https://mvnrepository.com/artifact/org.apache.commons/commons-io"
   }
}


dependencies {
   compile("org.springframework.boot:spring-boot-starter:2.5.0")
   compile("org.springframework:spring-web")
   compile("com.fasterxml.jackson.core:jackson-databind")

  // https://mvnrepository.com/artifact/org.apache.commons/commons-io
  compile group: 'org.apache.commons', name: 'commons-io', version: '1.3.2'
  compile group: 'commons-codec', name: 'commons-codec', version: '1.9'

  testCompile("junit:junit")
}
```

Gradle Wrapper

```
gradle wrapper
->
gradlew
gradlew.bat
```

Gradle Structure

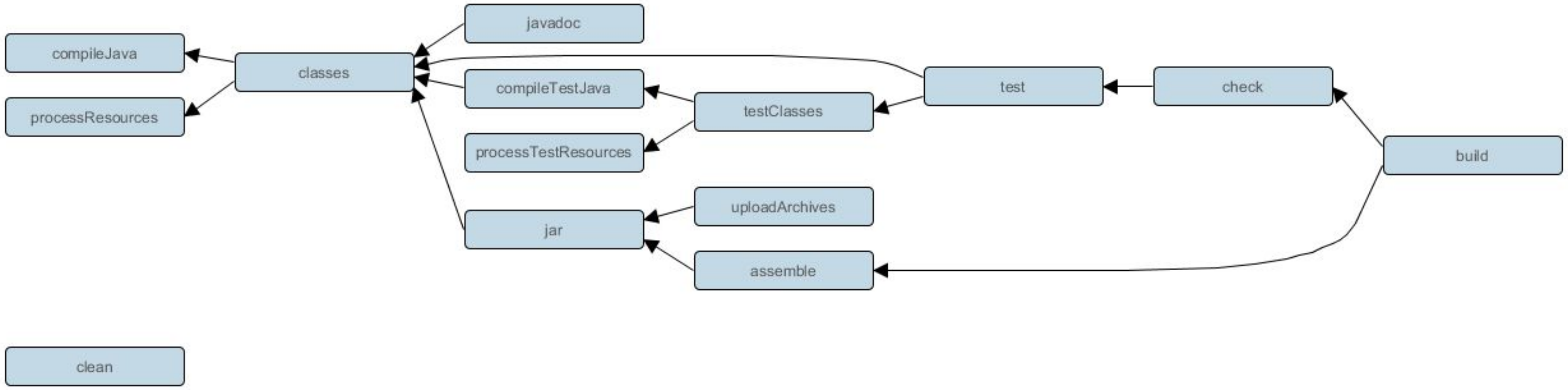
Plugins
Tasks
Dependencies



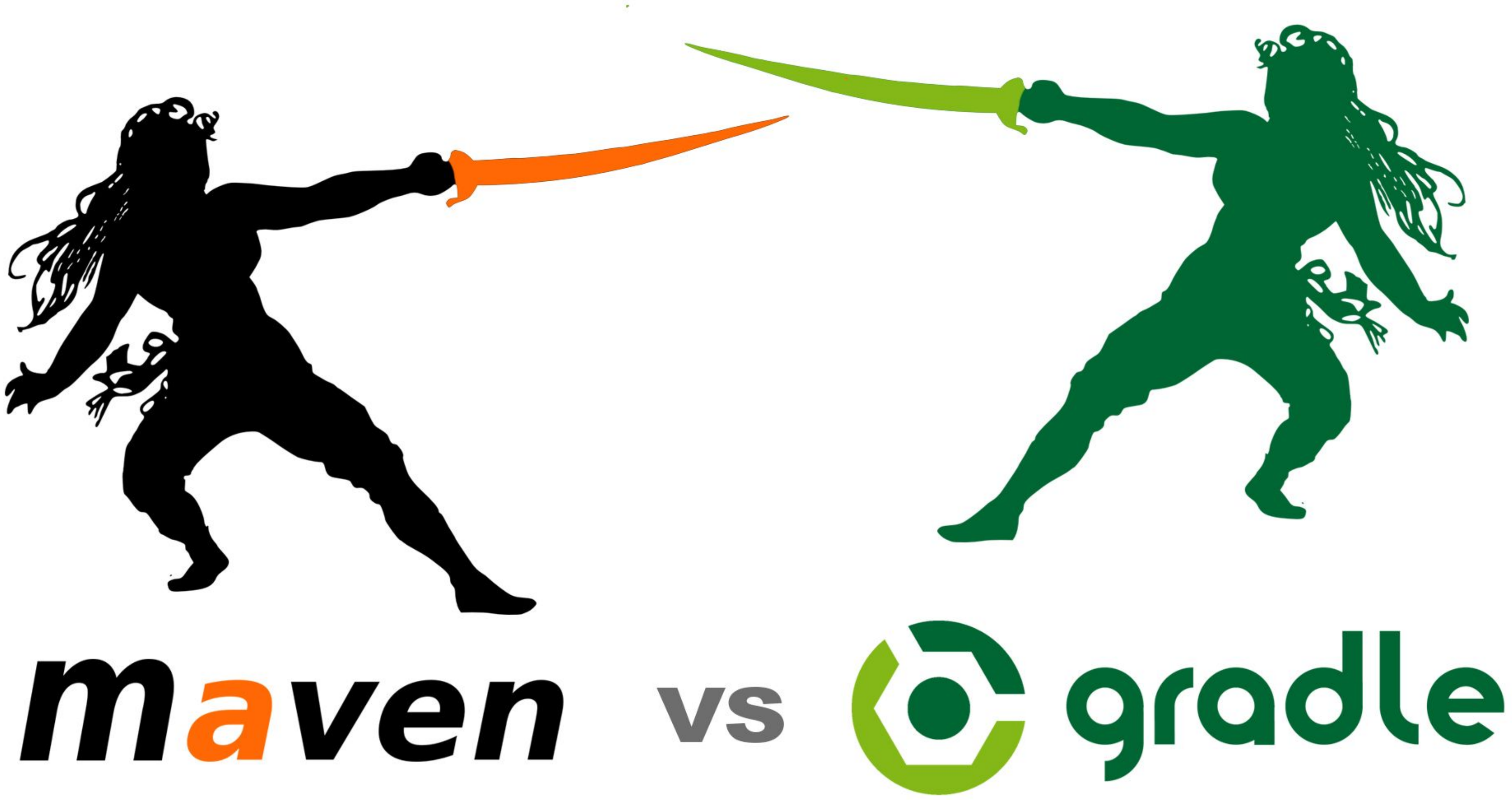
The Java plugin tasks:

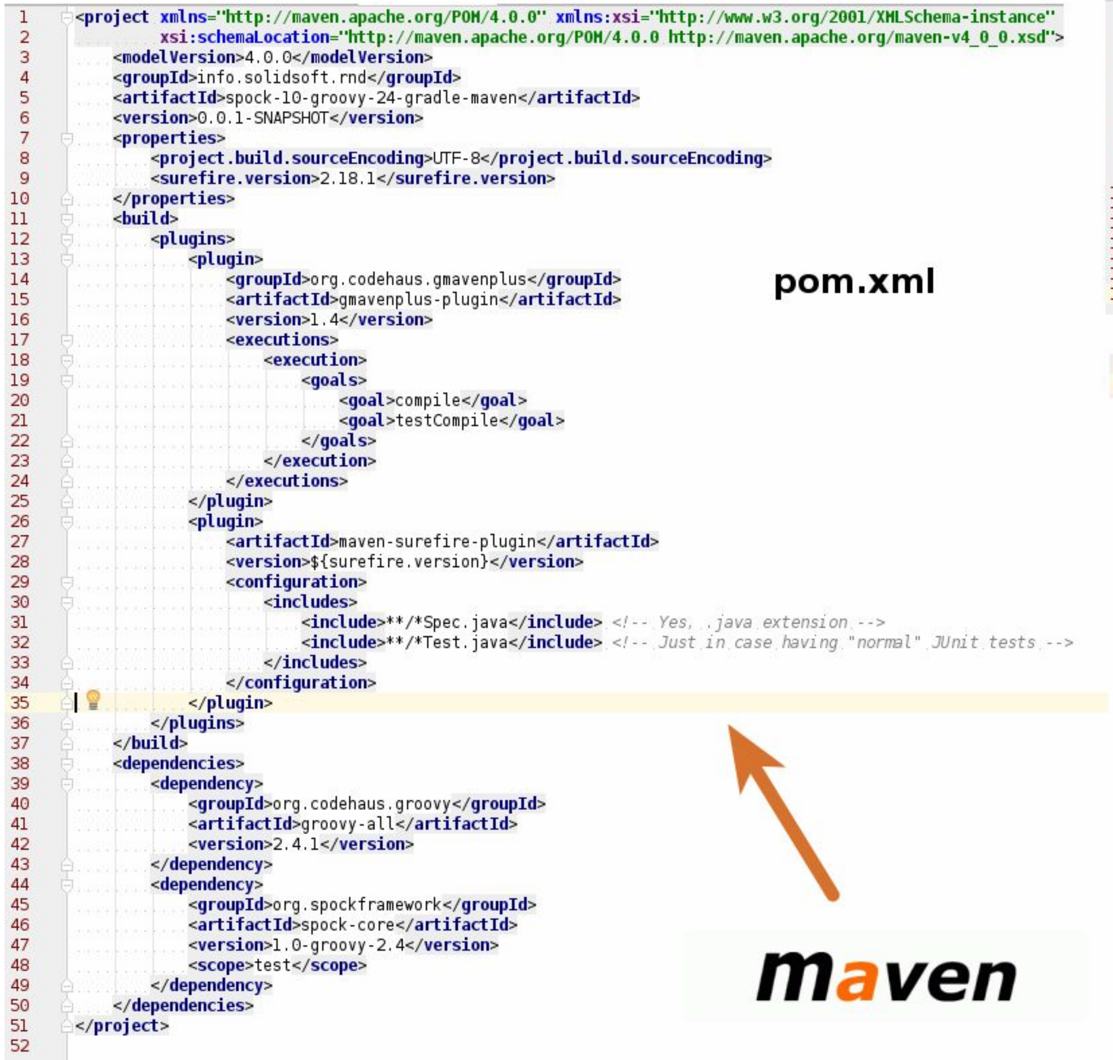
build, compileJava, assemble, jar, test, testJava, ...

The Java plugin tasks and dependencies



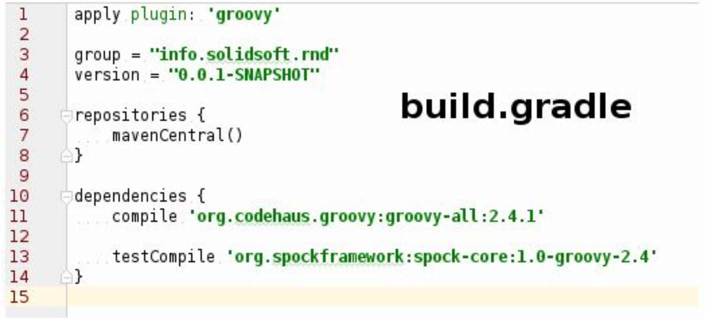
Comparison with examples


maven
vs
gradle



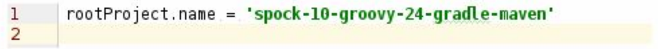
pom.xml

```xml
<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>info.solidsoft.rnd</groupId>
    <artifactId>spock-10-groovy-24-gradle-maven</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <surefire.version>2.18.1</surefire.version>
    </properties>
    <build>
        <plugins>
            <plugin>
                <groupId>org.codehaus.gmavenplus</groupId>
                <artifactId>gmavenplus-plugin</artifactId>
                <version>1.4</version>
                <executions>
                    <execution>
                        <goals>
                            <goal>compile</goal>
                            <goal>testCompile</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>
            <plugin>
                <artifactId>maven-surefire-plugin</artifactId>
                <version>${surefire.version}</version>
                <configuration>
                    <includes>
                        <include>**/*Spec.java</include> <!-- Yes, .java extension -->
                        <include>**/*Test.java</include> <!-- Just in case having "normal" JUnit tests -->
                    </includes>
                </configuration>
            </plugin>
        </plugins>
    </build>
    <dependencies>
        <dependency>
            <groupId>org.codehaus.groovy</groupId>
            <artifactId>groovy-all</artifactId>
            <version>2.4.1</version>
        </dependency>
        <dependency>
            <groupId>org.spockframework</groupId>
            <artifactId>spock-core</artifactId>
            <version>1.0-groovy-2.4</version>
            <scope>test</scope>
        </dependency>
    </dependencies>
</project>
```



build.gradle

```groovy
apply plugin: 'groovy'

group = "info.solidsoft.rnd"
version = "0.0.1-SNAPSHOT"

repositories {
    mavenCentral()
}

dependencies {
    compile 'org.codehaus.groovy:groovy-all:2.4.1'

    testCompile 'org.spockframework:spock-core:1.0-groovy-2.4'
}
```



```groovy
rootProject.name = 'spock-10-groovy-24-gradle-maven'
```

settings.gradle

maven

gradle

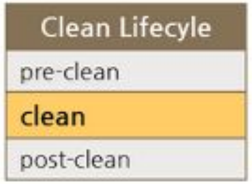
Terminology

---

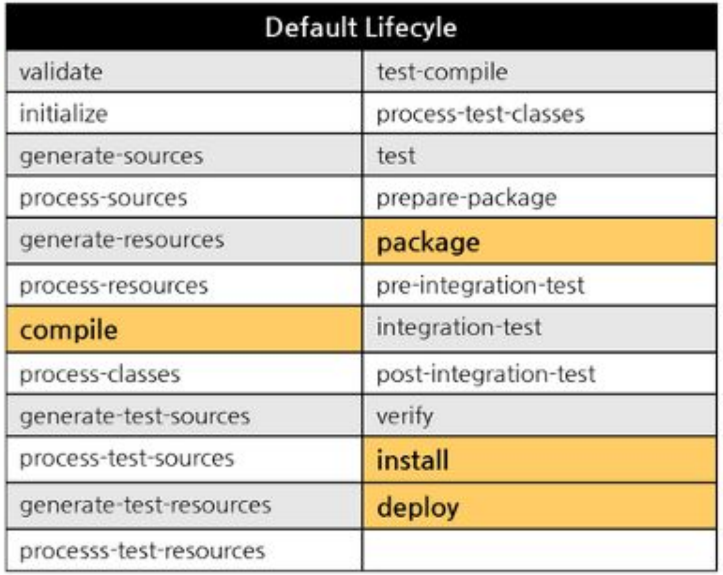
maven	gradle
• pom.xml	• build.gradle
• plugins & goals	• plugins & tasks
• lifecycle phases	• task dependencies
• XML	• Groovy based DSL

maven gradle

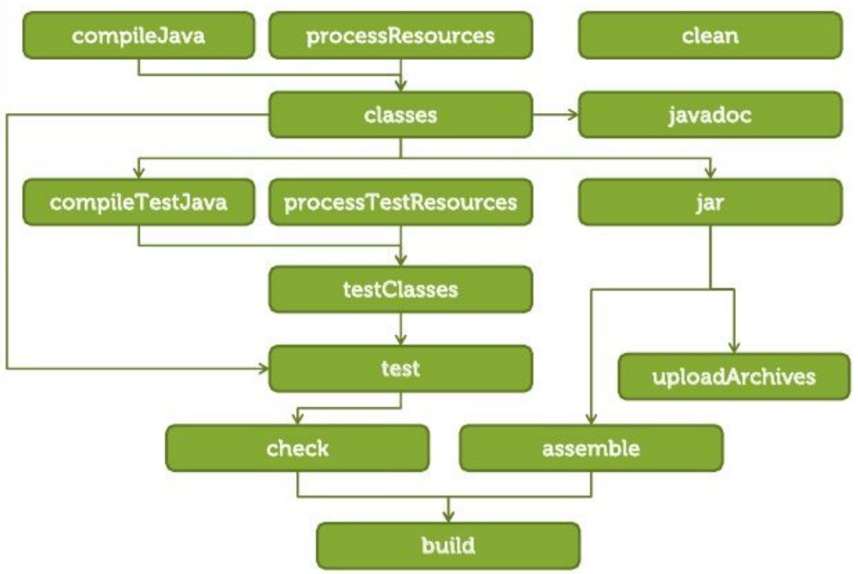
Lifecycle



Clean Lifecyle
pre-clean
clean
post-clean



Default Lifecyle	
validate	test-compile
initialize	process-test-classes
generate-sources	test
process-sources	prepare-package
generate-resources	**package**
process-resources	pre-integration-test
compile	integration-test
process-classes	post-integration-test
generate-test-sources	verify
process-test-sources	**install**
generate-test-resources	**deploy**
processs-test-resources	



Execution order

maven gradle

Generate a new project

```
$ mvn archetype:generate
```

```
-DgroupId=org.example.myorg
```

```
$ gradle init --type java-application
```

```
OR JUST
```

```
 $ gradle init
```

```
OVER THE CREATED MAVEN APPLICATION
```

maven gradle

Dependency

```xml
<dependencies>
 <dependency>

<groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>22.0</version>
    <scope>compile</scope>
 </dependency>
 <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>3.8.1</version>
    <scope>test</scope>
 </dependency>
</dependencies>
```

```groovy
dependencies {
    compile 'com.google.guava:guava:22.0'

    // Use JUnit test framework
    testCompile 'junit:junit:4.12'
}
```

maven gradle

Plugins

```xml
<build>
 <pluginManagement>
 <plugins>
   <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-compiler-plugin</artifactId>
     <configuration>
       <source>1.5</source>
       <target>1.5</target>
     </configuration>
   </plugin>
 </plugins>
 </pluginManagement>
</build>
```

```
apply plugin: 'java'
apply plugin: 'maven'
```

Or

```
plugins {
     id 'java'
     id 'application'
  }
```

maven gradle

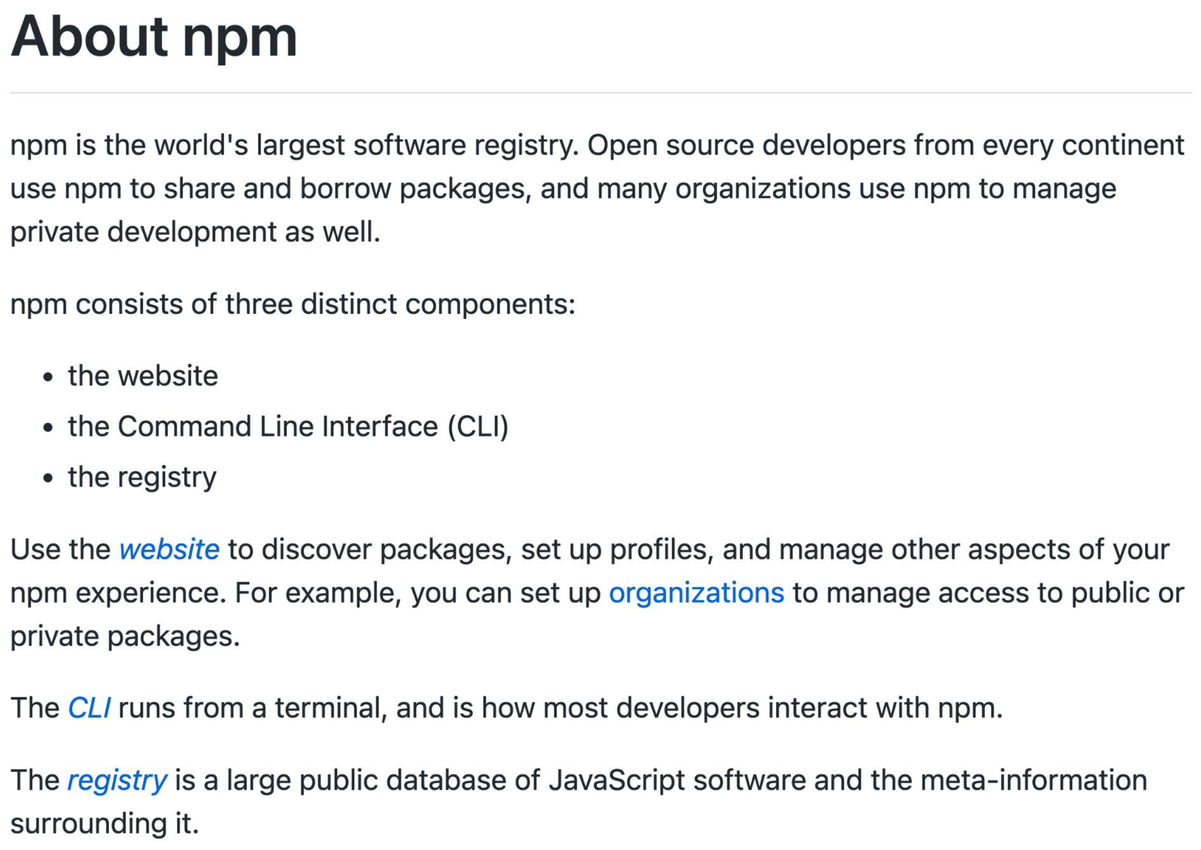
Building, Packaging and installing a java jar application

- `mvn package`
- `mvn install`

- `gradle build`

npm

- Dependency Management
- **Install Tools globally and run them**
- JSON config files
- package-lock.json
- Node Version Manager
- NPM Registry



About npm

npm is the world's largest software registry. Open source developers from every continent use npm to share and borrow packages, and many organizations use npm to manage private development as well.

npm consists of three distinct components:

- the website
- the Command Line Interface (CLI)
- the registry

Use the *website* to discover packages, set up profiles, and manage other aspects of your npm experience. For example, you can set up organizations to manage access to public or private packages.

The *CLI* runs from a terminal, and is how most developers interact with npm.

The *registry* is a large public database of JavaScript software and the meta-information surrounding it.

Source: official website - https://docs.npmjs.com/about-npm

Node Version Manager

package.json

https://docs.npmjs.com/cli/v9/configuring-npm/package-json

```
"name": "my-blog-backend",
"version": "1.0.0",
"description": "This project is the personal blog",
"private": true,
"main": "index.js",

"scripts": {
  "start": "npx babel-node src/server.js",
  "dev": "npx babel-node src/server.js",
  "test": "echo \"Error: no test specified\" && exit 1",
  "lint": "eslint ./server"
},
```

package.json Meta-Data

https://docs.npmjs.com/cli/v9/configuring-npm/package-json

```
"keywords": [
  "node",
  "vue"
],
"repository": {
  "type": "git",
  "url": "https://github.com/npm/cli.git"
},
"author": {
  "name": "Sanchitha",
  "email": "s@sharma.com",
  "url": "http://wordspoolsite.wordpress.com/"
},
"bugs":{
  "url": "https://github.com/owner/project/issues",
  "email": "project@hostname.com"
},
"homepage": "https://github.com/owner/project#readme",
"license": "MIT",
```

package.json dependencies

https://docs.npmjs.com/cli/v9/configuring-npm/package-json

```
"dependencies": {
  "express": "^4.17.1"
},
"devDependencies": {
  "@babel/cli": "^7.12.8",
  "@babel/core": "^7.12.9",
  "@babel/node": "^7.12.6",
  "@babel/preset-env": "^7.12.7"
}
```

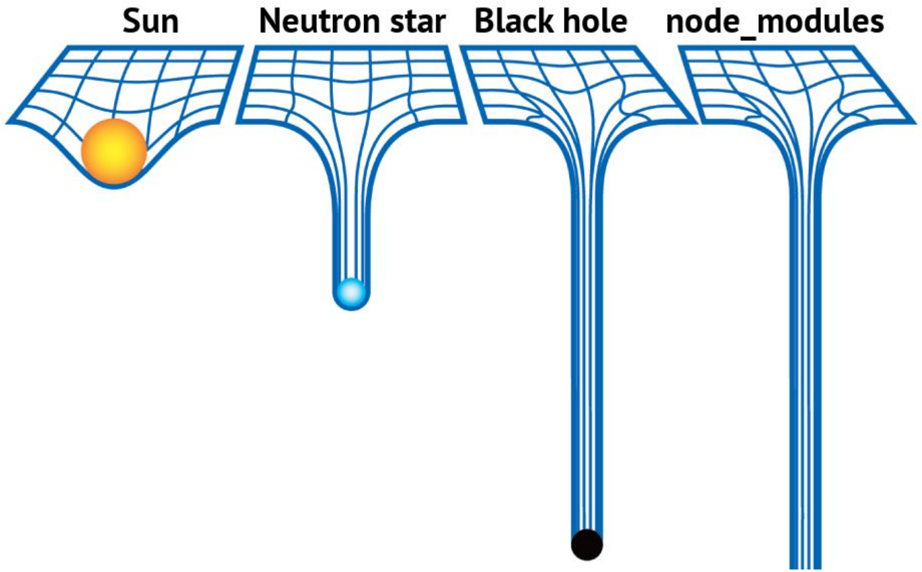
package-lock.json

https://docs.npmjs.com/cli/v9/configuring-npm/package-lock-json

```
{
   "requires" : true,
   "lockfileVersion" : 1,
   "dependencies" : {
       "abbrev" : {
           "version" : "1.1.1",
           "resolved" : "https://registry.npmjs.org/abbrev/-/abbrev-1.1.1.tgz",
           "integrity" : "sha512-nne9/IiQ/h…" ,
           "dev" : true
       },
       "accepts" : {
           "version" : "1.3.5",
           "resolved" : "https://registry.npmjs.org/accepts/-/accepts-1.3.5.tgz",
           "integrity" : "sha1-63d99gEXI6OxTopywIBcjoZ0a9I=" ,
           "dev" : true,
           "requires" : {
               "mime-types" : "~2.1.18",
               "negotiator" : "0.6.1"
           }
       }
  }
}
```

node_modules

Local copy of each dependency (incl. transitive) for each project.



Global Packages

```
npm i -g <package_name>
```

```
npm install -g @angular/cli
```

Terminology

maven

- pom.xml
- plugins & goals
- lifecycle phases
- XML

gradle

- build.gradle
- plugins & tasks
- task dependencies
- Groovy based DSL

npm

- package.json
- scripts
- Script dependencies
- JSON
- package-lock.json

Generating a new project

```
$ mvn archetype:generate
```

```
$ gradle init --type java-application
```

```
$ npm init
```

Dependencies

maven

```xml
<dependencies>
 <dependency>
   <groupId>com.google.guava </groupId>
   <artifactId>guava</artifactId>
   <version>22.0</version>
   <scope>compile</scope>
 </dependency>
 <dependency>
   <groupId>junit</groupId>
   <artifactId>junit</artifactId>
   <version>3.8.1</version>
   <scope>test</scope>
 </dependency>
</dependencies>
```

gradle

```groovy
dependencies {
   compile 'com.google.guava:guava:22.0'

   // Use JUnit test framework
   testCompile 'junit:junit:4.12'
}
```

npm

```json
"dependencies": {
  "express": "^4.17.1"
},
"devDependencies": {
  "@babel/cli": "^7.12.8",
  "@babel/core": "^7.12.9",
  "@babel/node": "^7.12.6",
  "@babel/preset-env": "^7.12.7"
}
```

Build Package and Install

maven

```
mvn package

mvn install
```

gradle

```
gradle build
```

npm

```
npm run “build”
```

Conclusion

Pros of build tools

maven Gradle npm

- Manage Dependencies
- Compile Code
- Test Code
- Package Projects
- Deploy & Share Projects
- Automate the development process
- Install & Use projects
- Extensible
- Easily configurable

Bibliography and References

- https://fr.slideshare.net/tomek_k/convention-over-configuration-maven-3-polyglot-maven-gradle-and-ant
- https://www.slideshare.net/joaomiguel.pereira/an-introduction-to-maven
- http://www.lihaoyi.com/post/WhatsinaBuildTool.html
- **http://books.sonatype.com/mvnex-book/reference/index.html**
- https://docs.gradle.org/current/userguide/userguide.html