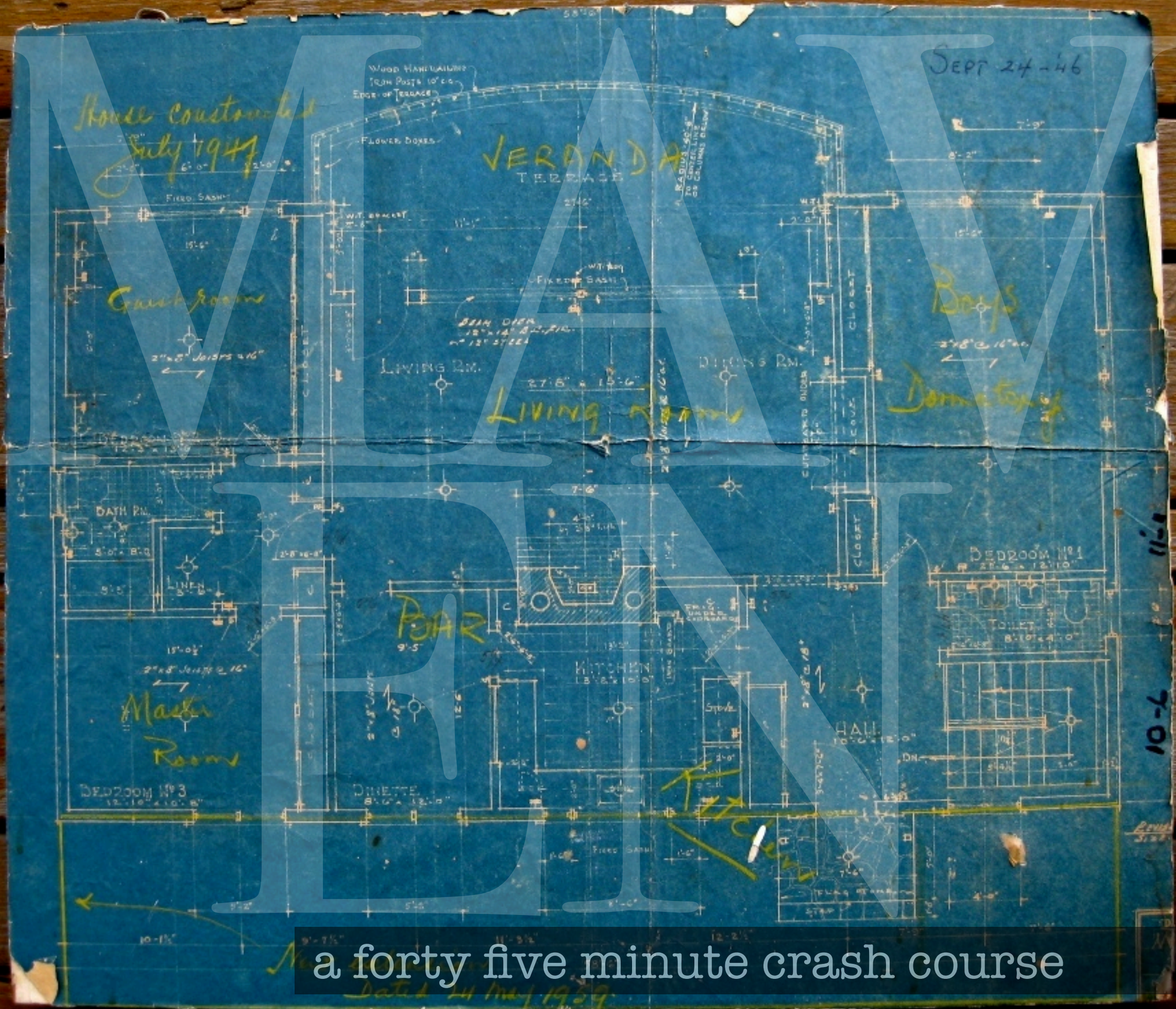


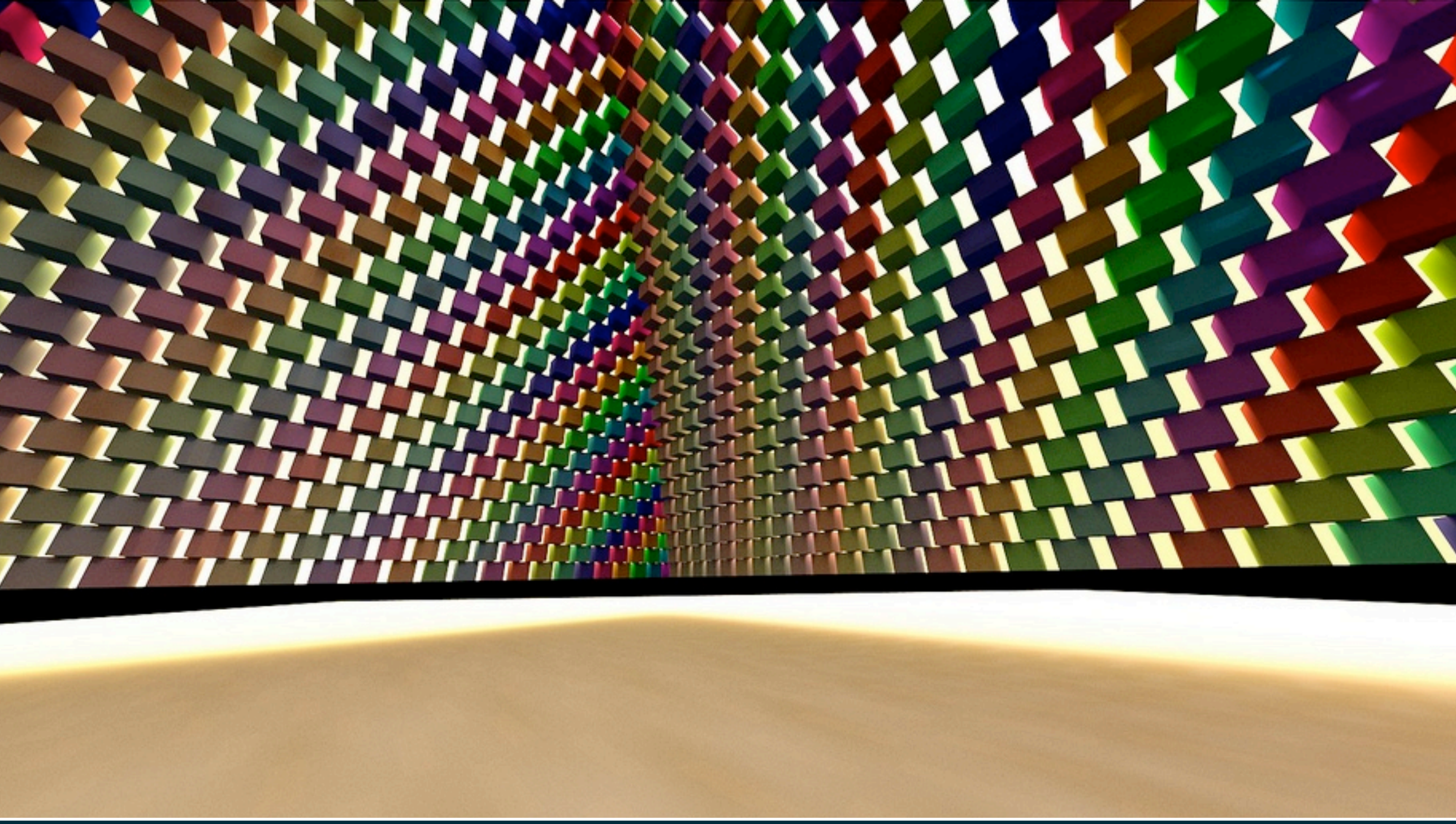
a forty five minute crash course

1. definition

Maven is a build automation tool.

It’s built in Java and lives under the Apache umbrella,
(∴ still new to Flex developers)

it promotes **convention over configuration**



and, it is **hierarchical.**

pom.xml # embed raw

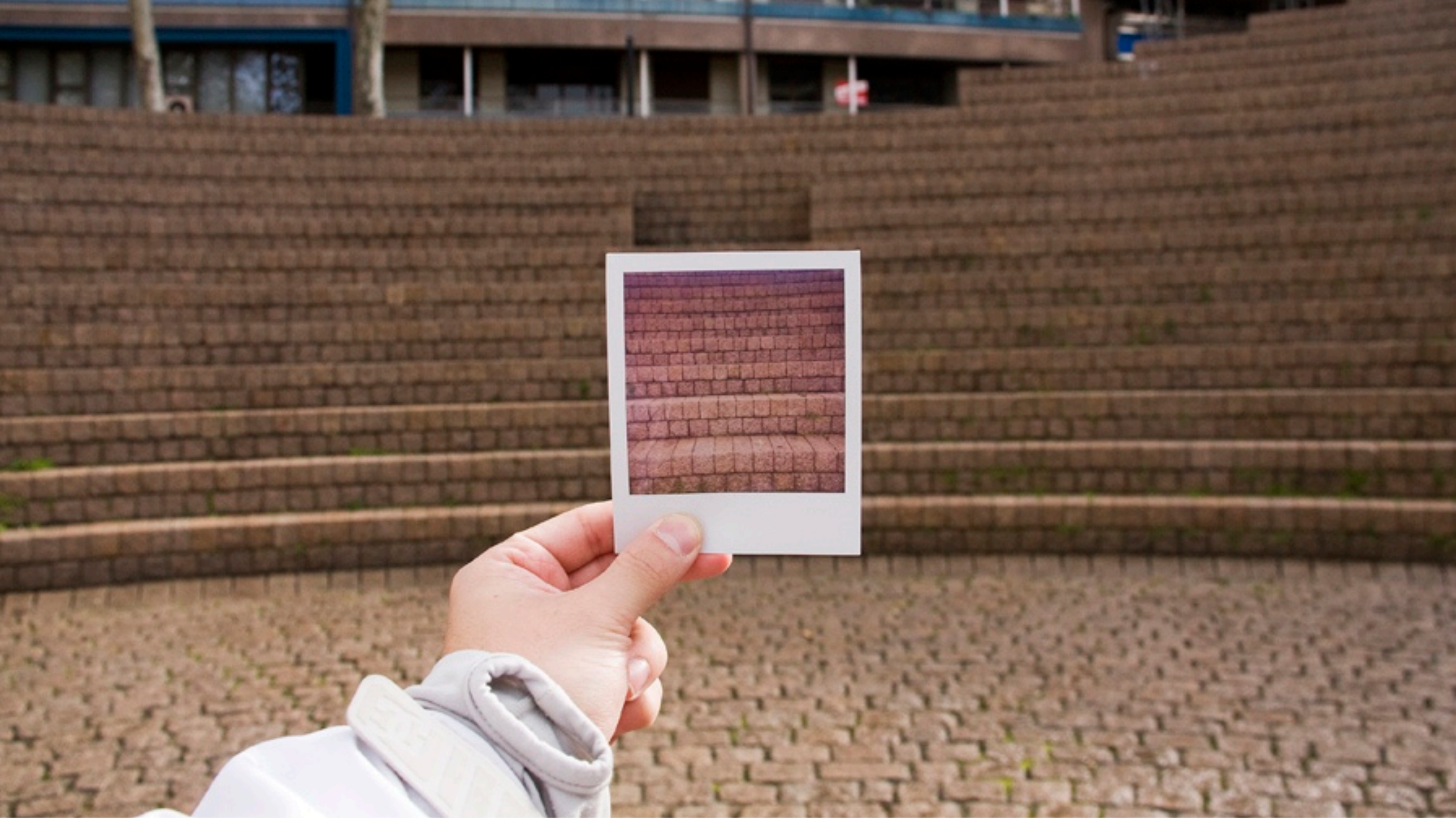
```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4_0_0.xsd">

        <modelVersion>4.0.0</modelVersion>
        <groupId>org.justinjmoses.examples</groupId>
        <artifactId>closures-cleaning-example</artifactId>
        <version>1.0-SNAPSHOT</version>

        <packaging>swf</packaging>

        <properties>
                <flex.version>4.5.1.21328</flex.version>
        </properties>

        <dependencies>
                <dependency>
                        <groupId>com.adobe.flex.framework</groupId>
                        <artifactId>flex-framework</artifactId>
                        <version>${flex.version}</version>
                        <type>pom</type>
                </dependency>
                <dependency>
                  <groupId>com.adobe.flex.framework</groupId>
                  <artifactId>spark</artifactId>
                  <version>${flex.version}</version>
                  <classifier>theme</classifier>
                  <type>css</type>
                  <scope>theme</scope>
                </dependency>
        </dependencies>
</project>
```

Everything builds via a **Project Object Model** (POM): aka the blueprint.

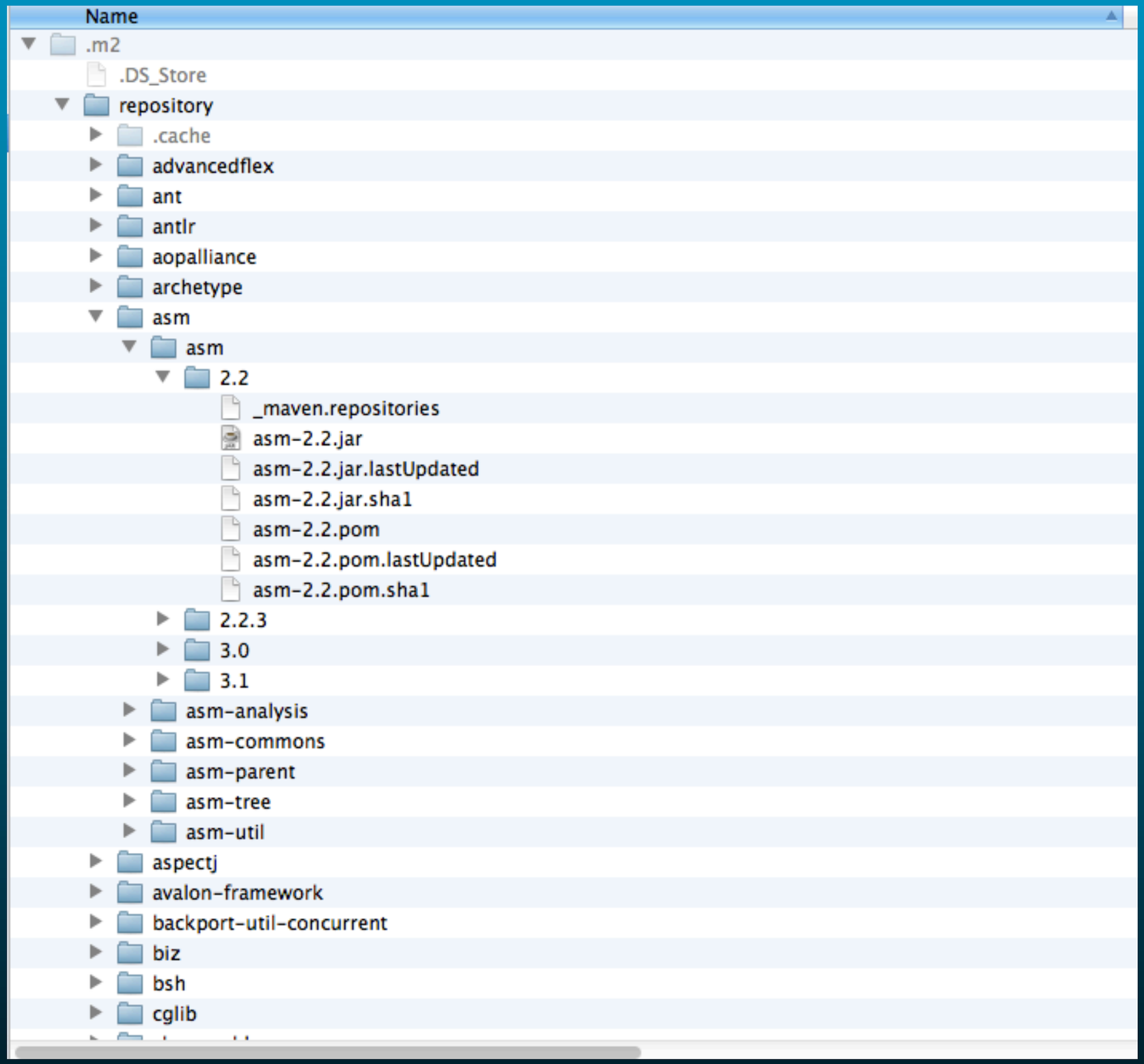
A build generates one or more **artifacts**.

(Typically one artifact per POM)

Artifacts are classified by groupId, artifactId, packaging (type) and version.



Versions stamped with **SNAPSHOT** are treated as such.



Artifacts are filed away in repositories using:

/[groupId]/[artifactId]/[version]/[artifactId]-[version].[ext]

eg. /org/sonatype/flexmojos/4.0-SNAPSHOT/flexmojos-4.0-SNAPSHOT.swc

2. building

A **goal** is a single action.

A **phase** is a collection of goals. [M:M]

A **lifecycle** is a sequence of phases (upto & including).

The odd couple: **clean** & **install**.

```
> mvn clean install
```

The **packaging** (JAR, SWC, etc) typically defines the goals within each phase.

In general, the default lifecycle involves the following phases:

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

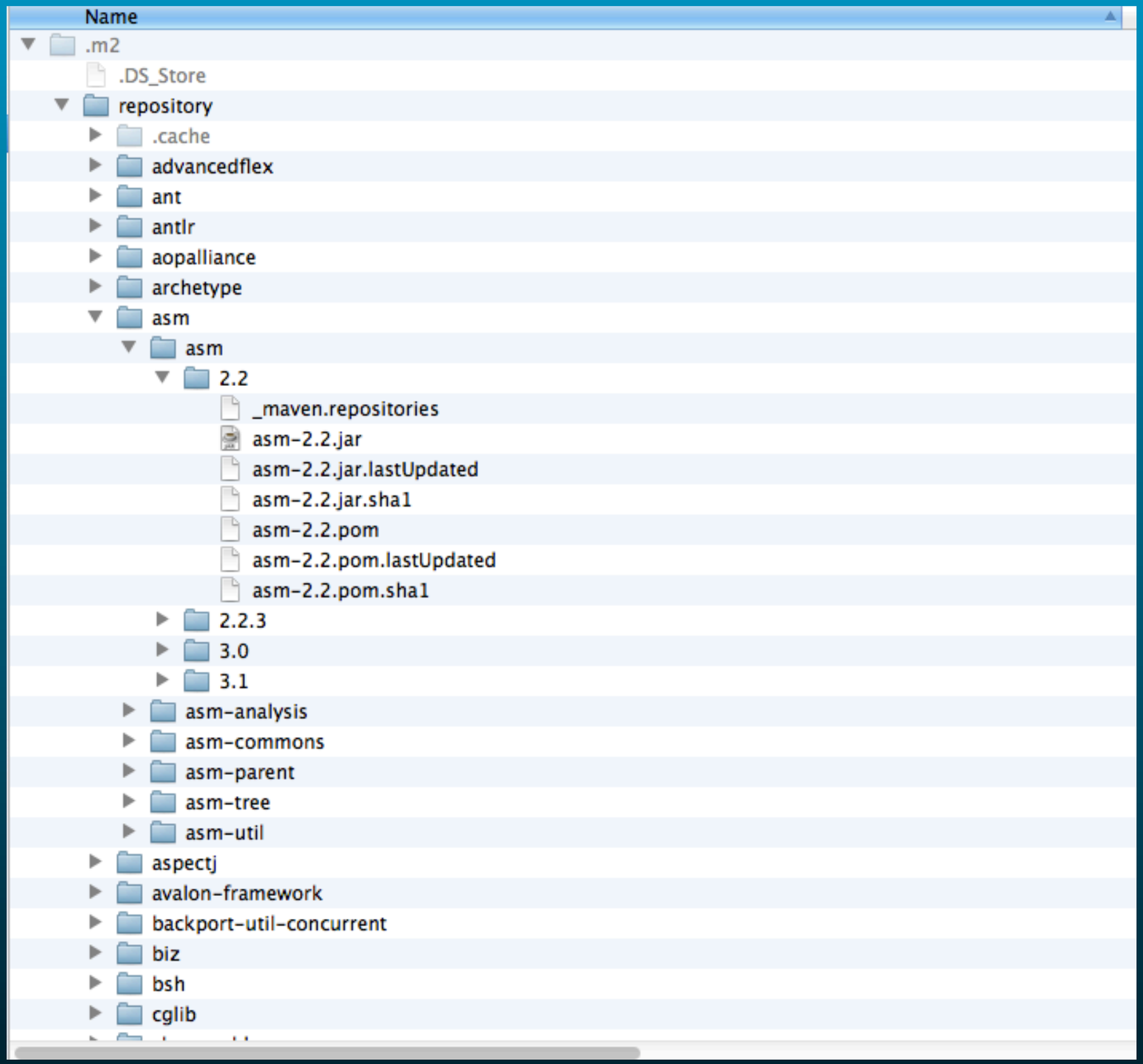
In addition, **plugins** provide goals and can bind them to phases.

eg. `> mvn compiler:compile compiler:testCompile`

Plugins include compiler, install, scm, release, javadoc, eclipse, etc.

3. dependencies

Dependencies are hosted in **repositories.**



Each install of Maven has a repository.
(**~/.m2** is your new best friend).

In order to add new dependencies to your repository, you can simply

```
> mvn install:install-file
```

```xml
<?xml version="1.0" encoding="UTF-8"?>
<settings>
    <profiles>
     <profile>
        <id>my-nexus</id>
        <repositories>
            <repository>
                    <id>justinjmoses-mbp-repository</id>
                    <url>http://localhost:8081/nexus/content/</url>
                    <releases> <enabled>true</enabled> </releases>
                    <snapshots> <enabled>true</enabled> </snapshots>
            </repository>
        </repositories>
     </profile>
     <profile>
        <id>flex-mojos</id>
            <repositories>
                <!-- Required for regular plugin use until moved over to Maven central repo. --
                <repository>
                    <id>flex-mojos-repository</id>
                    <url>http://repository.sonatype.org/content/groups/flexgroup/</url>
                    <releases> <enabled>true</enabled> </releases>
                    <snapshots> <enabled>true</enabled> </snapshots>
                </repository>

                <!-- Required for building plugin from source. -->
                <repository>
                    <id>flex-mojos-internal-repository</id>
                    <url>http://repository.sonatype.org/content/groups/public/</url>
                    <releases> <enabled>true</enabled> </releases>
                    <snapshots> <enabled>true</enabled> </snapshots>
                </repository>
            </repositories>
```

You add references to repositories either in your POM or in your **settings.xml** file.

When building, if a dependency is missing,
Maven will try to download it from an
upstream repository.

How are dependencies shared across repositories?

Nexus.

Nexus allows an organisation to share artifacts
both internally and externally.

It has out-of-the-box support for both
snapshot and **release** repositories.

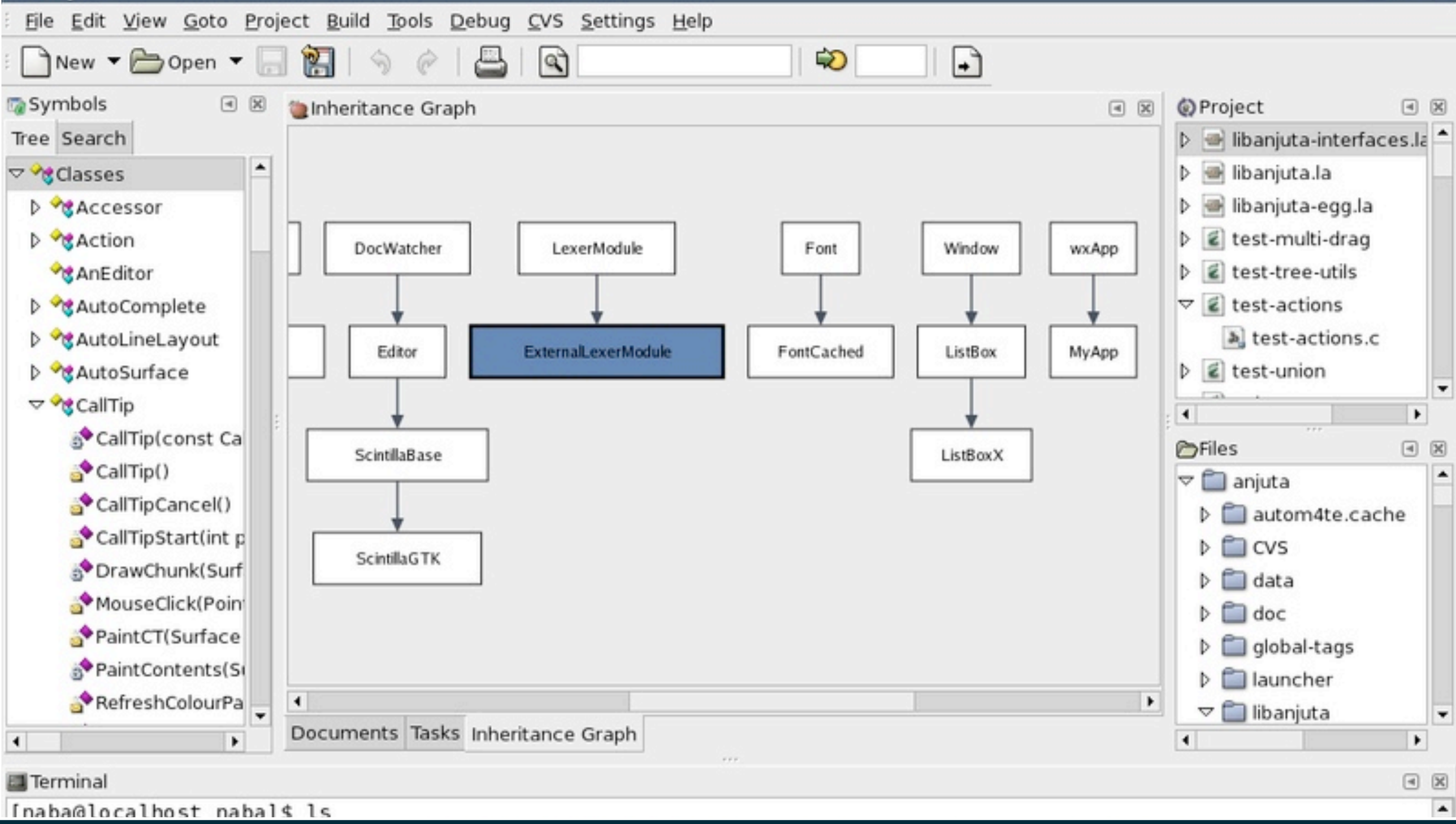
So, what about dependencies on teh internez?

Nexus will proxy to other external Nexus repositories.

If the dependency isn't hosted - there's a **3rd Party** repo that you can upload into.

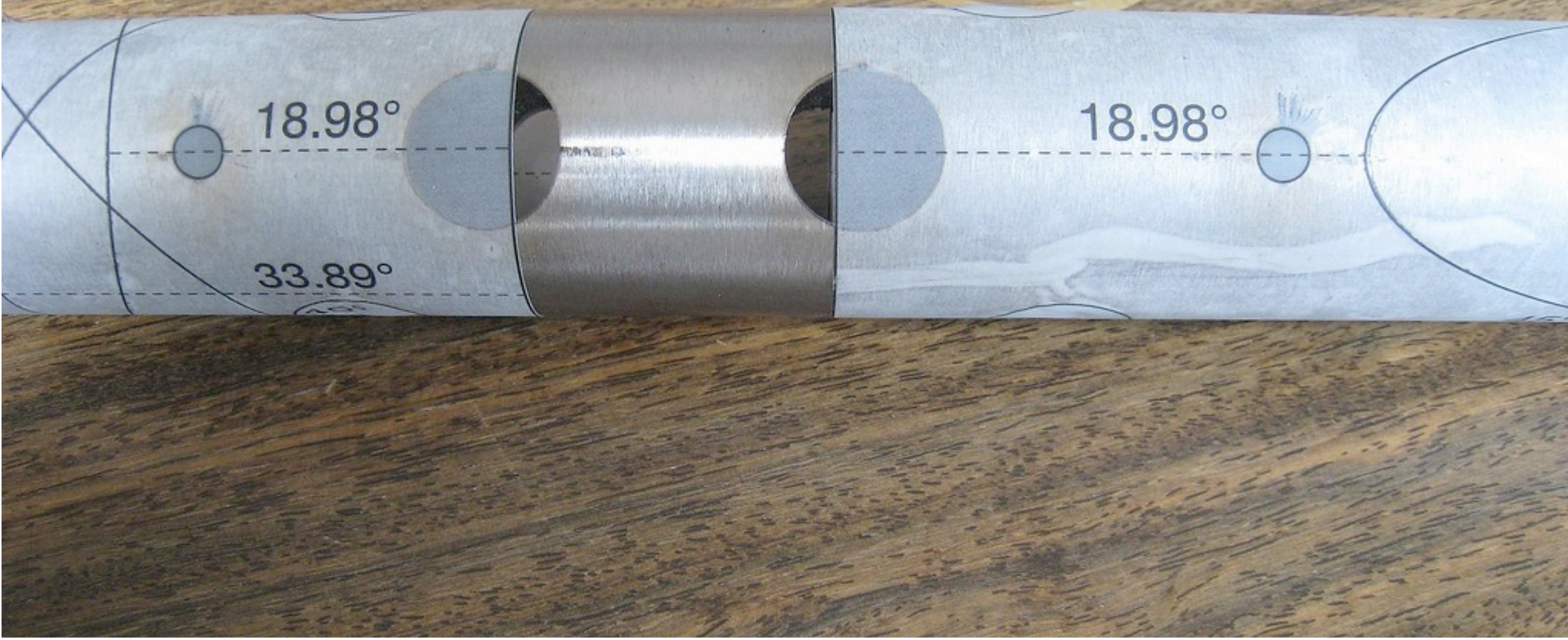
4. configuration

In order to build multiple artifacts, create **modules** for each artifact and one parent POM.



Maven can also create IDE projects from the POM.

(never check in a .project file or dependency again)



It also allows you to create **archetypes** as new project templates.

You can use **profiles** to customize the build. They can activate based on environment &/or set conditions.

5. denouement

Maven can be hours of endless frustration,
yet somehow it’s all worth it.

fin.