Maven Tutorial Notes

What is Maven?

Maven is a build automation tool primarily used for Java projects. It addresses two main aspects of building software: dependency management and project build lifecycle management. It simplifies the build process by managing dependencies, compiling source code, packaging deliverables, and deploying them to repositories. Maven is based on the Project Object Model (POM).

History and Evolution of Maven

Developed by the Apache Software Foundation and released in 2004, Maven was created to address the limitations of Apache Ant. It introduced a uniform build system, simplified dependency management, and improved project information sharing. Today, it's one of the most widely used build tools in the Java ecosystem.

Setting Up Maven

To install Maven, ensure Java (JDK) is installed and JAVA_HOME is configured. Then download and extract Maven, set environment variables (MAVEN_HOME and PATH), and verify installation via `mvn -version`. IDEs like IntelliJ IDEA have built-in Maven support, while Eclipse requires the m2e plugin.

Understanding .m2 Directory and settings.xml

The .m2 directory stores the local repository where Maven caches downloaded dependencies. The settings.xml file customizes Maven's behavior globally, allowing configurations for mirrors, proxies, and local repository paths.

Project Object Model (POM)

The pom.xml is the central configuration file in Maven projects, defining project metadata, dependencies, plugins, and build instructions. Key elements include groupId, artifactId, version, packaging, dependencies, and build configurations.

Maven Dependency Management

Maven simplifies dependency management by automatically handling both direct and transitive dependencies. It supports multiple scopes (compile, provided, runtime, test, system) to control availability and inclusion of dependencies during various build phases.

Maven Profiles

Profiles in Maven allow customization of builds for different environments (development, testing, production). Profiles can be activated manually or automatically based on OS, JDK version, or environment variables.

Maven Build Lifecycle

Maven defines three primary lifecycles: Default (handles project builds and deployment), Clean (removes previous build artifacts), and Site (generates project documentation). Common commands include `mvn compile`, `mvn test`, `mvn package`, `mvn install`, and `mvn deploy`.

Maven Repositories

Maven repositories store and retrieve project artifacts. Types include Local (~/.m2/repository), Central (public repository), and Remote (custom/private). Maven checks local, then remote, then central repositories when resolving dependencies.

Best Practices for Dependency Management

Keep dependencies minimal, centralize versions using dependencyManagement, exclude unnecessary transitive dependencies, resolve version conflicts explicitly, and keep libraries updated for security and performance.

Credits

This tutorial note was sourced and adapted from Learn Code With Durgesh.

Original Source: https://learncodewithdurgesh.com/blogs/maven-tutorial