What is Maven?

Maven is a build automation and project management tool primarily used in Java projects. It simplifies the process of building, managing dependencies, and deploying projects. Maven uses an XML file called `pom.xml` (Project Object Model) to manage a project's build configuration, dependencies, and plugins.

Why is Maven Popular?

1. **Dependency Management**: Maven automatically downloads project dependencies from a central repository, simplifying the process of managing libraries.

2. **Standardized Project Structure**: It enforces a common project structure, making it easier for developers to understand and navigate.

3. **Plugins and Extensibility**: Maven has a rich ecosystem of plugins that can be used to extend its capabilities, from compiling code to packaging, testing, and deploying.

4. **Consistency Across Projects**: By standardizing the build process, Maven ensures consistency across multiple projects.

5. **Integration with CI/CD Tools**: Maven integrates seamlessly with Continuous Integration and Continuous Deployment (CI/CD) tools like Jenkins, making it easier to automate builds and deployments.

How Maven Works

Maven works by reading the `pom.xml` file, which contains information about the project, dependencies, and build instructions. When you run Maven commands, it performs various tasks like downloading dependencies, compiling code, running tests, and packaging the project. Maven downloads dependencies from remote repositories and caches them locally to avoid repeated downloads.

How to Create a Maven Project

1. Using Command Line:

- Open the terminal and navigate to the directory where you want to create the project.

- Run the following command:

```
mvn archetype:generate -DgroupId=com.example -DartifactId=my-app -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

- This command generates a simple Java project with the specified `groupId` and `artifactId`.

2. Using an IDE (like IntelliJ IDEA or Eclipse):

- Go to `File` -> `New` -> `Project`.

- Choose `Maven` as the project type.

- Fill in the `groupId`, `artifactId`, and other required fields.

- Finish the setup to create the project.

How to Create a JAR File and Run on JDK of Another Machine

1. Package the Project:

- In the terminal, navigate to the project directory and run:

```
mvn clean package
```

- This command will compile the code, run tests, and package the compiled code into a JAR file inside the `target` directory.

2. Copy the JAR File to Another Machine:

- Copy the JAR file from the `target` directory to the other machine.

3. Run the JAR File on Another Machine:

- Ensure the other machine has the correct JDK version installed.
- Open the terminal on the other machine and run the JAR file using the following command:

```
java -jar my-app.jar
```

10 Popular Maven Commands

1. mvn clean:

- Cleans the project by removing the `target` directory where compiled code and packaged files are stored.

2. mvn compile:

- Compiles the source code of the project.

3. mvn test:

- Runs the tests using the testing framework specified in the project.

4. mvn package:

- Compiles the code and packages it into a JAR/WAR file.

5. mvn install:

- Installs the packaged JAR/WAR file into the local repository for use as a dependency in other projects.

6. mvn deploy:

- Deploys the packaged artifact to a remote repository for sharing with other developers.

7. mvn dependency:resolve:

- Resolves and displays all the dependencies required for the project.

8. mvn site:

- Generates a site for the project, including reports on test coverage, Javadoc, etc.

9. mvn dependency:tree:

- Displays a tree representation of the project dependencies.

10. mvn exec:java -Dexec.mainClass="com.example.Main":

- Runs the main method of the specified class within the project without packaging it into a JAR file.