Content

- What is Maven
- Build Tools
- Various Build Tools
- Ant Vs Maven Vs Gradle

Problem area

- **Adding** set of **Jars** in each project
- **Creating** the right **project structure**
- **Building** and **Deploying** the project

◆ Large software projects usually contain tens or even hundreds of projects/modules
◆ Will become messy and incomprehensible the projects don’t adhere to some common principles
◆ Will be time time-consuming consuming to build all projects manually

The preferred solution

- Use a project management tool (like Ant,Maven,Gradle)
- Maven helps you with various aspects:

  I. **Build process**
  II. **Project structure**
  III. **Dependency management**
  IV. **Access to information and documentation**

What is Maven

- A Java ***project management*** and ***integration build*** tool.
- Based on the concept of XML Project Object Model (POM).
- Originally developed for building Turbine.
- A small core with numerous plugins (in Jelly).

What Maven does?

Apache Maven helps to manage

- Builds (Build Process)
- Documentation
- Reporing
- SCMs(Source Code Management)
- Distribution

Build Process

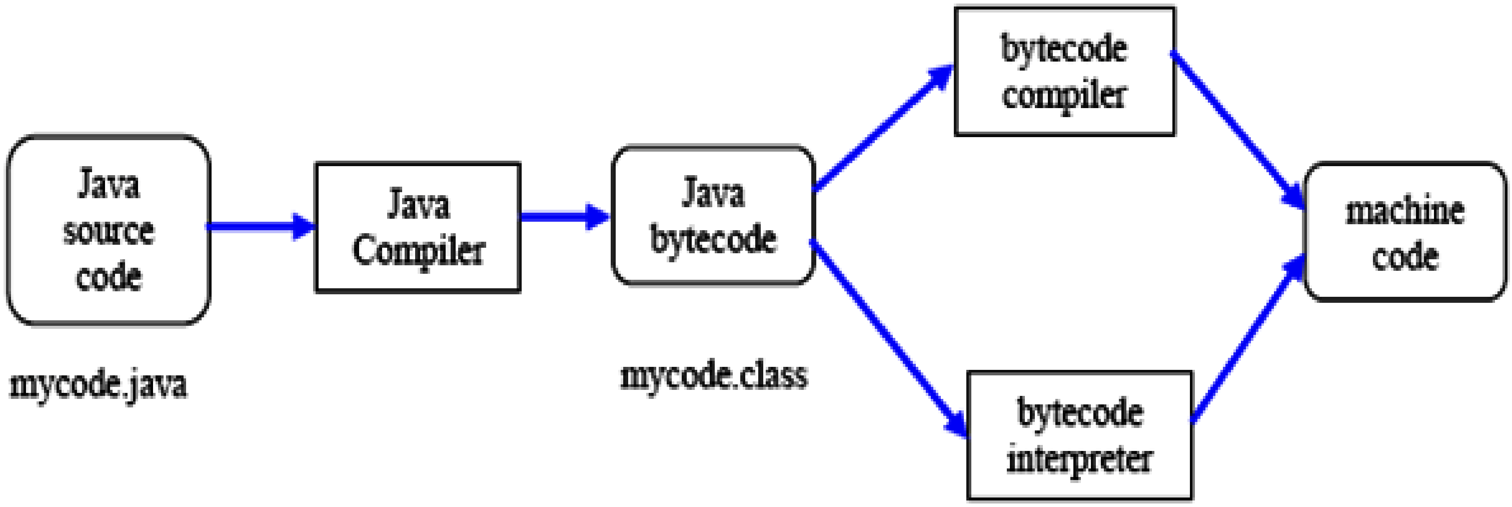
The "Build" is a process that **covers all the steps** required to create a **deliverable product** of your software into preproduction and production .
In the Java world, this typically includes:

1. Generating source.
2. Compiling sources.
3. Executing tests (unit tests, integration tests, etc).
4. Packaging (into jar, war, ejb-jar, ear).
5. Running health checks (static analyzers like CheckstylePMD, test coverage, etc).
6. Generating reports.

Note:

A defined build process is an essential part of any ***development cycle*** because it helps close the gap between the **development**, **integration, test**, and **production** environments.

Compilation and Execution



Various build tools available:

- For java – Ant,Maven,Gradle.
- For .NET framework – Nant
- c# - MsBuild.

SetUp and Installation for Maven

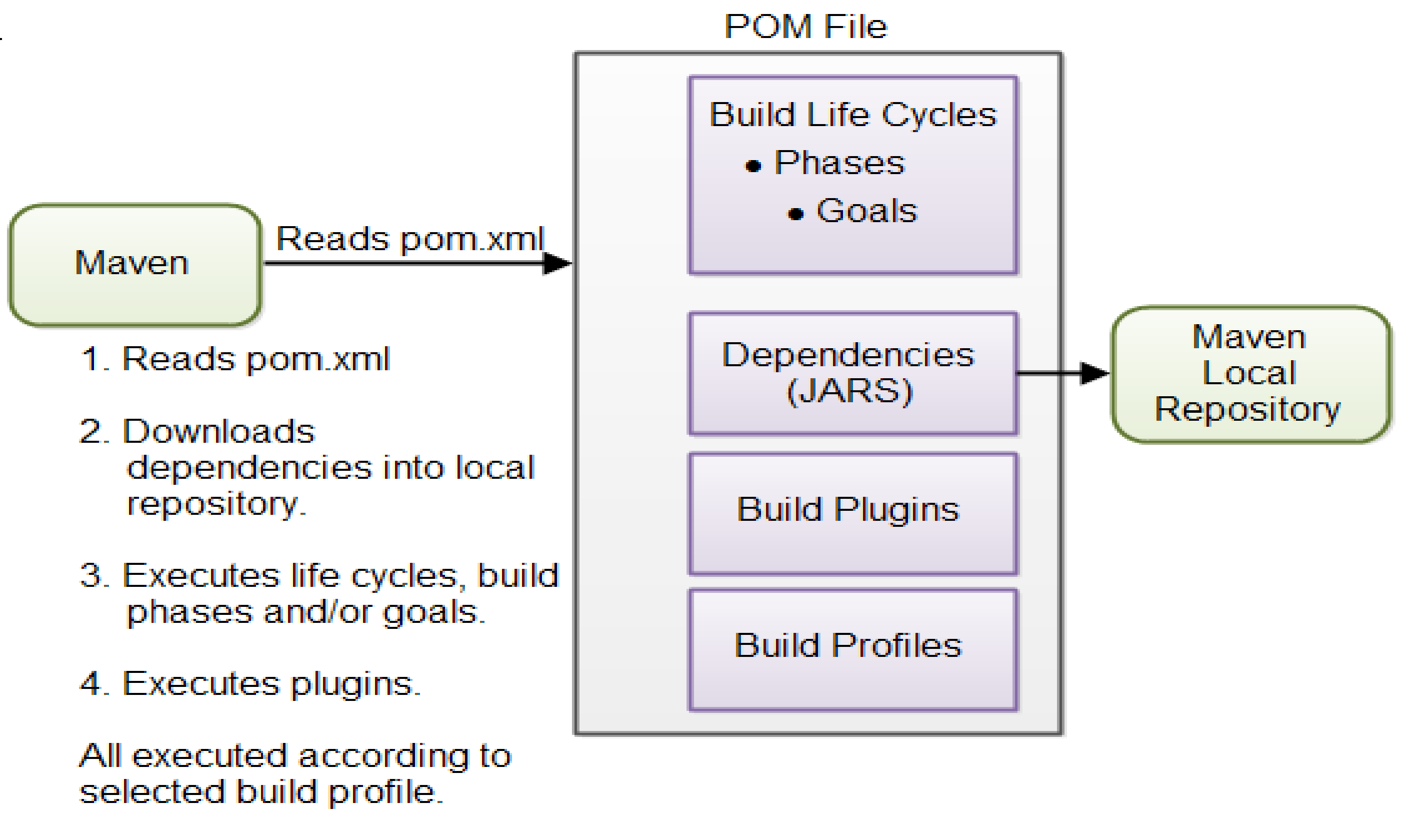
Note: Maven is Java based tool, so the very first requirement is to have **JDK installed** on your machine(for Maven Java is prerequisite).

Download Maven archive

https://maven.apache.org/download.cgi

OS	Archive name
Windows	apache-maven-3.5.0-bin.zip
Linux	apache-maven-3.5.0-bin.tar.gz **or** sudo apt-get install maven (debian based)
Mac	apache-maven-3.5.0-bin.tar.gz

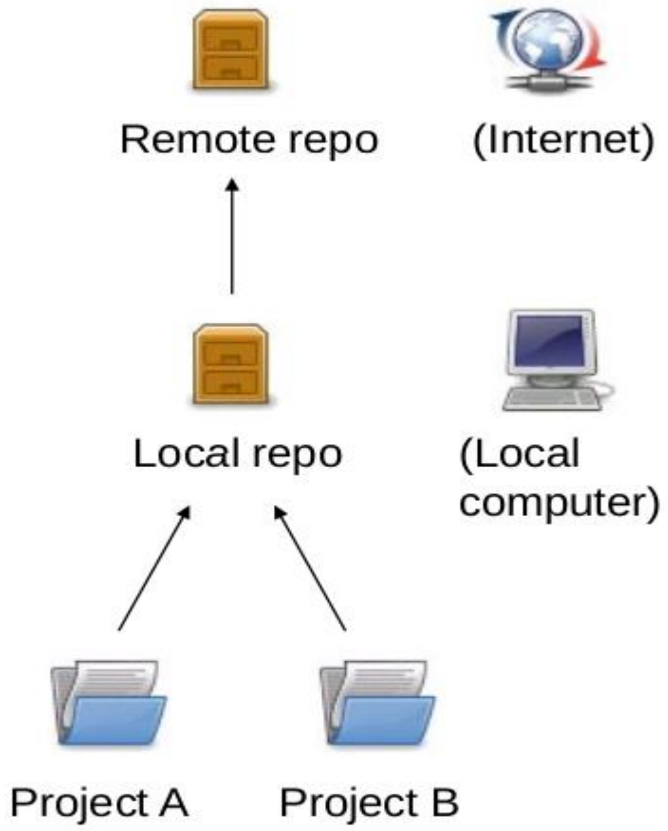
Working of maven



1. Reads pom.xml
2. Downloads dependencies into local repository.
3. Executes life cycles, build phases and/or goals.
4. Executes plugins.

All executed according to selected build profile.

Repositories



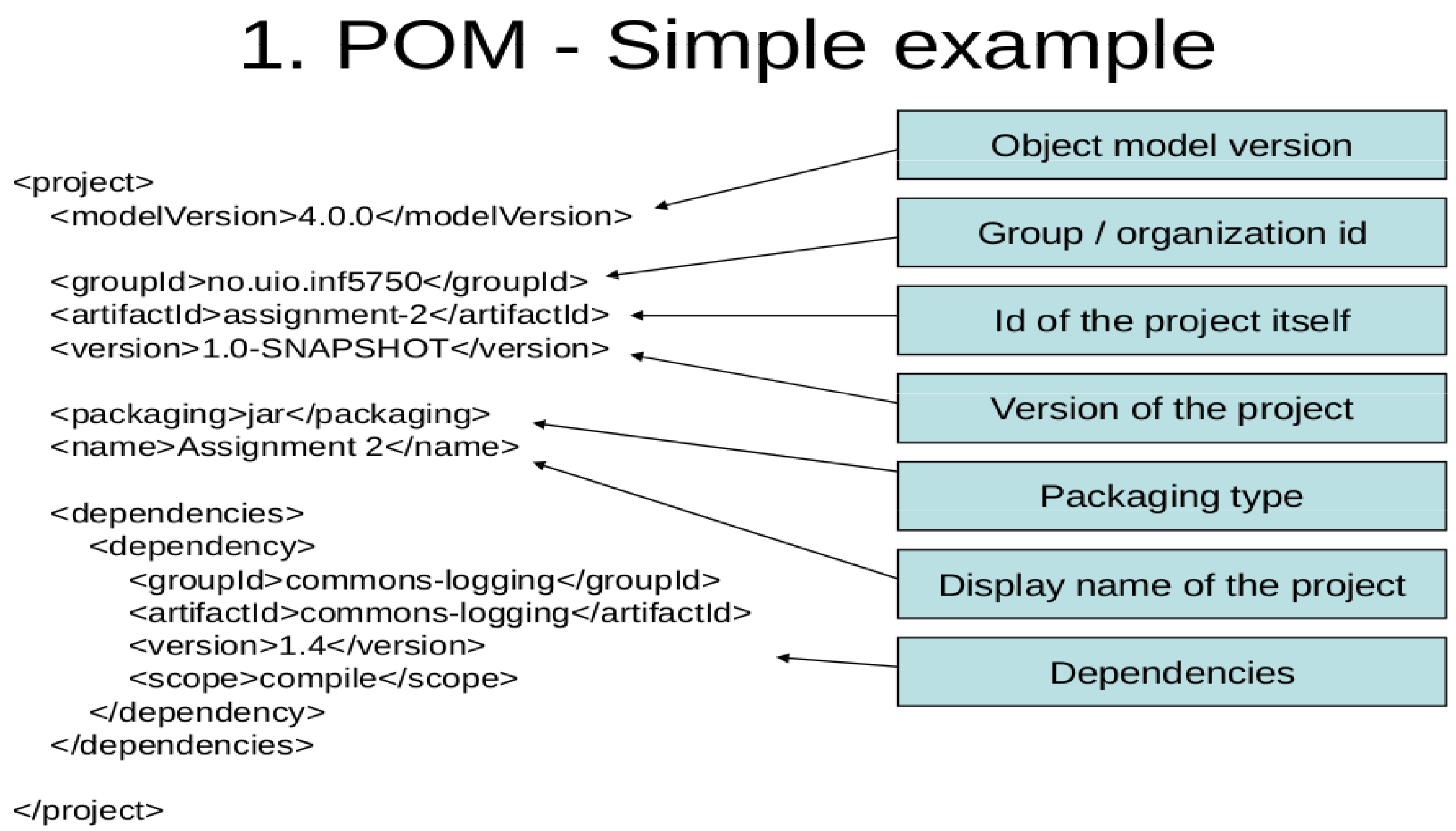
- Remote repository:
  - Provides software artifacts (dependencies) for download
  - E.g. repo1.maven.org houses Maven's central repository
- Local repository:
  - Copy on local computer which is a cache of the remote downloads
  - May contain project-local build artifacts as well
  - Located in USER_HOME/.m2/repository
  - Same structure as remote repos

Dependencies

- Affects the classpath used for various build tasks
- Can be defined for all dependencies, *compile* default
- 5 dependency scopes available:
  - Compile: Available in all classpaths (default)
  - Provided: The JDK or the container provides it
  - Runtime: Only required for execution, not for compilation
  - Test: Only required for testing, not for normal use (not deployed)
  - System: You provide it locally, not looked up in a repo

```
<dependency>
   <groupId>commons-logging</groupId>
   <artifactId>commons-logging</artifactId>
   <version>1.4</version>
   <scope>compile</scope>
</dependency>
```

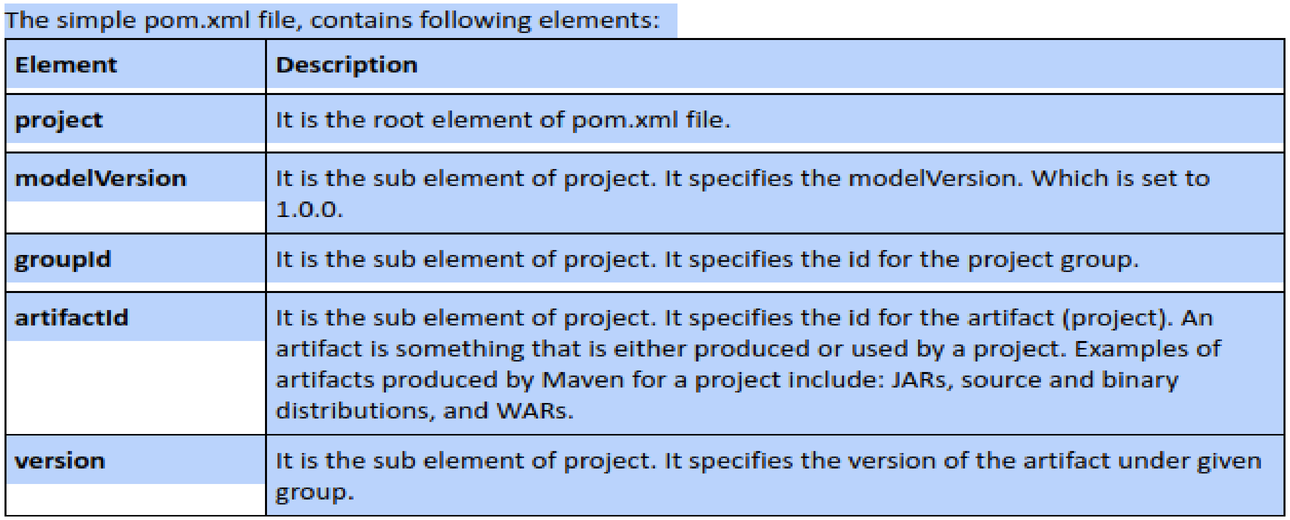
Pom.xml Structure



1. POM - Simple example

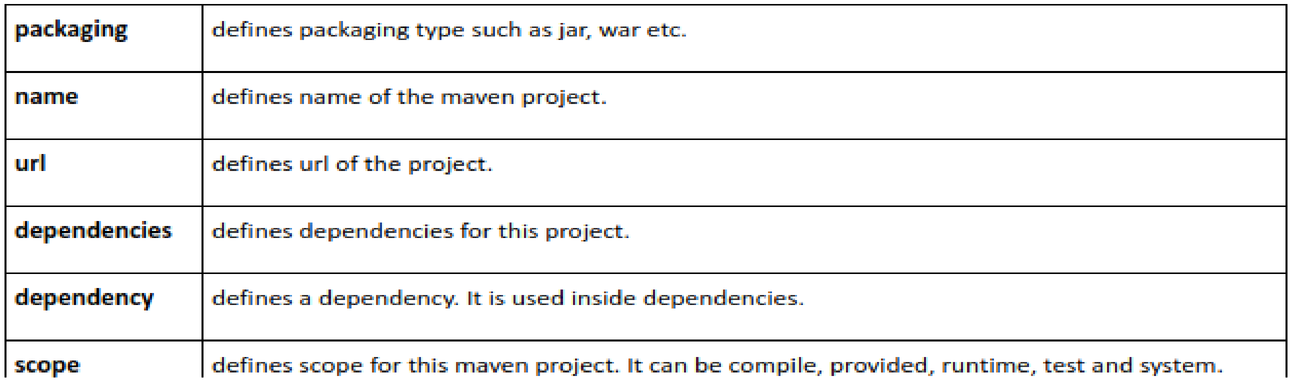
```xml
<project>
    <modelVersion>4.0.0</modelVersion>

    <groupId>no.uio.inf5750</groupId>
    <artifactId>assignment-2</artifactId>
    <version>1.0-SNAPSHOT</version>

    <packaging>jar</packaging>
    <name>Assignment 2</name>

    <dependencies>
        <dependency>
            <groupId>commons-logging</groupId>
            <artifactId>commons-logging</artifactId>
            <version>1.4</version>
            <scope>compile</scope>
        </dependency>
    </dependencies>

</project>
```

Pom.xml Description



The simple pom.xml file, contains following elements:

Element	Description
project	It is the root element of pom.xml file.
modelVersion	It is the sub element of project. It specifies the modelVersion. Which is set to 1.0.0.
groupId	It is the sub element of project. It specifies the id for the project group.
artifactId	It is the sub element of project. It specifies the id for the artifact (project). An artifact is something that is either produced or used by a project. Examples of artifacts produced by Maven for a project include: JARs, source and binary distributions, and WARs.
version	It is the sub element of project. It specifies the version of the artifact under given group.



packaging	defines packaging type such as jar, war etc.
name	defines name of the maven project.
url	defines url of the project.
dependencies	defines dependencies for this project.
dependency	defines a dependency. It is used inside dependencies.
scope	defines scope for this maven project. It can be compile, provided, runtime, test and system.

Maven Build Lifecycle and Phases

The build lifecycle is the process of building and distributing an artifact

- A phase is a step in the build lifecycle
- Most important default phases:

✓ Validate
✓ Compile
✓ Test
✓ Package
✓ Install
✓ Deploy

- Some common phases not default:

✓ Clean
✓ Site

- For each step, step all previous steps are executed

Maven Build Phases

Validate - validate the project is correct and all necessary information is available

Compile - compile the source code of the project

test - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed

Package - take the compiled code and package it in its distributable format, such as a JAR.

Verify - run any checks on results of integration tests to ensure quality criteria are met

Install - install the package into the local repository, for use as a dependency in other projects locally

Deploy - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

Thank You !!