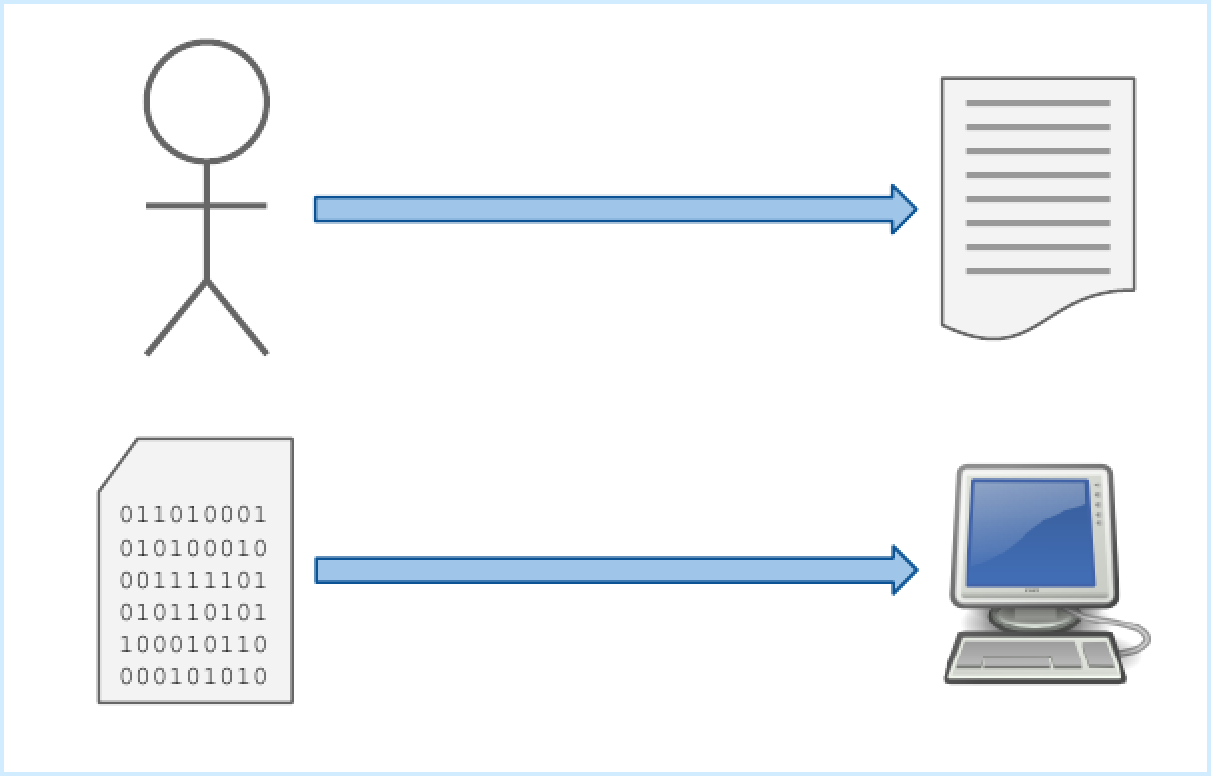
Sperasoft
Your game dev partner

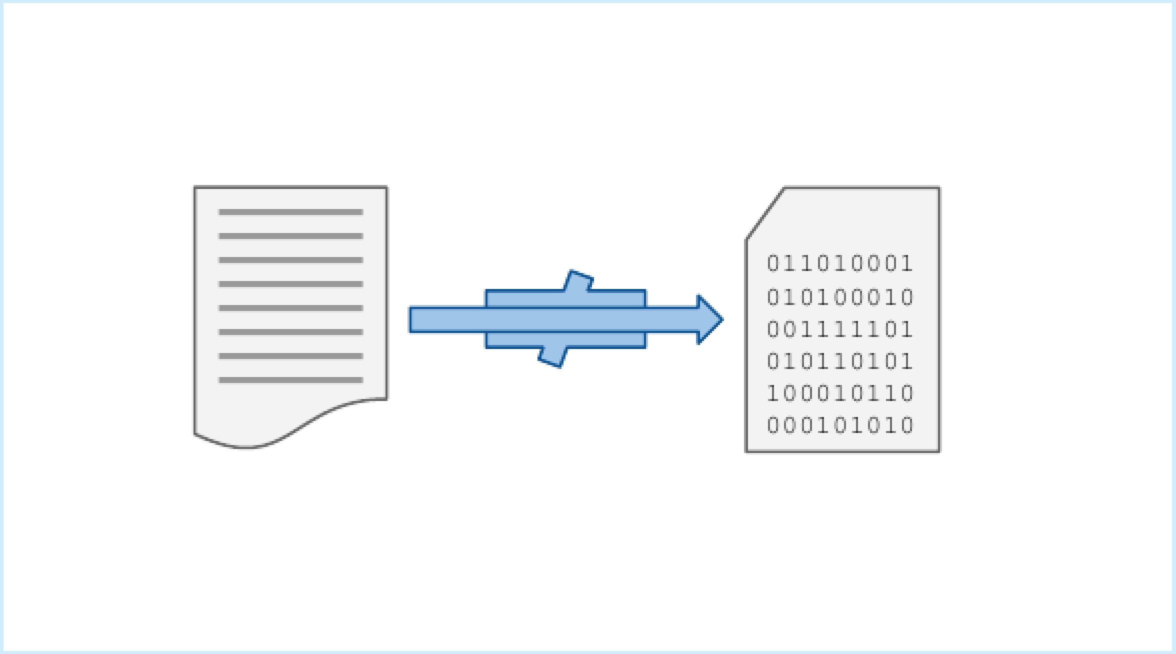
Introduction to Maven

TOC

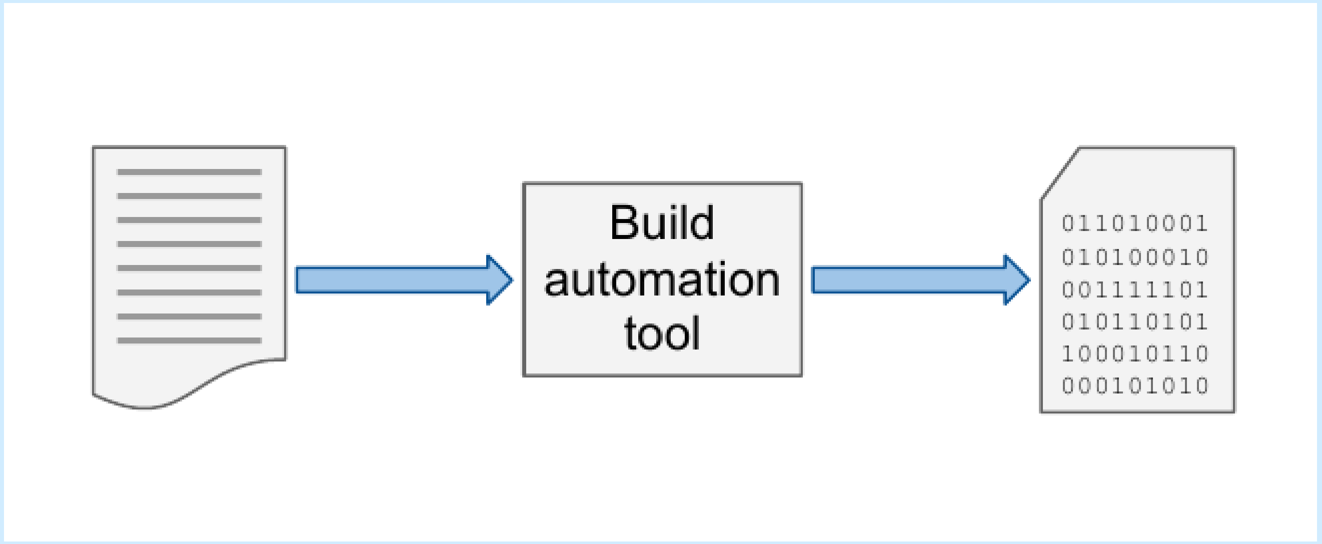
Software Development



Software Discrepancy



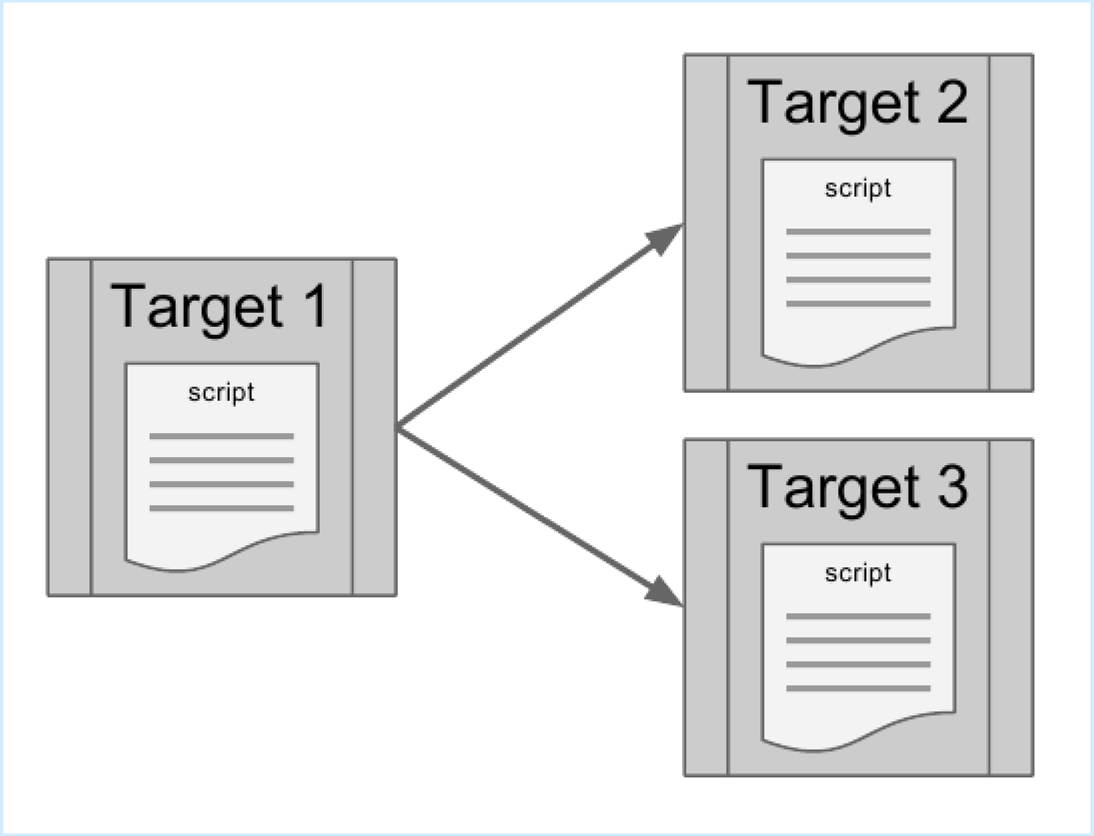
Software Discrepancy Solution



Examples: Build tools

- shell script
- make (1977)
- Apache Ant (2000)
- MSBuild (2005)
- Apache Maven (2002)

make



Holy War

Maven Vs. Ant

Example: Ant Script

```
<?xml version="1.0" encoding="UTF-8"?>
<project name="AntProject" basedir="." default="jar">
    <property file="nbproject/nbjdk.properties"/>
    <property name="user.properties.file"
location="${netbeans.user}/build.
properties"/>
    <property file="${user.properties.file}"/>
    <import file="nbproject/jdk.xml"/>
    <target name="-init" depends="-jdk-init">
        <property file="user.build.properties"/>
        <property file="build.properties"/>
    </target>
    <target name="compile" depends="-init" description="Compile main
sources.">
        <mkdir dir="${classes.dir}"/>
        <depend srcdir="${src.dir}" destdir="${classes.dir}"
cache="build/depcache"
            <classpath path="${cp}"/>
        </depend>
        <javac srcdir="${src.dir}" destdir="${classes.dir}" source="1.5"
debug="${d
            <classpath path="${cp}"/>
            <compilerarg value="-Xlint:unchecked"/>
        </javac>
        <copy todir="${classes.dir}">
            <fileset dir="${src.dir}" excludes="${jar.excludes}"/>
        </copy>
    </target>
    <target name="jar" depends="compile" description="Build JAR file for
main sour
        <jar jarfile="${jar}" compress="true"><!--
manifest="${manifest}" -->
            <fileset dir="${classes.dir}"/>
        </jar>
    </target>
```

```
 <target name="run" depends="compile" description="Run application.">
       <fail unless="main.class">Must set property 'main.class' (e.g. in
build.pro
       <java classname="${main.class}" fork="true" failonerror="true">
           <classpath path="${run.cp}"/>
           <jvmarg value="-ea"/>
       </java>
    </target>
    <target name="compile-tests" depends="compile">
       <mkdir dir="${test.classes.dir}"/>
       <depend srcdir="${test.dir}" destdir="${test.classes.dir}"
cache="build/test
           <classpath path="${test.cp}"/>
       </depend>
       <javac srcdir="${test.dir}" destdir="${test.classes.dir}" source="1.5"
debug
           <classpath path="${test.cp}"/>
           <compilerarg value="-Xlint:unchecked"/>
       </javac>
       <copy todir="${test.classes.dir}">
           <fileset dir="${test.dir}" excludes="${jar.excludes}"/>
       </copy>
    </target>
    <target name="run-tests" depends="compile-tests" description="Run JUnit
tests."
       <mkdir dir="${test.results.dir}"/>
       <junit failureproperty="tests.failed" showoutput="true" fork="true">
           <batchtest todir="${test.results.dir}">
               <fileset dir="${test.dir}">
                   <include name="**/*Test.java"/>
               </fileset>
           </batchtest>
           <classpath path="${test.run.cp}"/>
           <formatter type="brief" usefile="false"/>
           <formatter type="xml"/>
       </junit>
```

Example: Maven POM

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:x
<modelVersion>4.0.0</modelVersion>
<groupId>com.sperasoft</groupId>
<artifactId>helloworld</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
</project>
```

Apache Maven

Download: http://maven.apache.org

Version: 2.2.1

Requires: JDK 5+

Nature: Java command line program

Invocation: mvn

Install Apache Maven

1. Get apache-maven-2.2.1-bin.zip
2. Unpack it to a folder
3. Set M2_HOME env var to the above folder

Quote

"A maven (also mavin) is a trusted expert in a particular field, who seeks to pass knowledge on to others."

(http://en.wikipedia.org/wiki/Maven)

or

Configuration by Convention

or

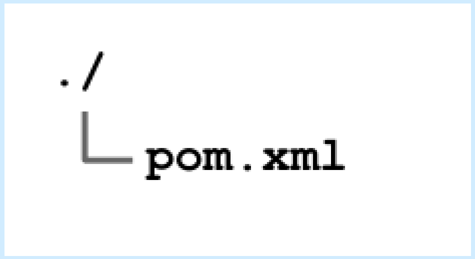
Do it right way

Maven Conventions

1. Project layout
2. Build lifecycle

Project Layout

Simple

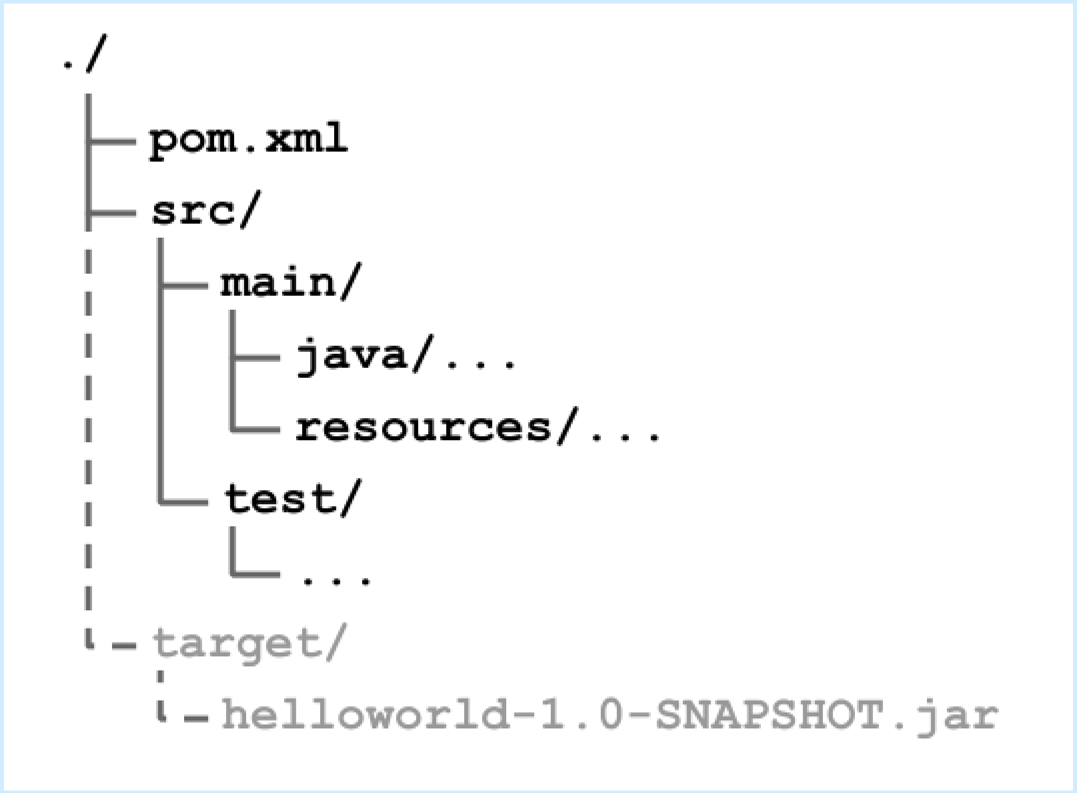


```
./
└ pom.xml
```

pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:x
<modelVersion>4.0.0</modelVersion>
<groupId>com.sperasoft</groupId>
<artifactId>helloworld</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
</project>
```

Project Layout: Typical



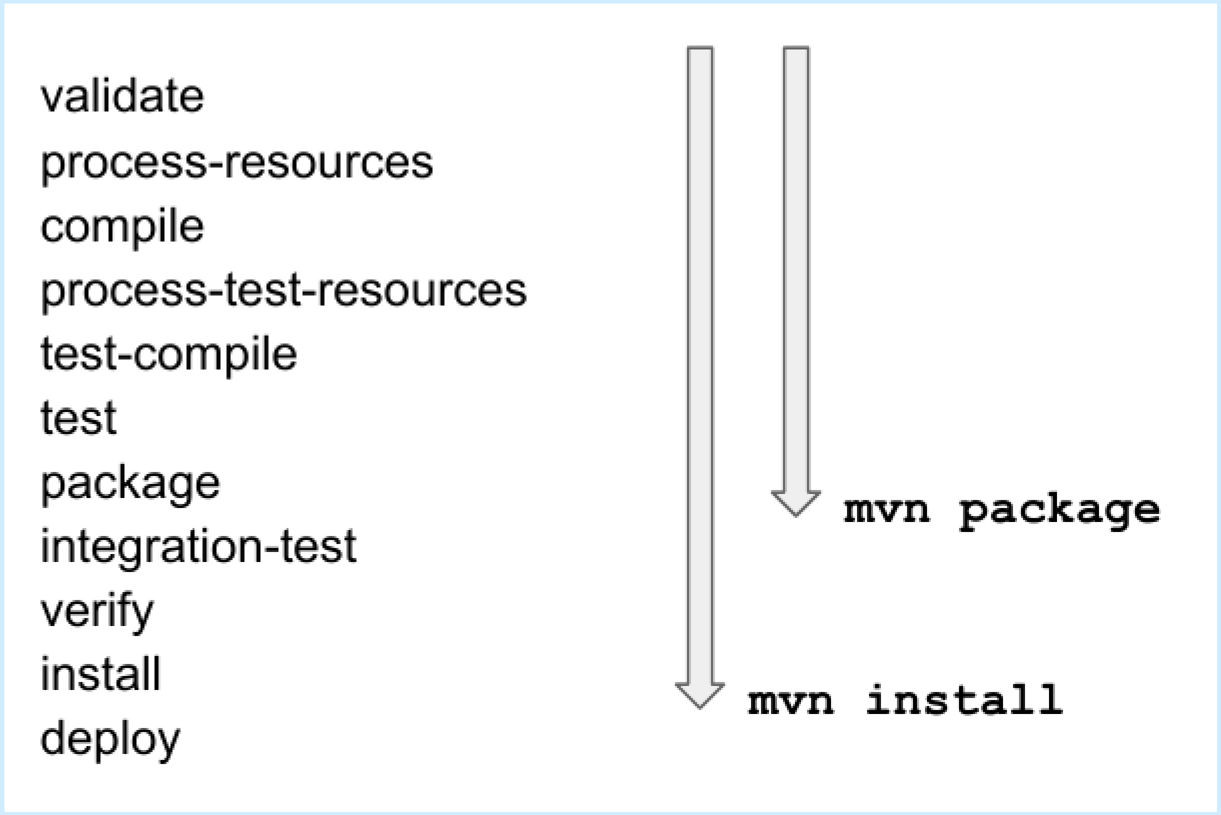
```
./
├── pom.xml
├── src/
│   ├── main/
│   │   ├── java/...
│   │   └── resources/...
│   └── test/
│       └── ...
└── target/
    └── helloworld-1.0-SNAPSHOT.jar
```

Build Lifecycle

Lifecycles

- Default lifecycle (build)
- Clean lifecycle
- Site lifecycle

Default Lifecycle


validate
process-resources
compile
process-test-resources
test-compile
test
package
integration-test
verify
install
deploy

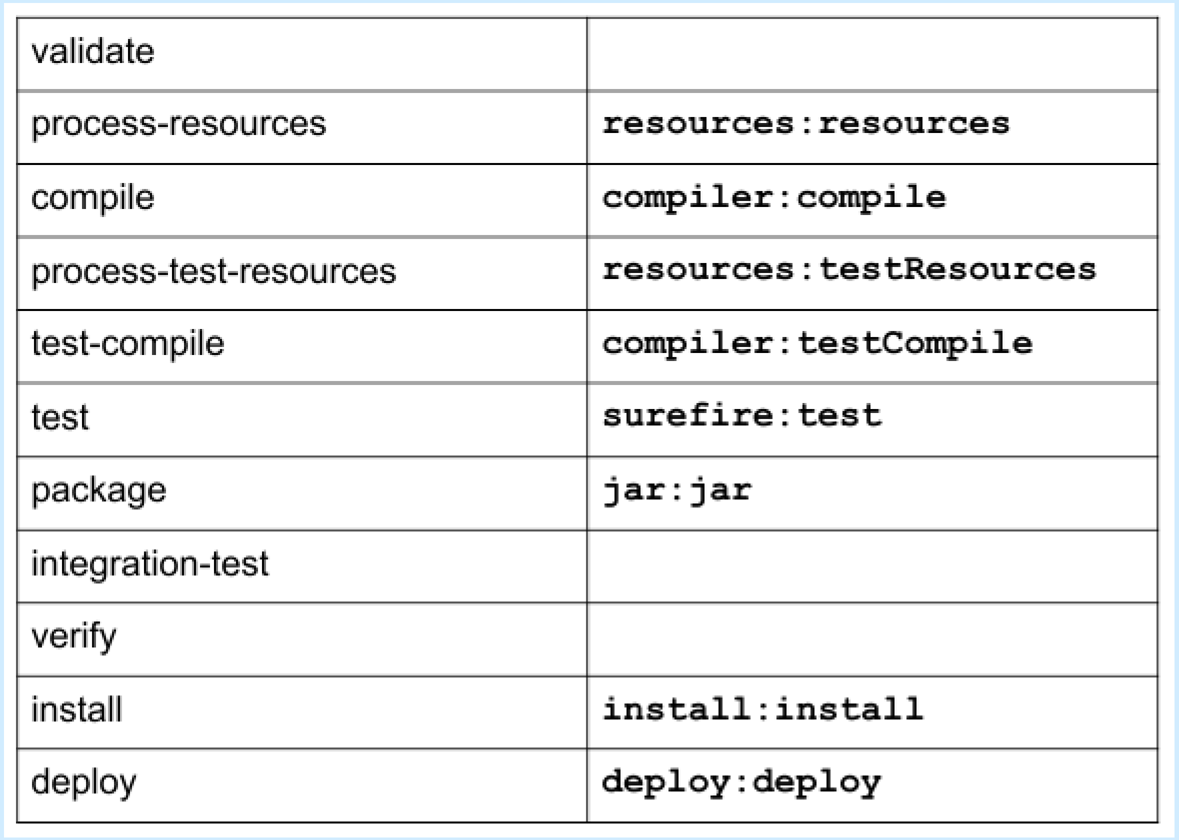
```
mvn package
```

```
mvn install
```

Example: Compiler Plugin

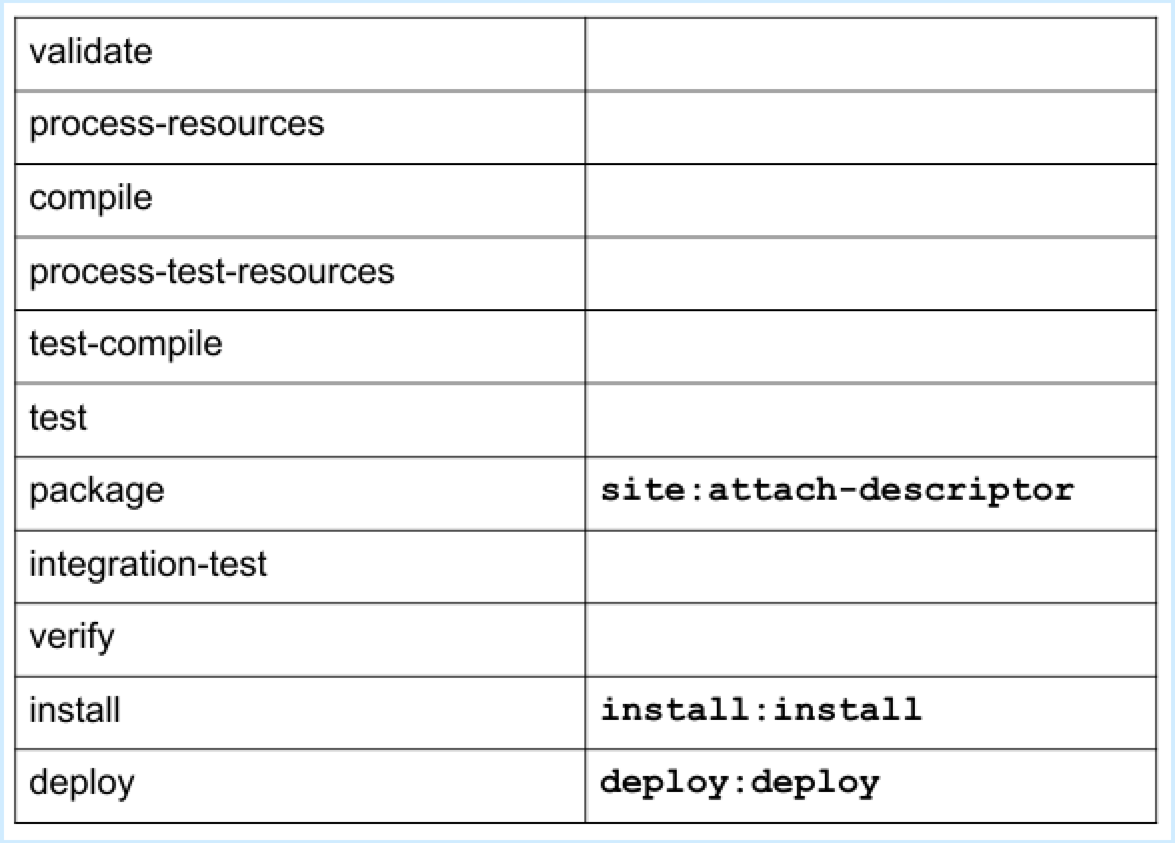
- compiler
    - compiler:compile
    - compiler:testCompile

Lifecycle Bindings: jar



validate	
process-resources	`resources:resources`
compile	`compiler:compile`
process-test-resources	`resources:testResources`
test-compile	`compiler:testCompile`
test	`surefire:test`
package	`jar:jar`
integration-test	
verify	
install	`install:install`
deploy	`deploy:deploy`

Lifecycle Bindings: pom

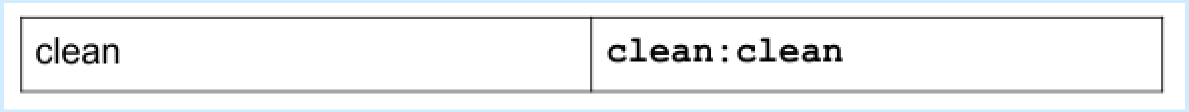


validate	
process-resources	
compile	
process-test-resources	
test-compile	
test	
package	`site:attach-descriptor`
integration-test	
verify	
install	`install:install`
deploy	`deploy:deploy`

pom.xml: packaging

```
<project
xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:x
<modelVersion>4.0.0</modelVersion>
<groupId>com.sperasoft</groupId>
<artifactId>helloworld</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
</project>
```

Clean lifecycle bindings



clean	`clean:clean`

Dependency Management Basis

● Dependency identification	Maven Coordinates
● Getting dependency	Repositories

Maven Coordinates

groupId: org.testng

artifactId: testng

version: 5.11

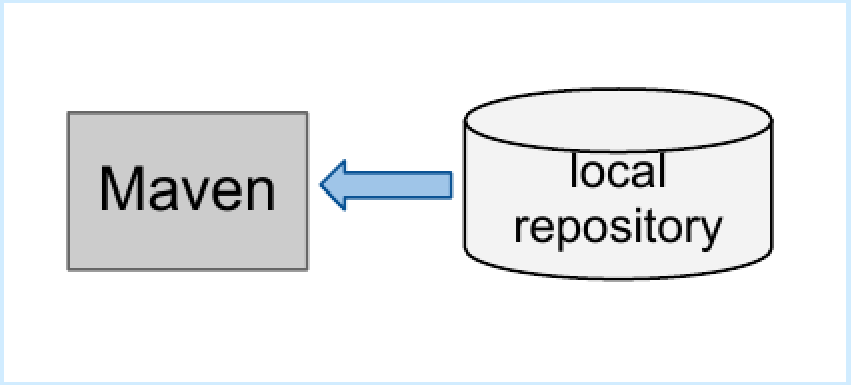
packaging: jar

classifier: jdk15

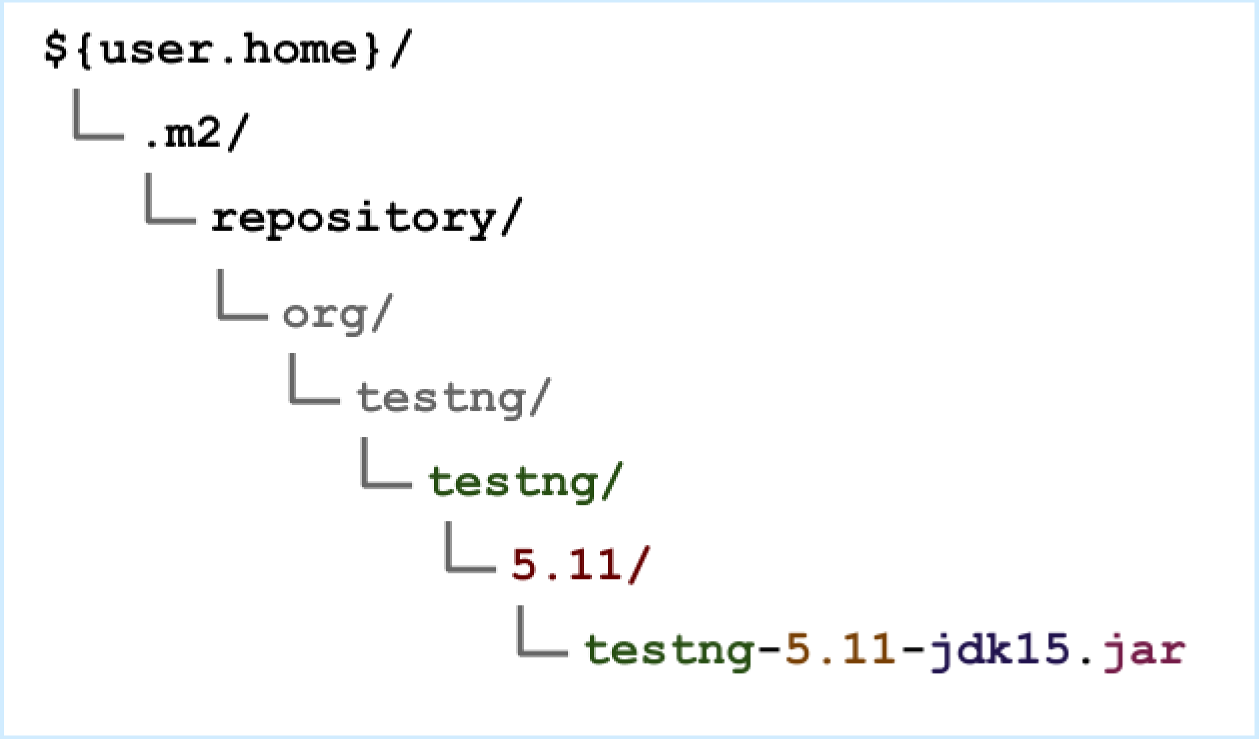
pom.xml: coordinates

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:x
<modelVersion>4.0.0</modelVersion>
<groupId>com.sperasoft</groupId>
<artifactId>helloworld</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>
</project>
```

Artifact Resolution

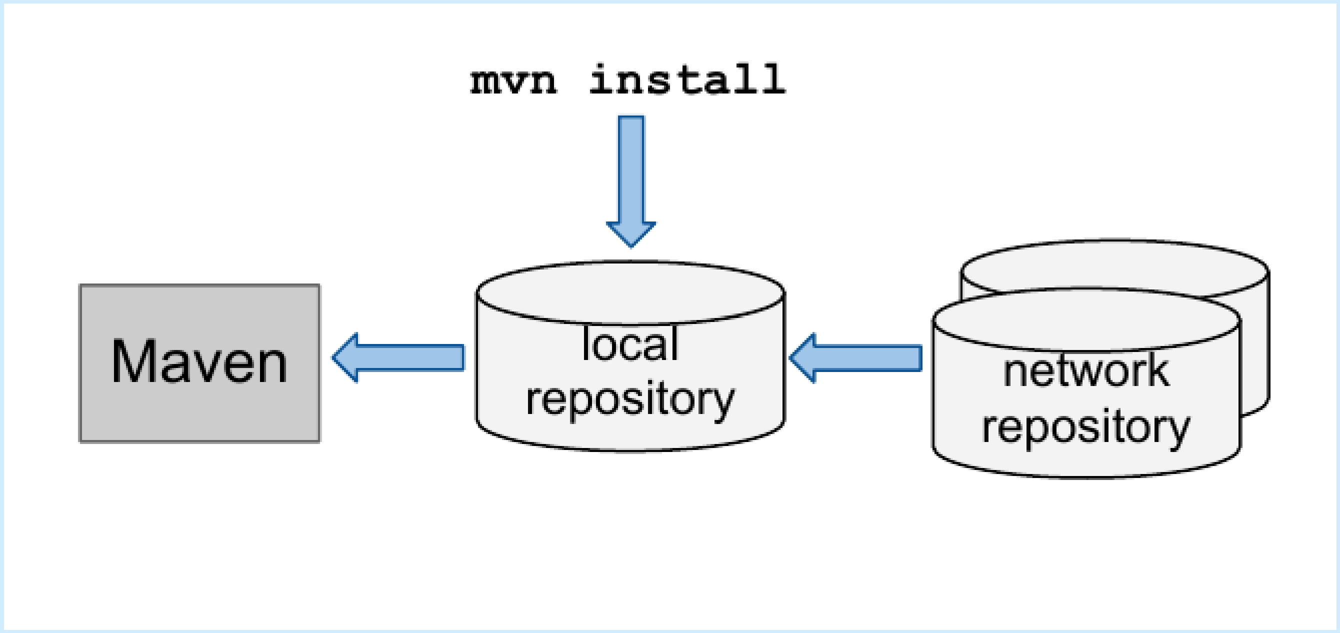


Local Repository



```
${user.home}/
 └─ .m2/
     └─ repository/
         └─ org/
             └─ testng/
                 └─ testng/
                     └─ 5.11/
                         └─ testng-5.11-jdk15.jar
```

Local Repository Population



Maven Central

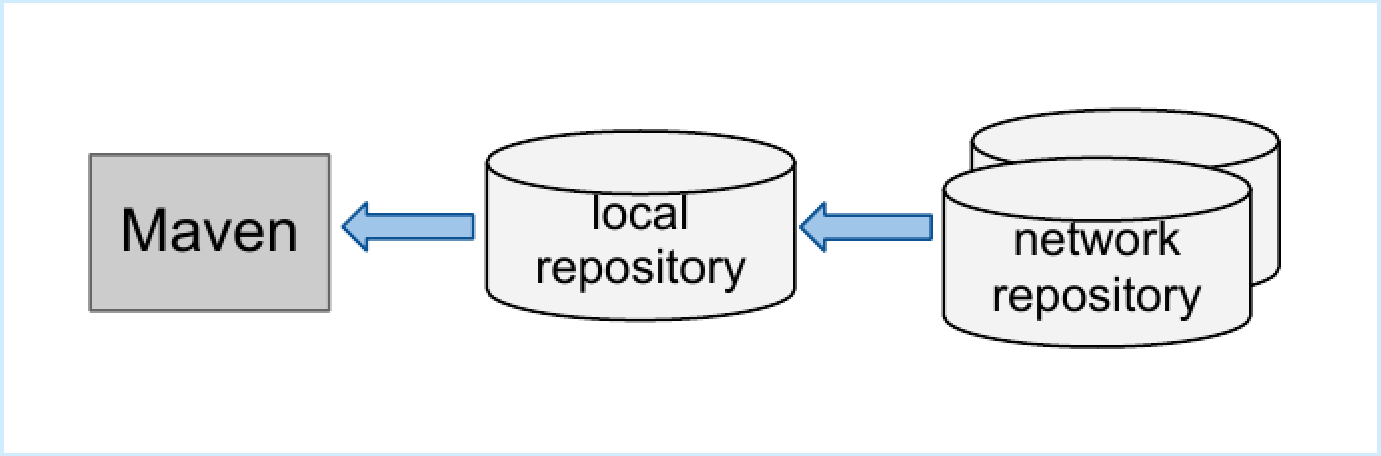
http://repo1.maven.org/maven2

Number of artifacts (GAV): **313,955**

Number of unique artifacts (GA): **38,372**

Size of repository: **457,310 MB**

Artifact Resolution



Using External Libs

1. Choose a library
2. Find Maven coordinates
3. Add `<dependency>` to POM
4. Use it in your code

External Lib Selection

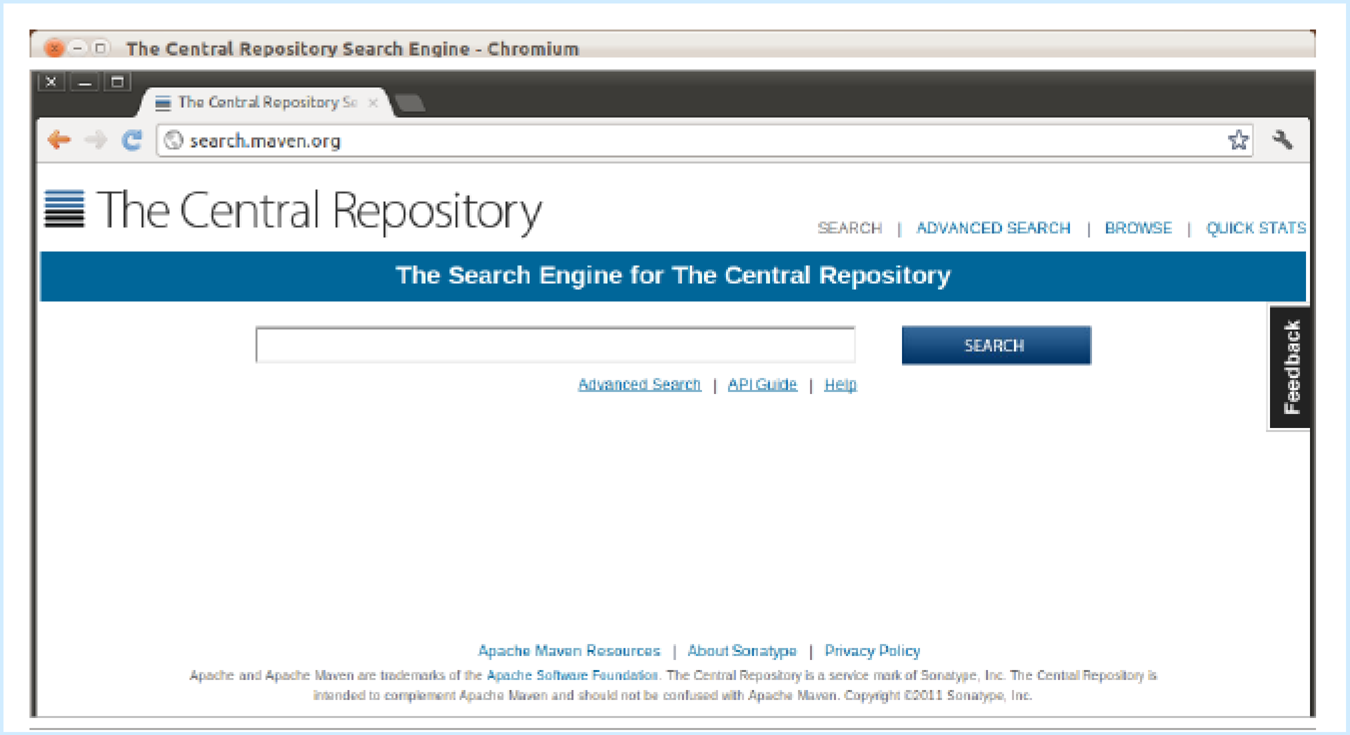
1. Good quality
2. Mature
3. Supported
4. Local expertise
5. Has a reliable source

Where to Find Maven Coordinates

1. Maven Central search
2. Project documentation / site
3. Google
4. Not found? Do not use this crap.

Maven Central Search

http://search.maven.org/


The Central Repository Search Engine - Chromium
The Central Repository Se
search.maven.org

The Central Repository

SEARCH | ADVANCED SEARCH | BROWSE | QUICK STATS

The Search Engine for The Central Repository

Advanced Search | API Guide | Help

Apache Maven Resources | About Sonatype | Privacy Policy

Apache and Apache Maven are trademarks of the Apache Software Foundation. The Central Repository is a service mark of Sonatype, Inc. The Central Repository is intended to complement Apache Maven and should not be confused with Apache Maven. Copyright ©2011 Sonatype, Inc.

POM: Add Dependency

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:x
...
<dependencies>
     ...
     <dependency>
         <groupId>javax.mail</groupId>
         <artifactId>mail</artifactId>
         <version>1.4.5</version>
     </dependency>
     ...
</dependencies>
...
</project>
```

Maven Talks: Next

1. Dependency management explained
2. IDE (Eclipse) integration
3. Testing with Maven
4. Building web applications
5. Serving static content
6. Using binary libs