Maven™

SPRING BOOT

thaparohan2019@gmail.com

Rohan Thapa

What is Maven?

Maven is a powerful **build** automation tool primarily used for **Java projects**.

It simplifies the process of **building**, **managing dependencies**, and **documenting** your project.

Maven uses a **Project Object Model (POM)** file to describe the project's **configuration**, **dependencies**, **build process**, and **more**.

Why Maven?

- **Dependency Management:** Automatically handles project dependencies, ensuring your project has all the required libraries.

- **Build Automation:** Simplifies the **build process**, making it easy to **compile**, **test**, **package**, and **deploy** your application.

- **Project Structure:** Enforces a standard project structure, making it easier for developers to understand and contribute to the project.

- **Integration with IDEs:** Seamlessly integrates with popular IDEs like **IntelliJ IDEA, Eclipse, and NetBeans**.

Maven Structure and Components

Maven's functionality revolves around several key components:

1. **POM (Project Object Model) File:**
    - The **pom.xml** is the **heart** of a Maven project. It contains details like project **information, dependencies, plugins, build profiles**, etc.
    - Key sections include:
        - **GroupId:** Identifies the project uniquely across all projects (typically your organization).
        - **ArtifactId:** The name of the project.
        - **Version:** The project's version.
        - **Dependencies:** Specifies the external libraries required by the project.

Maven Structure and Components

2.Maven Lifecycle:

- Maven defines a standard build lifecycle consisting of phases like:
  - **Validate:** Validate the project is correct and all necessary information is available.
  - **Compile:** Compile the source code.
  - **Test:** Run unit tests.
  - **Package:** Package the compiled code into a distributable format (e.g., JAR, WAR).
  - **Install:** Install the package into the local repository.
  - **Deploy:** Deploy the package to a remote repository for sharing with other developers.

Maven Lifecycle Demo

```
[INFO] Scanning for projects...
[INFO]
[INFO] ---------------------------< com.rohan:demo >---------------------------
[INFO] Building demo 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ demo ---
[INFO] Copying 1 resource from src\main\resources to target\classes
[INFO] Copying 0 resource from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.13.0:compile (default-compile) @ demo ---
[INFO] Recompiling the module because of changed source code.
[INFO] Compiling 7 source files with javac [debug parameters release 22] to target\classes
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ demo ---
[INFO] skip non existing resourceDirectory C:\Users\RohanThapa\Desktop\learnjava\demo\src\test\resources
[INFO]
[INFO] --- compiler:3.13.0:testCompile (default-testCompile) @ demo ---
[INFO] Recompiling the module because of changed dependency.
[INFO] Compiling 1 source file with javac [debug parameters release 22] to target\test-classes
[INFO]
[INFO] --- surefire:3.2.5:test (default-test) @ demo ---
[INFO] Tests are skipped.
[INFO]
[INFO] --- spring-boot:3.3.2:process-aot (process-aot) @ demo ---

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.2)

2024-08-25T19:31:21.908+05:45  INFO 6888 --- [demo] [           main] com.rohan.demo.DemoApplication           : Starting DemoApplication using
 Java 22.0.1 with PID 6888 (C:\Users\RohanThapa\Desktop\learnjava\demo\target\classes started by RohanThapa in
 C:\Users\RohanThapa\Desktop\learnjava\demo)
2024-08-25T19:31:21.916+05:45  INFO 6888 --- [demo] [           main] com.rohan.demo.DemoApplication           : No active profile set, falling
 back to 1 default profile: "default"
[INFO]
[INFO] --- jar:3.4.2:jar (default-jar) @ demo ---
[INFO] Building jar: C:\Users\RohanThapa\Desktop\learnjava\demo\target\demo-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- spring-boot:3.3.2:repackage (repackage) @ demo ---
[INFO] Replacing main artifact C:\Users\RohanThapa\Desktop\learnjava\demo\target\demo-0.0.1-SNAPSHOT.jar with repackaged archive, adding nested
 dependencies in BOOT-INF/.
[INFO] The original artifact has been renamed to C:\Users\RohanThapa\Desktop\learnjava\demo\target\demo-0.0.1-SNAPSHOT.jar.original
[INFO]
[INFO] --- install:3.1.2:install (default-install) @ demo ---
[INFO] Installing C:\Users\RohanThapa\Desktop\learnjava\demo\pom.xml to C:\Users\RohanThapa\.m2\repository\com\rohan\demo\0.0.1-SNAPSHOT\demo-0.0
 .1-SNAPSHOT.pom
[INFO] Installing C:\Users\RohanThapa\Desktop\learnjava\demo\target\demo-0.0.1-SNAPSHOT.jar to C:\Users\RohanThapa\
 .m2\repository\com\rohan\demo\0.0.1-SNAPSHOT\demo-0.0.1-SNAPSHOT.jar
[INFO]
[INFO] --- deploy:3.1.2:deploy (default-deploy) @ demo ---
[INFO] ------------------------------------------------------------------------
```

validate

compile

test

packaging

install

deploy

Maven Structure and Components

3.Maven Plugins:

- Maven plugins execute during the build lifecycle. Common plugins include:
  - **maven-compiler-plugin:** Compiles Java code.
  - **maven-surefire-plugin:** Runs unit tests.
  - **maven-jar-plugin:** Packages the project as a JAR file.

Standard Maven Directory Structure

Maven enforces a standard project structure, which includes:

```
my-app/
├── src/
│   ├── main/
│   │   ├── java/              # Application/Library sources
│   │   ├── resources/         # Configuration files and other resources
│   │   └── webapp/            # Web application resources (for WAR packaging)
│   │
│   └── test/
│       ├── java/              # Test sources
│       └── resources/         # Test resources
│
├── pom.xml                    # Project Object Model (POM) file
└── target/                    # Output directory for built artifacts
```

POM File Structure

The **pom.xml** file is divided into several key sections:

- **Model Version:** Specifies the POM model version.
- **GroupId, ArtifactId, Version:** Unique identifiers for the project.
- **Properties:** Defines custom properties like Java version.
- **Dependencies:** Lists the libraries required by the project.
- **Build:** Contains plugins that define the build process.
- **Repositories:** Specifies locations from which Maven should download dependencies.
- **Profiles:** Allows different build configurations for different environments (e.g., development, production).

pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent...>
    <groupId>com.rohan</groupId>
    <artifactId>demo</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>demo</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>22</java.version>
    </properties>
    <dependencies>
        <dependency...>
        <dependency...>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

Dependencies Management

Maven manages project dependencies using a hierarchical approach:

- **Direct Dependencies:** Defined in the dependencies section of the POM file.

- **Transitive Dependencies:** Dependencies of your dependencies are automatically included.

- **Dependency Scopes:** Define the visibility and availability of dependencies, such as compile, test, provided, and runtime.

How Maven Works

1. **Reading POM:** Maven starts by reading the pom.xml.
2. **Dependency Resolution:** Maven resolves the dependencies specified in the POM file by downloading them from the specified repositories.
3. **Build Execution:** Maven executes the build phases in order, executing the corresponding plugins.
4. **Lifecycle Phases:** Maven follows the lifecycle phases to **validate, compile, test, package, and deploy the project.**

Benefits of Maven

- **Consistency:** Maven standardizes the build process across multiple projects.

- **Dependency Management:** Automatically handles dependency resolution and versioning.

- **Integration:** Easily integrates with CI/CD tools and IDEs.

- **Extensibility:** Maven plugins extend its functionality to cover a wide range of tasks.

Disadvantage of Maven

- **Learning Curve:** Maven's complexity can be daunting for beginners.

- **Configuration Overhead:** Managing large POM files can become cumbersome.

- **Performance:** Maven's reliance on XML and its dependency resolution process can slow down builds, especially for large projects.

Conclusion

Maven is a robust tool that simplifies **Java project management** by providing a **standardized approach** to building, managing dependencies, and more.

Its integration with **IDEs** and **CI/CD pipelines** makes it an essential tool in modern Java development.

By understanding **Maven's structure, lifecycle, and configuration**, you can efficiently manage your projects and streamline your development workflow.

Thank You

thaparohan2019@gmail.com

Rohan Thapa