JAVA ACADEMY - XIDERAL MONTERREY, N.L

WEEK 2 INTRODUCTION TO MAVEN

Made by:

Luis Miguel Sánchez Flores

August 24th, 2024

Introduction

Apache Maven is a project management tool created in 2001, which is used for dependency management, compilation tools, and even as a documentation tool. It is open source and free. Its main function is to streamline the code development processes by providing developers with a coherent structure for their projects, while also taking care of its dependencies it requires to run on..

Maven automates repetitive tasks, allowing programmers to focus on the development of application logic. In addition, they simplify collaboration between different developers due to the standardized project structure, allowing team members to work on the project without having to tweak the configurations and dependencies the software uses.

Although it can be used in several programming languages such as C#, Ruby, Scala, among others, it is mainly used in Java projects, as it was originally designed for the Jakarta Turbine project for creating Java web applications, so its core features are well suited for the Java ecosystem. In fact, Java frameworks such as Spring, and Spring Boot use it by default, while popular Java IDEs such as Eclipse and NetBeans integrate Maven to easily import / export projects.

Maven uses conventions as to where it should place special files for the build process of a project, leaving the developer to only set up the exceptions. Furthermore, Maven is considered to be a declarative build tool, meaning that the developers only describes **what** they want to achieve with the build process, rather than specifying **how** it should do it, which allows for a consistent build process between projects, as well as sharing and reusing build logic. The dependencies, external components, compilation commands, etc., are stored in an XML file.

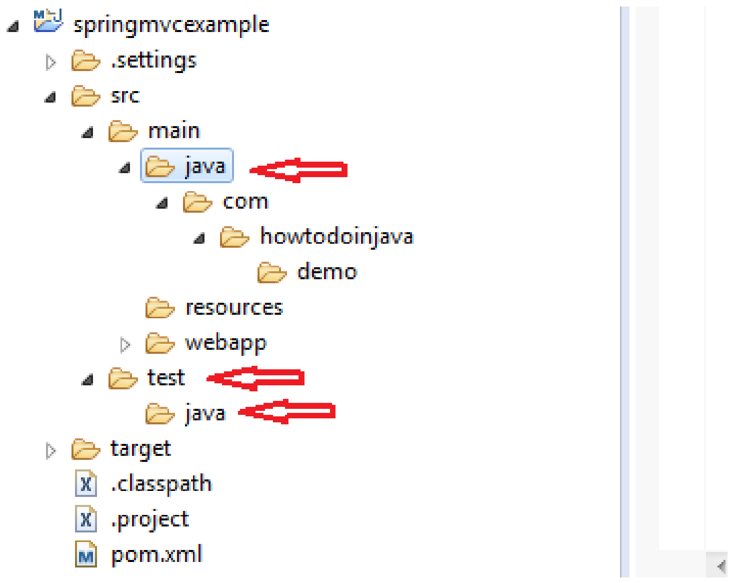
Key characteristics of the Maven tool includes:

- Manage project dependencies, to download and install modules, packages, and tools that are relevant.
- Automatically compile the source code of the application.
- Package the code in .jar or .zip files
- Generate the documentation of the project with ease.
- Manage the different phases of the life cycle of the builds, validations, source code generation, processing, resource generation, complications, test executions, among others.

Key Concepts

Some of the key concepts to take into account when starting out with the Maven tool are as follows:

- **Project structure:** Maven provides a standard project structure that ensures a consistent project organization that allows developers familiar with Maven to understand and navigate the project's files quickly. Maven stores the source code in the src/main/java directory, the resources in the src/main/resources, and the test codes into the src/test/java directory.



- **Build Lifecycle:** Well-defined processes for building and distributing projects that allow developers to manage and automate project builds with ease. Maven offers three built-in build lifecycles: **default, clean** and **site.** Each Build Lifecycle is composed of various **phases,** that are stages that cover a series of **Goals** in which executes specific tasks in a sequential manner in order to achieve a determined end goal.

- **Plugins:** Group of goals that support a particular task, such as compiling code, running tests, packaging the project and deploying artifacts.

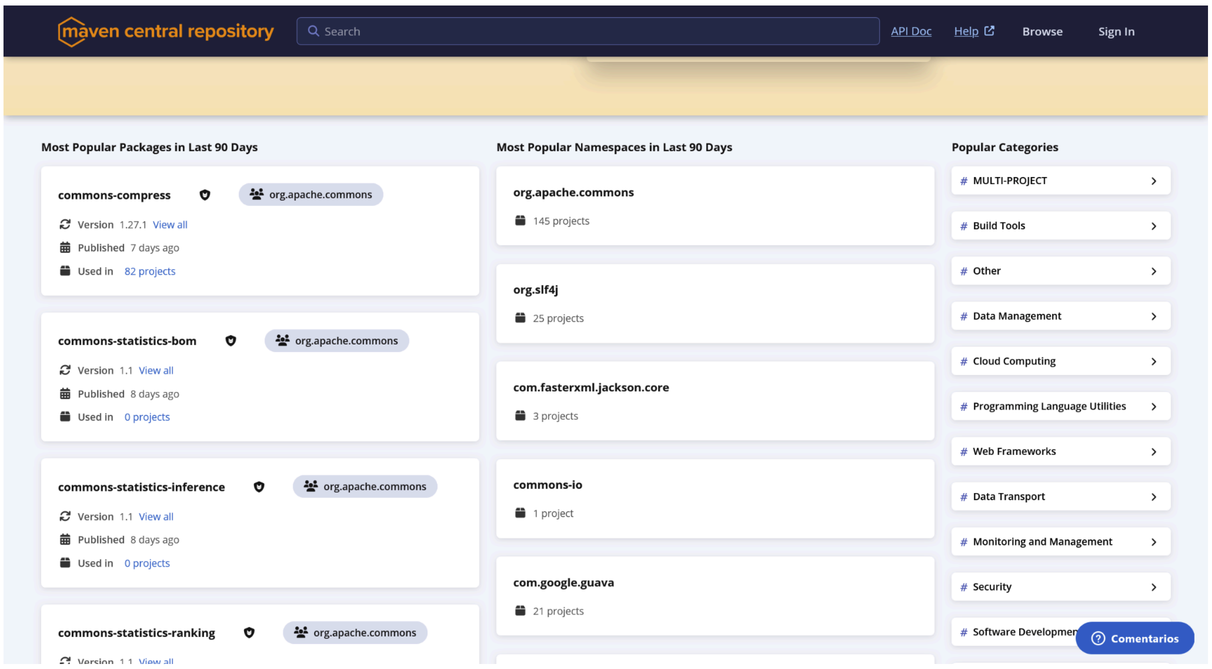
- **Project Object Model :** Also known as POM. Is the basic unit of a Maven project that consists of a XML file containing information about the project itself, its settings, dependencies, etc. The POM contains **coordinates** that act as a unique identifier of the project and is composed of a version number, group ID and a project's ID.

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

  <groupId>com.mycompany.app</groupId>
  <artifactId>my-app</artifactId>
  <version>1</version>
</project>
```

- **Dependencies:** Set of libraries or modules required by the project in order to work. Maven allows the declaration of the project's dependencies in the POM file, after which it will automatically download and include it onto the project.

- **Repository:** Locations where Maven stores and retrieves the dependencies a project relies on. There are two types of repositories: **local** (the developer's machine) or **remote** (server that hosts dependencies such as Maven Central).



maven central repository | Search | API Doc | Help | Browse | Sign In

Most Popular Packages in Last 90 Days

commons-compress — org.apache.commons
Version 1.27.1 View all
Published 7 days ago
Used in 82 projects

commons-statistics-bom — org.apache.commons
Version 1.1 View all
Published 8 days ago
Used in 0 projects

commons-statistics-inference — org.apache.commons
Version 1.1 View all
Published 8 days ago
Used in 0 projects

commons-statistics-ranking — org.apache.commons
Version 1.1 View all

Most Popular Namespaces in Last 90 Days

org.apache.commons
145 projects

org.slf4j
25 projects

com.fasterxml.jackson.core
3 projects

commons-io
1 project

com.google.guava
21 projects

Popular Categories

- MULTI-PROJECT
- Build Tools
- Other
- Data Management
- Cloud Computing
- Programming Language Utilities
- Web Frameworks
- Data Transport
- Monitoring and Management
- Security
- Software Developmen

Comentarios

- **Archetypes:** Project templates that allow developers to quickly start new projects through a predefined project structure and configuration.

List of Commands

Similarly, is worth considering the following commands to effectively work on a Maven project:

1. **mvn clean:** Deletes the target directory containing the compiled classes and other generated files. Useful when there's a need to start fresh before a build or resolve issues from outdated artifacts.

2. **mvn compile:** Compiles the source code of the project, without running any tests or packaging the project.

3. **mvn test:** Run the unit tests of the project, as to ensure that the project works as expected and without any major problem.

4. **mvn package:** Compiles the source files, runs the unit tests and packages the project into the final JAR or WAr file, transforming the project into a deployable artifact.

5. **mvn install:** Builds the maven project and installs the artifact onto the local Maven repository, making it available to use in other local projects.

6. **mvn deploy:** Deploys the maven project into a remote repository, making it available to use to other developers and systems. Requires a repository configuration.

7. **mvn site:** Generates the project's documentation page using the information included in the POM file and the source code's comments.

8. **mvn dependency:tree :** Display the project's dependencies and their versions in a tree-like structure, along with the relationships between dependencies.