Build tools. Introduction to Maven. Examples and usage.

PRAGMATIC IT Learning & Outsourcing Center

www.pragmatic.bg

Lector: Stefan Vadev
Skype: stefanvadev77
E-mail: stefan.vadev@gmail.com
Facebook:
https://www.facebook.com/stefan.vadev.7

Summary

- What is maven
- Our first maven project
- Maven commands
- Maven in Idea

What is maven

Maven™

What is maven

- Maven is a powerful build tool for Java software projects. Actually, you can build software projects using other languages too, but Maven is developed in Java, and is thus historically used more for Java projects.

- Actually, the Maven developers claim that Maven is more than just a build tool. You can read what they believe it is, in their document Philosophy of Maven. But for now, just think of it as a build tool. You will find out what Maven really is, once you understand it and start using it.

What is a build tool

- **What is a Build Tool?**
- A build tool is a tool that automates everything related to building the software project. Building a software project typically includes one or more of these activities:

- Generating source code (if auto-generated code is used in the project).
- Generating documentation from the source code.
- Compiling source code.
- Packaging compiled code into JAR files or ZIP files.
- Installing the packaged code on a server, in a repository or somewhere else.

What is a build tool

- Any given software project may have more activities than these needed to build the finished software. Such activities can normally be plugged into a build tool, so these activities can be automated too.

- The advantage of automating the build process is that you minimize the risk of humans making errors while building the software manually. Additionally, an automated build tool is typically faster than a human performing the same steps manually.

Installing maven

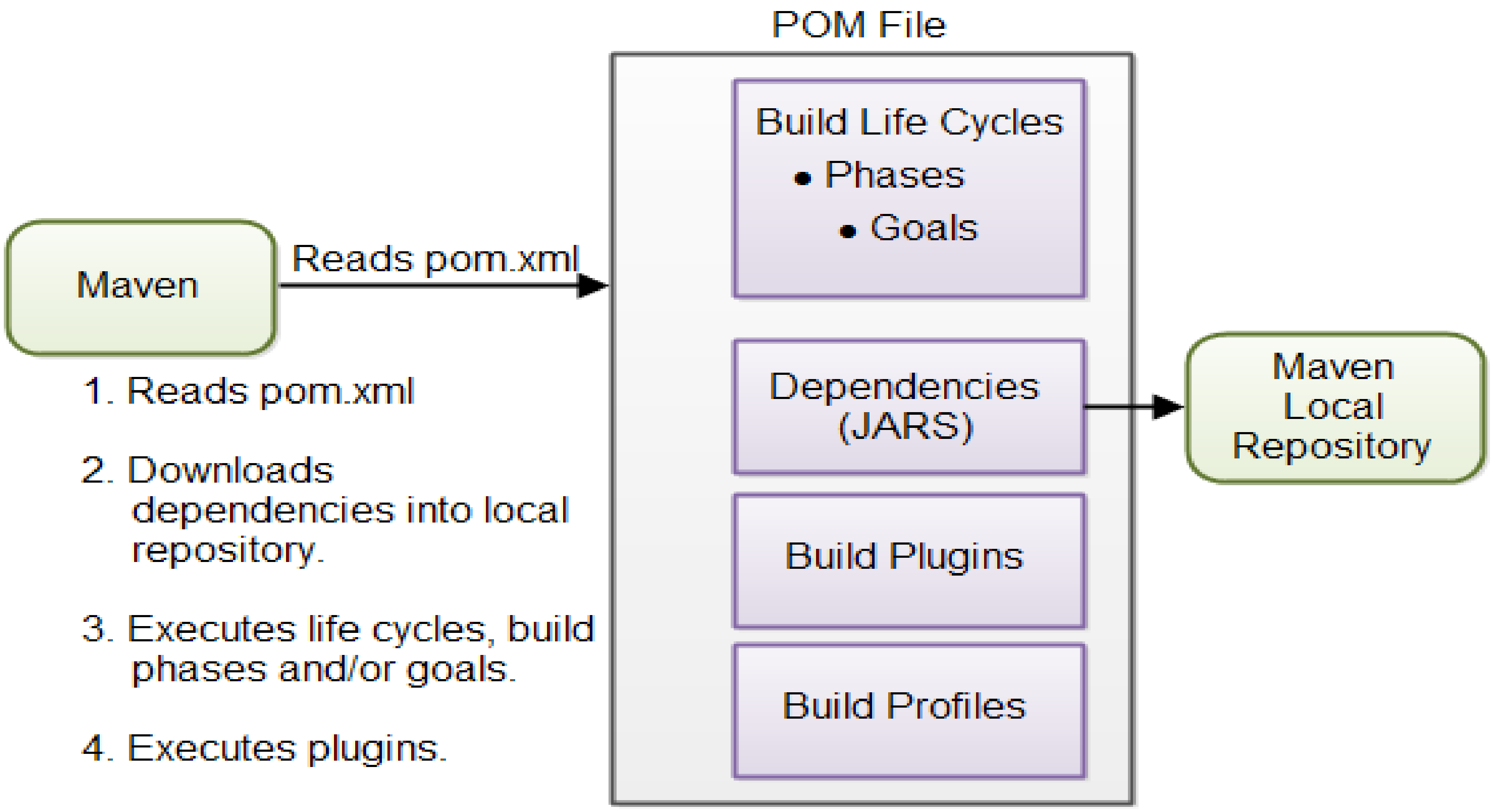
- Set the JAVA_HOME environment variable to point to a valid Java SDK (e.g. Java 8).
- Download and unzip Maven.
- Set the M2_HOME environment variable to point to the directory you unzipped Maven to.
- Set the M2 environment variable to point to M2_HOME/bin (%M2_HOME%\bin on Windows, $M2_HOME/bin on unix).
- Add M2 to the PATH environment variable (%M2% on Windows, $M2 on unix).
- Open a command prompt and type 'mvn -version' (without quotes) and press enter.
- After typing in the mvn -version command you should be able to see Maven execute, and the version number of Maven written out to the command prompt.

Maven overview

- Maven is centered around the concept of POM files (Project Object Model). A POM file is an XML representation of project resources like source code, test code, dependencies (external JARs used) etc. The POM contains references to all of these resources. The POM file should be located in the root directory of the project it belongs to.

- Here is a diagram illustrating how Maven uses the POM file, and what the POM file primarily contains:

Maven overview



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.

Core concepts

POM Files
When you execute a Maven command you give Maven a POM file to execute the commands on. Maven will then execute the command on the resources described in the POM.

Build Life Cycles, Phases and Goals
The build process in Maven is split up into build life cycles, phases and goals. A build life cycle consists of a sequence of build phases, and each build phase consists of a sequence of goals. When you run Maven you pass a command to Maven. This command is the name of a build life cycle, phase or goal. If a life cycle is requested executed, all build phases in that life cycle are executed. If a build phase is requested executed, all build phases before it in the pre-defined sequence of build phases are executed too.

Core concepts

Dependencies and Repositories

One of the first goals Maven executes is to check the dependencies needed by your project. Dependencies are external JAR files (Java libraries) that your project uses. If the dependencies are not found in the local Maven repository, Maven downloads them from a central Maven repository and puts them in your local repository. The local repository is just a directory on your computer's hard disk. You can specify where the local repository should be located if you want to (I do). You can also specify which remote repository to use for downloading dependencies.

Core concepts

Build Plugins
Build plugins are used to insert extra goals into a build phase. If you need to perform a set of actions for your project which are not covered by the standard Maven build phases and goals, you can add a plugin to the POM file. Maven has some standard plugins you can use, and you can also implement your own in Java if you need to.

Build Profiles
Build profiles are used if you need to build your project in different ways. For instance, you may need to build your project for your local computer, for development and test. And you may need to build it for deployment on your production environment. These two builds may be different. To enable different builds you can add different build profiles to your POM files. When executing Maven you can tell which build profile to use.

Maven POM files

A Maven POM file (Project Object Model) is an XML file that describe the resources of the project. This includes the directories where the source code, test source etc. is located in, what external dependencies (JAR files) your projects has etc.

The POM file describes what to build, but most often not how to build it. How to build it is up to the Maven build phases and goals. You can insert custom actions (goals) into the Maven build phase if you need to, though.

Each project has a POM file. The POM file is named pom.xml and should be located in the root directory of your project. A project divided into subprojects will typically have one POM file for the parent project, and one POM file for each subproject. This structure allows both the total project to be built in one step, or any of the subprojects to be built separately.

Maven POM files

Here is a minimal POM file:

```
<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/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>

  <groupId>bg.pragmatic</groupId>
  <artifactId>online-store</artifactId>
  <version>1.0.0</version>
</project>
```

POM files explained

The **modelVersion** element sets what version of the POM model you are using. Use the one matching the Maven version you are using. Version 4.0.0 matches Maven version 2 and 3.

The **groupId** element is a unique ID for an organization, or a project (an open source project, for instance). Most often you will use a group ID which is similar to the root Java package name of the project. For instance, for my Java Web Crawler project I may choose the group ID bg.pragmatic. If the project was an open source project with many independent contributors, perhaps it would make more sense to use a group ID related to the project than an a group ID related to my company. Thus, online-store could be used.

POM files explained

The group ID does not have to be a Java package name, and does not need to use the . notation (dot notation) for separating words in the ID. But, if you do, the project will be located in the Maven repository under a directory structure matching the group ID. Each . is replaced with a directory separator, and each word thus represents a directory. The group ID bg.pragmatic would then be located in a directory called MAVEN_REPO/bg/pragmatic. The MAVEN_REPO part of the directory name will be replaced with the directory path of the Maven repository.

POM files explained

The group ID does not have to be a Java package name, and does not need to use the . notation (dot notation) for separating words in the ID. But, if you do, the project will be located in the Maven repository under a directory structure matching the group ID. Each . is replaced with a directory separator, and each word thus represents a directory. The group ID bg.pragmatic would then be located in a directory called MAVEN_REPO/bg/pragmatic. The MAVEN_REPO part of the directory name will be replaced with the directory path of the Maven repository.

POM files explained

The artifactId element contains the name of the project you are building. In the case of my Java online store project, the artifact ID would be online-store. The artifact ID is used as name for a subdirectory under the group ID directory in the Maven repository. The artifact ID is also used as part of the name of the JAR file produced when building the project. The output of the build process, the build result that is, is called an artifact in Maven. Most often it is a JAR, WAR or EAR file, but it could also be something else.

POM files explained

The versionId element contains the version number of the project. If your project has been released in different versions, for instance an open source API, then it is useful to version the builds. That way users of your project can refer to a specific version of your project. The version number is used as a name for a subdirectory under the artifact ID directory. The version number is also used as part of the name of the artifact built.
The above groupId, artifactId and version elements would result in a JAR file being built and put into the local Maven repository at the following path (directory and file name):

MAVEN_REPO/bg/pragmatic/online-store/1.0.0/online-store-1.0.0.jar

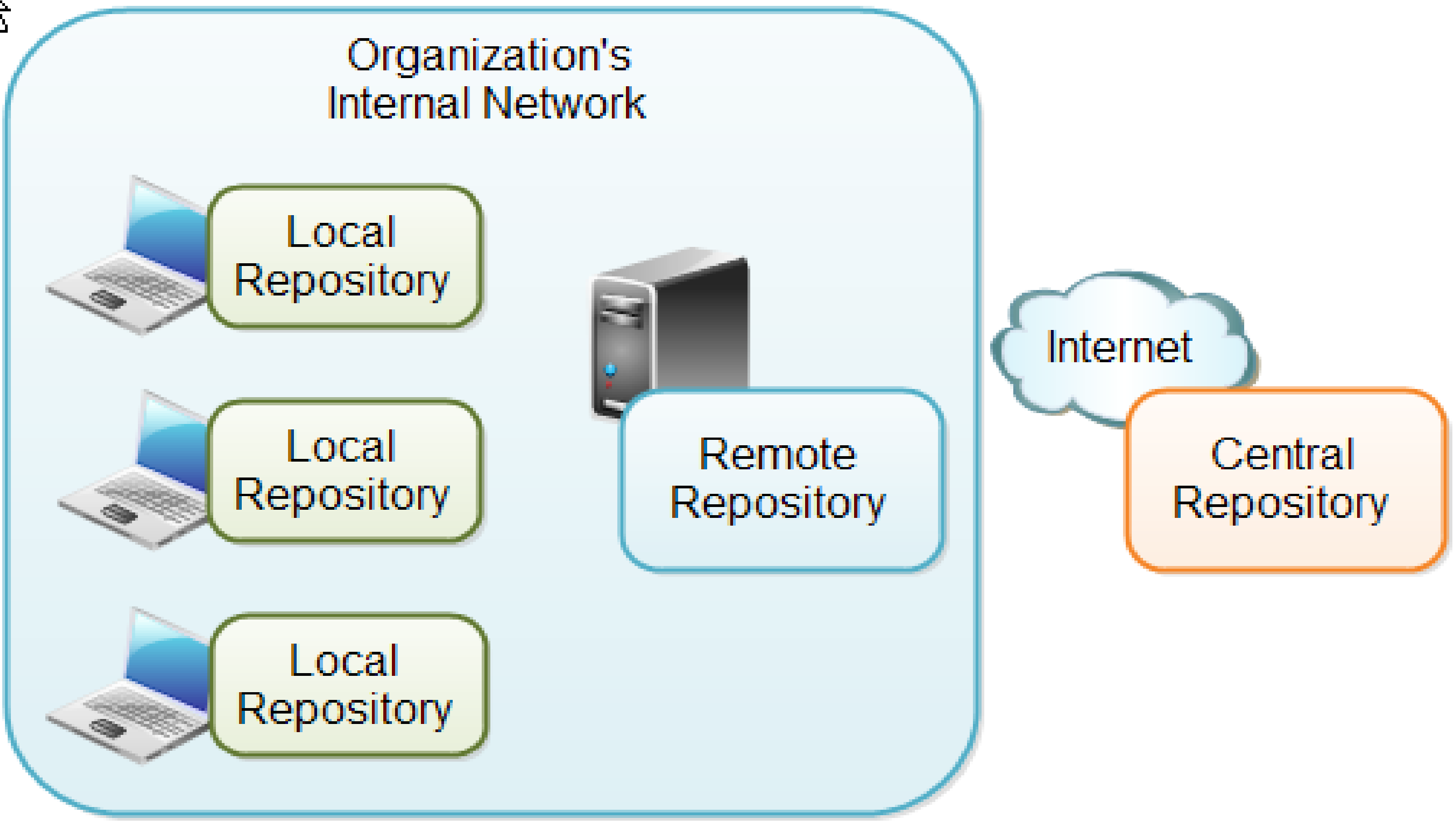
Maven repositories

Maven has three types of repository:

Local repository
Central repository
Remote repository

Maven searches these repositories for dependencies in the above sequence. First in the local repository, then in the central repository, and third in remote repositories if specified in the POM.

Maven repositories



POM files explained

Now let’s continue here:
http://tutorials.jenkov.com/maven/maven-tutorial.html
https://maven.apache.org/guides/getting-started/index.html

https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

https://maven.apache.org/guides/getting-started/

Documentation: http://maven.apache.org/guides/index.html

Summary

- What is maven
- Our first maven project
- Maven commands
- Maven in Idea