MAVEN

SURESH KUMAR U

What is Maven? A Powerful Build Tool for Java Projects

---

1. Introduction

- **What is a Build Process?**
  - Developers write source code, compile it, test it, and then package it into a distributable format (e.g., JAR, WAR, EXE).
- **Why Use Build Tools?**
  - Automates repetitive tasks like compiling, testing, and packaging, saving time and effort.

---

2. What is Maven?

- A build tool for Java projects that automates the entire build process, from compiling the code to deploying it.
- **Maven** automates the process of compiling, testing, packaging, and deploying Java applications.
- Uses an XML configuration file called **pom.xml** to define dependencies and configurations.

3.Why Use Build Tools:

- The build process involves multiple steps and can be tedious to do manually. Build tools like Maven automate this process, making it easier and more efficient for developers.

---

4. Common Build Tools

- **Maven** (Java, XML-based)
- **Ant** (Java, script-based)
- **Gradle** (Groovy-based)
- **MSBuild** (Microsoft)
- **Make** (general-purpose)

---

5. Maven Build Phases

Maven has several phases that are executed in order:

- **Validate**: Checks if the project structure is correct.
- **Compile**: Compiles the source code.
- **Test**: Runs unit tests.
- **Package**: Packages the compiled code into a distributable format (JAR, WAR, etc.).
- **Verify**: Checks the quality of the tests.
- **Install**: Downloads and stores dependencies locally.
- **Deploy**: push the packaged(compiled) application to a remote repository.

---

6. Understanding pom.xml

- **pom.xml** is the Project Object Model file where you define:
    - Dependencies: Libraries your project needs.
    - Plugins: Tools used during the build process.
    - Build Configurations: How the build should behave.

---

7. Key Elements of POM.xml

- **<modelVersion>**: Specifies the version of the POM model. The current version is 4.0.0.

- **<groupId>**: This defines the group or organization that is responsible for the project. It usually follows a reverse-domain-name pattern (e.g., com.example).

- **<artifactId>**: The unique name of the project or artifact. This is the name that will be used to identify your project's built files (e.g., JAR or WAR).

- **<version>**: Specifies the version of your project. The versioning helps in identifying different releases of the same project. For example, 1.0-SNAPSHOT represents the development version.

---

8. Maven Commands

- **Common Commands**:
  - **mvn validate** - Validate the project structure.
  - **mvn compile** - Compile the code.
  - **mvn test** - Run unit tests.
  - **mvn package** - Package the project into JAR/WAR.
  - **mvn install** - Install the package to the local repository.
  - **mvn deploy** - Deploy the package to a remote repository.

---

9. Best Practices with Maven

- **Project Structure**: Follow the standard Maven directory layout.
- **Dependency Management**: Use Maven's central repository for dependencies and handle version conflicts carefully.

---

10. Conclusion

- **Maven** automates the build process, saving developers time and ensuring consistency across builds.
- It simplifies tasks like compiling code, running tests, and deploying artifacts.
