Contents

[Lesson 1. Intro 2](#_Toc71974739)

[Lesson 2. Installing 2](#_Toc71974740)

[Lesson 3. Plugins and MOJO 2](#_Toc71974741)

[Lesson 4. JVM arguments, Program arguments, Environment variables 2](#_Toc71974742)

[Lesson 5. Archetype plugin 3](#_Toc71974743)

[Lesson 6. POM Project Object Model 3](#_Toc71974744)

[Lesson 7. Effective POM 3](#_Toc71974745)

[Lesson 8. Dependency management 3](#_Toc71974746)

[Lesson 9. Dependency scope, Dependency plugin 4](#_Toc71974747)

## Lesson 1. Intro

Before build tools projects were built by batch/shell scripts. It had drawbacks:

* Platform dependence (unix/windows required different scripts)
* Scripts from on project didn’t fit to another
* No uniform project structure (more onboarding time when you come on new project)

2000 – released Apache Ant which were more convenient than scripts.

2002 – released Apache Maven which was better because supplied uniformed project structure.

2007 – released Gradle which can use Groovy/Kotlin instead of xml.

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 2. Installing

About installing of environment variables (JAVA\_HOME, JDK) for unix/windows and about apache maven in the video. About installing of Maven Wrapper in the last video of the course.

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 3. Plugins and MOJO

Maven is just a set of plugins. Each plugin is a separate java project. Plugins contain commands (goals).

Goals are MOJO (Maven plain Old Java Object) – ordinary java classes with method execute(). So we can create own goals and plugins.

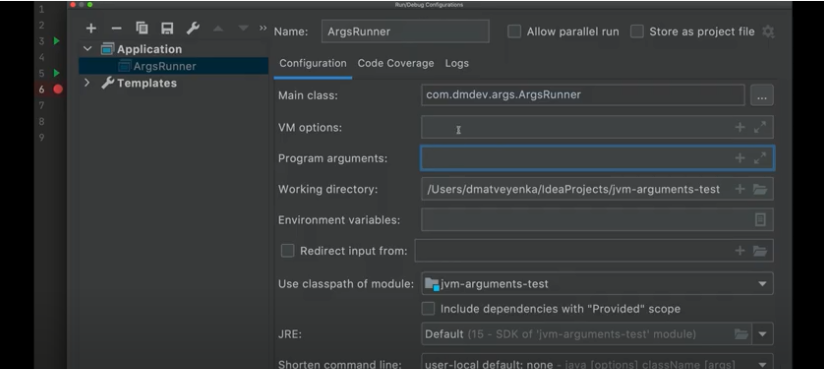
mvn plugin:goal (e.g. mvn compiler:compile)

Each plugin has goal “help”.

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 4. JVM arguments, Program arguments, Environment variables

IDEA->Run->Edit Configuration



Program arguments = arguments of main()

VM options = JVM arguments:

* User args Dkey=value
* Reserved args XkeyValue Xms512m – heap size during program start

XX… for tuning JIT compiler or garbage collector

To look at arguments passed to our program:

IDEA->DEBUG->evaluate expression->System.getProperties();

To look at environment variables:

IDEA->DEBUG->evaluate expression->System.getEnv(name:);

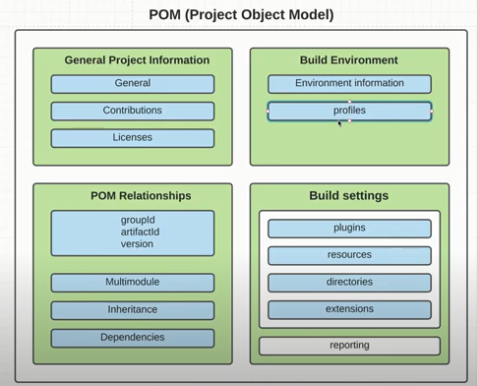
-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 5. Archetype plugin

Generates project of required structure

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 6. POM Project Object Model



-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 7. Effective POM

All POM files inherit super POM (like object in java) and parent POM files. To see result POM (effective) there is a goal:

mvn help:effective-pom

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 8. Dependency management

Classpath – directory where all our dependencies (.jar) are stored. They are not in our project but in maven local repository in our PC (~/.m2/repository). If we add a new dependency in our project maven first look for it in local repo and then in remote repo.

${maven}/conf/settings.xml – here we can change path to local repo (better not to do it)

-----------------------------------------------------------------------------------------------------------------------------------------------------------

## Lesson 9. Dependency scope, Dependency plugin

|  |  |  |
| --- | --- | --- |
| TYPE | DESCRIPTION | EXAMPLE |
| Compile (Default) | Dependency is needed to compile our project |  |
| Provided | By another dependency. | Jackarta.servlet-api is provided by Tomcat |
| Runtime | Dependency is needed in runtime. | Database driver |
| System | Jar is on our pc. | DO NOT USE AT ALL! |
| Test | Dependency is needed during test phase | Junit |

mvn dependency:analyze - shows unused or required dependencies

mvn dependency:tree - tree of dependencies

-----------------------------------------------------------------------------------------------------------------------------------------------------------