|  |
| --- |
|  |
|  | **Command used in maven?**  **Maven Commands:** |
|  | mvn clean: Cleans the project and removes all files generated by the previous build. |
|  | mvn compile: Compiles source code of the project. |
|  | mvn test-compile: Compiles the test source code. |
|  | mvn test: Runs tests for the project. |
|  | mvn package: Creates JAR or WAR file for the project to convert it into a distributable format. |
|  | mvn install: Deploys the packaged JAR/ WAR file to the local repository. |
|  | mvn deploy: Copies the packaged JAR/ WAR file to the remote repository after compiling, running tests and building the project. |
|  | Maven Commands |
|  | mvn clean: Cleans the project and removes all files generated by the previous build. |
|  | mvn compile: |
|  | mvn test-compile: |
|  | mvn --version Prints out the version of Maven you are running. |
|  | mvn clean |
|  | mvn package -Dmaven.test.skip=true |
|  | mvn clean package |
|  | mvn clean package -Dmaven.test.skip=true |
|  | mvn verify |
|  | mvn clean verify |
|  | mvn install |
|  | mvn install -Dmaven.test.skip=true |
|  | mvn clean install |
|  | mvn clean install -Dmaven.test.skip=true |
|  | mvn dependency:copy-dependencies |
|  | mvn clean dependency:copy-dependencies |
|  | mvn dependency:tree |
|  | mvn dependency:tree -Dverbose |
|  | mvn dependency:tree -Dincludes=com.fasterxml.jackson.core |
|  | mvn dependency:tree -Dverbose -Dincludes=com.fasterxml.jackson.core |
|  | mvn dependency:build-classpath |
|  | $ mvn clean |
|  | $ mvn compiler:compile |
|  | $ mvn compiler:testCompile |
|  | $ mvn package |
|  | $ mvn install |
|  |  |
|  |  |
|  | **Maven Life cycle?** |
|  | When we build a Maven project, it executes a set of clearly defined tasks based on the project pom.xml configuration and the command-line options. |
|  | This standard set of tasks creates the maven build lifecycle. |
|  | The benefit of a clearly defined lifestyle is that we have to remember only a few sets of commands to compile, build, install, and deploy our projects. |
|  | 1)There are three built-in build lifecycles. |
|  | default: handles project build and deployment |
|  | clean: handles project cleaning |
|  | site: handles the creation of project site documentation. |
|  | 2)Maven build lifecycle goes through a set of stages, they are called build phases. |
|  | For example, the default lifecycle is made up of the following phases. |
|  | validate, compile, test, package, verify, install, deploy |
|  | The build phases are executed sequentially. When we run a maven build command, we specify the phase to be executed. |
|  | Any maven build phases that come before the specified phase is also executed. |
|  | For example, if we run mvn package then it will execute validate, compile, test, and package phases of the project. |
|  | 3)A build phase is made up of a set of goals. Maven goals represent a specific task that contributes to the building and managing of a project. |
|  | Sometimes, a maven goal is not bound to a build phase. We can execute these goals through the command line. |
|  | The syntax to execute a goal is:$ mvn plugin-prefix:goal |
|  | $ mvn plugin-group-id:plugin-artifact-id[:plugin-version]:goal |