

DAY-2 | MAVEN

Maven is a powerful build automation and dependency management tool used primarily for Java projects, although it can be used for projects in other programming languages as well. It provides a comprehensive set of features to help manage the software development lifecycle and simplify the build process.

Real World Scenario

Let's consider the following scenario:

Company XYZ is developing a web application called "MyWebApp" using Java and the Spring framework. The application consists of multiple modules and relies on external libraries and dependencies. The development team wants to use Maven to manage the build process and handle dependencies effectively.

Here's how Maven can be used in this scenario:

- **Project Setup:** The development team creates a new Maven project for "MyWebApp" using Maven's project archetype. They define the project structure and configuration files such as pom.xml, which is the heart of the Maven project.
- **Dependency Management:** Maven allows the team to specify project dependencies in the pom.xml file. They can easily add dependencies on external libraries, frameworks, and other projects. Maven automatically resolves and downloads the required dependencies from remote repositories, simplifying the process.
- **Build Automation:** The team defines build configurations and plugins in the pom.xml file. They can specify tasks such as compiling source code, running tests, generating documentation, and packaging the application into an executable format (e.g., JAR or WAR file).

- **Continuous Integration:** To ensure a smooth development workflow, the team sets up a continuous integration (CI) system like Jenkins or Travis CI. The CI system is configured to monitor the version control repository (e.g., Git) for changes. Whenever a new commit is made, the CI system automatically triggers a build process using Maven.
- **Testing and Quality Assurance:** Maven integrates well with testing frameworks like JUnit. The team writes unit tests for their code and configures Maven to execute these tests as part of the build process. Additionally, they can configure Maven plugins to perform static code analysis, code coverage analysis, and other quality checks.
- **Deployment and Release:** Once the development team is satisfied with the application's quality, they can use Maven to create a deployable package (e.g., a WAR file) with all the necessary dependencies included. Maven can also automate the deployment process to application servers or cloud platforms like Tomcat, JBoss, or AWS.
- **Dependency Updates:** As the project evolves, new versions of libraries and dependencies are released. Maven simplifies the task of updating dependencies by providing commands to check for updates and resolve any compatibility issues automatically.

Overall, Maven streamlines the build and deployment process, ensuring consistency and reproducibility across different environments. It saves developers' time by automating repetitive tasks and simplifying dependency management.

Maven Build Lifecycle:

Maven has three built-in build lifecycles: clean, default, and site.

- Clean Lifecycle:

clean: Deletes any build output generated by previous builds.

- Default Lifecycle:

validate: Validates the project structure and verifies if all necessary information is available.

compile: Compiles the project's source code.

test: Runs unit tests against compiled source code.

package: Packages the compiled code into a distributable format (e.g., JAR, WAR).

install: Installs the package into the local repository for use as a dependency in other projects. .m2

deploy: Deploys the package to a remote repository for sharing with other developers or environments. Push to Nexus

- Site Lifecycle:

site: Generates project documentation and reports.

site-deploy: Deploys the generated documentation to a remote web server.

Maven Command Examples:

- mvn compile: Compiles the project's source code.
- mvn test: Runs unit tests against compiled code.
- mvn package: Packages the compiled code into a distributable format.
- mvn install: Installs the package into the local repository.
- mvn deploy: Deploys the package to a remote repository.
- mvn site: Generates project documentation and reports.
- mvn site-deploy: Deploys the generated documentation to a remote web server.
- mvn clean: Executes the clean phase, deleting any previous build outputs.