Understanding Maven with Spring Boot

youeleven

.

Follow

2 min read

.

May 6, 2024

--

Listen

Share

Maven is a powerful build automation tool primarily used for Java projects. When combined with Spring Boot, it streamlines the development process by managing dependencies, building projects, and handling configurations efficiently. In this article, we'll delve into Maven's role in Spring Boot development and provide a hands-on example to illustrate its usage.

Maven Fundamentals

Dependency Management

One of Maven's key features is its robust dependency management system. Developers can specify project dependencies in the `pom.xml` file, and Maven resolves these dependencies from repositories such as Maven Central or custom repositories.

```
<dependencies>
   <dependency>
       <groupId>org.springframework.boot</groupId>
       <artifactId>spring-boot-starter-web</artifactId>
       <version>2.6.3</version>
   </dependency>
   <!-- Other dependencies -->
</dependencies>
```

In the above snippet, we're declaring a dependency on `spring-boot-starter-web`, which includes essential libraries for building web applications with Spring Boot.

What happens when you run your Spring Boot application?

`mvn compile`:

- Maven executes the `compile` phase, which compiles the source code in the `src/main/java` directory.
- It checks for syntax errors and generates the corresponding bytecode (.class files) in the `target/classes` directory.
- Dependencies specified in the `pom.xml` are downloaded if not already present in the local Maven repository (`~/.m2/repository`).

`mvn test`:

- Maven executes the `test` phase, which runs unit tests located in the `src/test/java` directory.
- It uses a test framework like JUnit or TestNG to execute the tests and generates test reports.

`mvn package`:

- Maven executes the `package` phase, which packages the compiled source code, resources, and dependencies into a distributable format (e.g., JAR or WAR).
- The packaged artifact is created in the `target` directory (`target/application-name.jar` for a JAR-based project).

`mvn install`:

- Maven executes the `install` phase, which installs the packaged artifact (`application-name.jar`) into the local Maven repository (`~/.m2/repository`).
- The installed artifact can now be used as a dependency in other Maven projects on the same machine.

`mvn spring-boot:run`:

- When you run `mvn spring-boot:run`, Maven invokes the Spring Boot Maven Plugin's `run` goal.
- This goal triggers the execution of the `compile`, `test`, and `package` phases automatically.
- After successful compilation, testing, and packaging, the Spring Boot application is started using an embedded Tomcat or Jetty server, depending on your configuration.

So, when you run your Spring Boot application using `mvn spring-boot:run`, Maven handles the compilation, testing, packaging, and execution stages seamlessly, leveraging the Spring Boot Maven Plugin for application execution.