Java SE & Spring

Module 1: Java SE

17.Maven Build Tool

What is Maven?

In the official Maven website, Maven describes itself as, At first glance Maven can appear to be many things, but in a nutshell Maven is an attempt to apply patterns to a project's build infrastructure in order to promote comprehension and productivity by providing a clear path in the use of best practices. Maven is essentially a project management and comprehension tool and as such provides a way to help with managing:

- Builds
- Documentation
- Reporting
- Dependencies

etc.

Further reading: https://maven.apache.org/guides/getting-started/index.html#What_is_Maven

What is Maven?

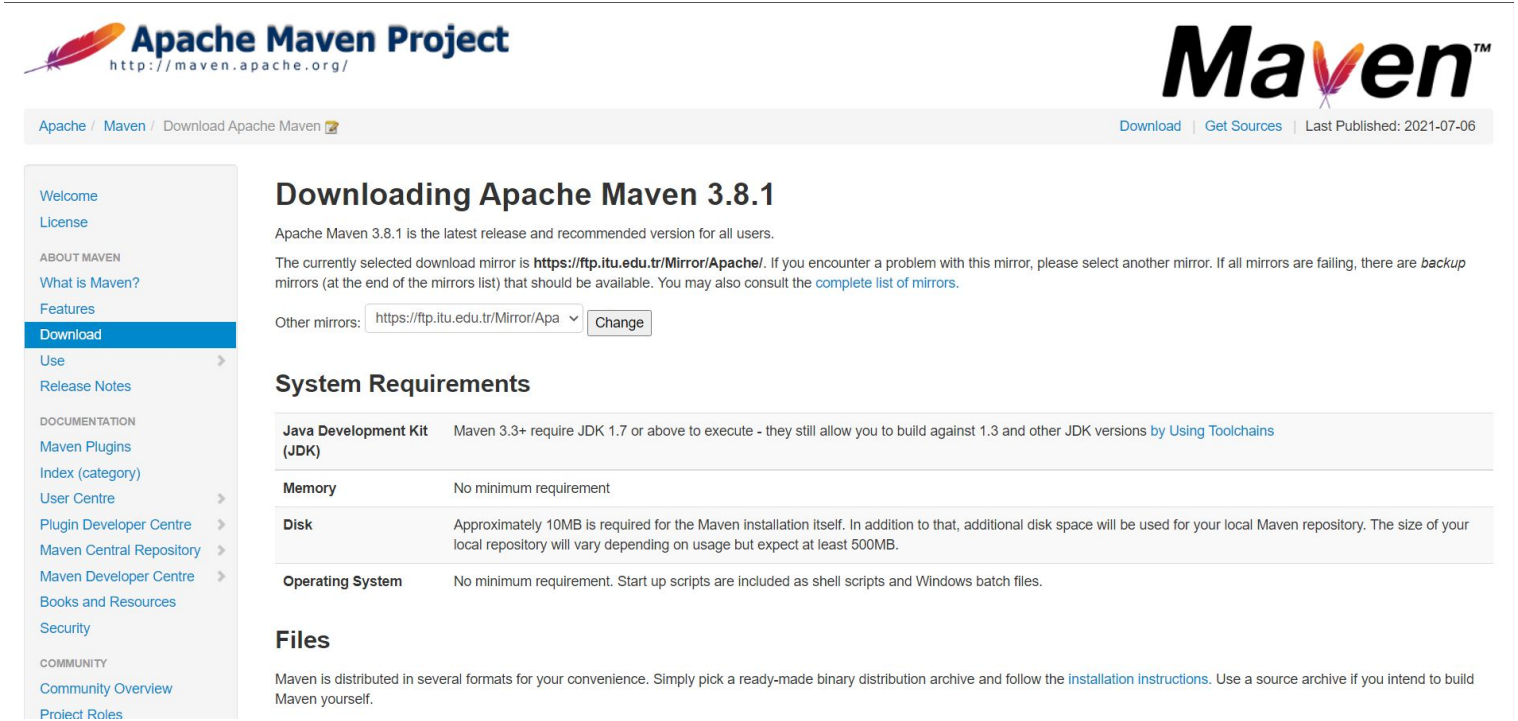
Maven provides a standard project structure. So we can open a Java project in different IDEs. Also provides dependency management, so downloading and loading dependency jars to the classpath is not required with Maven. We managed dependency management and project structure manually so far. But by using Maven, manual configurations decreases significantly. The other benefit of the Maven is, it simplifies version management. We can easily specify the version of our product in pom.xml file. The last benefit of the maven we will talk about is, build and test lifecycle is quite easily handled in Maven. It will automatically carry out unit and integration tests and builds the product, then generates a package.

Starting from this chapter, we will no longer create Java projects as empty or HelloWorld template project.

Further reading: https://maven.apache.org/guides/getting-started/index.html#What_is_Maven

How to Install Maven?

You can download the latest version of the Maven from here.



How to Install Maven?

First as a prerequisite, our computer should has JAVA_HOME environment variable. Then, we need to download the Maven with extension .zip or .tar for Windows, Linux or macOS respectively. After download finished, we will extract the compressed file to a suitable directory. (C:\Program Files\Apache Software Foundation maybe). For other steps, visit the installation guide in official maven website.

Dependency Management

In Maven, dependency is another archive—JAR, ZIP, and so on—which your current project needs in order to compile, build, test, and/or to run. The dependencies are gathered in the **pom.xml** file, inside of a **<dependencies>** tag.

When you run a build or execute a maven goal, these dependencies are resolved, and are then loaded from the local repository. If they are not present there, then Maven will download them from a remote repository and store them in the local repository. You are allowed to manually install the dependencies as well.

POM (Project Object Model) File

POM (Project Object Model) is an XML file that contains information about the project and configuration details used by Maven to build the project i.e. source code location, project dependencies, running unit tests etc. This file must be named as pom.xml and placed under root folder of project. When executing a task or goal, maven reads the POM, gets the needed configuration information, then executes the goal.

Repositories in Maven

Maven repositories are physical directories which contain packaged JAR files along with extra meta data about these jar files. This meta data is in form of POM files which have jar file project information, including what other external dependencies this JAR file has. These other external dependencies are downloaded transitively into your project and become part of effective pom for the project. There are three type of repositories in Maven. These are:

- Local repository
- Central repository
- Remote repository

Local Repository

- Maven local repository reside in the developer's machine. Whenever you run maven goals which require these dependencies, maven will download the dependencies from remote servers and store them into developer's machine.
- By default, Maven create the local repository inside user home directory i.e. C:/Users/your_username/.m2 directory. You can change the location of the local repository in setting.xml file using localRepository tag.
- Having stored the dependencies into local machine has two main benefits. First, multiple projects can access same artifact so it reduces the storage need. Second, as dependency is downloaded only once, it reduces the network usage as well.

Central Repository

- Maven central repository is located at http://repo.maven.apache.org/maven2/. Whenever you run build job, maven first try to find dependency from local repository. If it is not there, then, by default, maven will trigger the download from this central repository location.
- To override this default location, you can can make changes to your settings.xml file to use one or more mirrors.
- You do not need to make any special configuration to allow access the central repository, except network proxy settings if you are behind any firewall.

Remote Repository

- Apart from central repository, you may have needed artifacts deployed on other remote locations. For example, in your corporate office there may be projects or modules specific to organization only. In this cases, organization can create remote repository and deploy these private artifacts. This remote repository will be accessible only inside organization.
- These maven remote repository work exactly same way as maven's central repository. Whenever an artifact is needed from these repositories, it is first downloaded to developer's local repository and then it is used.
- You can configure a remote repository in the POM file or super POM file in remote repository itself.

Questions ?

Next:Introduction to Spring