Lab Sheet - Maven

Introduction to Maven

Maven is a powerful build tool used for managing the entire lifecycle of a software project. It automates tasks like compiling source code, managing dependencies, running tests, packaging artifacts, and more.

Installing Maven

Windows

1. Download Maven:
    a. Open your web browser and go to the official Maven website: https://maven.apache.org/download.cgi
    b. Scroll down to the "Files" section and look for the latest version of Maven (e.g., "Apache Maven 3.8.4"). Click on the link to download the ZIP file.
2. Choose Installation Directory:
    a. Create a directory on your system where you want to install Maven. For example, you can create a directory named C:\Program Files\Apache\.
    b. Extract the contents of the downloaded ZIP file into this directory. After extraction, you should have a folder like C:\Program Files\Apache\apache-maven-3.8.4 (version number might differ).
3. Set Environment Variables:
    a. Right-click on the "This PC" (or "My Computer") icon on your desktop or in the File Explorer.
    b. Choose "Properties" from the context menu.
    c. In the System Properties window, click on the "Advanced system settings" link on the left side.
4. Environment Variables:
    a. In the System Properties window, click on the "Environment Variables..." button.
    b. Under the "System variables" section, click "New" to create a new environment variable.
5. Create MAVEN_HOME Variable:
    a. Variable name: MAVEN_HOME
    b. Variable value: Path to your Maven installation directory (e.g., C:\Program Files\Apache\apache-maven-3.8.4).
6. Update PATH Variable:
    a. Find the "Path" variable under "System variables" and click "Edit."
    b. Click "New" and add %MAVEN_HOME%\bin to the list of paths.
7. Verify Installation:

a. Open a new Command Prompt (CMD) or PowerShell window.
b. Run the following commands to verify Maven installation:

```
mvn –version
```

Mac OS

```
brew install maven
```

Linux

```
sudo apt install maven
```

Create Project with maven

Let’s first create a project with maven and maven will do the following for you,

- Create a project with standard structure
- Create a sample test path
- Create pom.xml file for you (Project Object Model)

Run the following command

```
mvn archetype:generate -DgroupId=com.mycompany.app
-DartifactId=hello-world
-DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4
```

- mvn: This is the command-line tool for Maven. It's used to execute Maven commands from the terminal.
- archetype:generate: This part of the command tells Maven to generate a new project based on an archetype. An archetype is a template or blueprint for creating a specific type of project.
- -DgroupId=com.mycompany.app: This parameter specifies the Group ID of the project. The Group ID is a unique identifier for your project's group or organization.
- -DartifactId=hello-world: This parameter specifies the Artifact ID of the project. The Artifact ID is a unique identifier for your project within the Group ID.
- -DarchetypeArtifactId=maven-archetype-quickstart: This parameter specifies the Artifact ID of the archetype to use. In this case, it's maven-archetype-quickstart, which is a commonly used archetype for creating a simple Java project.

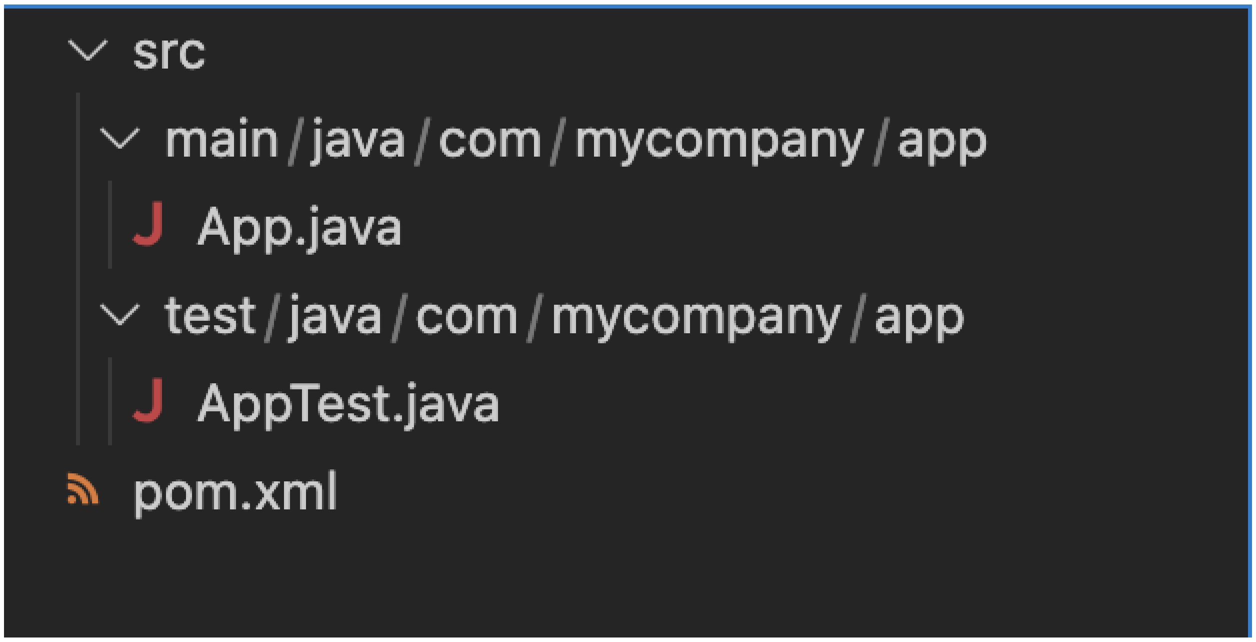
- -DarchetypeVersion=1.4: This parameter specifies the version of the archetype to use. In this case, it's version 1.4 of the maven-archetype-quickstart.

Putting it all together, when you run this command, Maven will use the maven-archetype-quickstart archetype to generate a new project with the Group ID com.mycompany.app and the Artifact ID hello-world. The generated project will have a basic directory structure, a sample Java class, and a pom.xml file for managing the project's build configuration and dependencies.

After generating the project, you can navigate into the project's directory and start working on your Java application using Maven.

Project Structure

Following is the project structure you will get after running the above command. It will create a simple hello world program.



```
src
  main / java / com / mycompany / app
    App.java
  test / java / com / mycompany / app
    AppTest.java
pom.xml
```

- src/main - Contain all the source code
- src/test - Contains all the codes for testing
- Pom.xml - Contains all the data related to the project

POM Xml

The pom.xml (Project Object Model) is an XML file used in Maven projects to define project configuration, build settings, dependencies, plugins, and other project-related information. It serves as the heart of your project's build and management process. Here's a general guide to understanding and configuring the pom.xml file:

Basic Structure

```
<project>
   <modelVersion>4.0.0</modelVersion>

   <!-- Project coordinates -->
   <groupId>com.example</groupId>
   <artifactId>my-project</artifactId>
   <version>1.0.0</version>

   <!-- Project properties -->
   <properties>
      <java.version>1.8</java.version>
      <!-- Other properties -->
   </properties>

   <!-- Dependencies, build configuration, plugins, etc. -->
</project>
```

Key Elements

- <modelVersion>: Specifies the version of the POM format being used.
- <groupId>: Specifies a unique identifier for your project's group or organization.
- <artifactId>: Specifies the unique identifier for your project within the group.
- <version>: Specifies the version of your project.
- <properties>: Allows you to define project properties that can be reused throughout the POM.

Dependencies

```
<dependencies>
   <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-core</artifactId>
      <version>5.3.8</version>
   </dependency>
   <!-- Other dependencies -->
</dependencies>
```

- <dependencies>: Defines the project's dependencies. Dependencies are typically external libraries required for your project.

Build Configurations

```
<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.8.1</version>
         <configuration>
            <source>1.8</source>
            <target>1.8</target>
         </configuration>
      </plugin>
      <!-- Other plugins -->
   </plugins>
</build>
```

- <build>: Contains build-related configuration.
- <plugins>: Lists the plugins used in the build process.
- <configuration>: Customizes plugin behavior. For example, the maven-compiler-plugin sets Java version.

Maven Commands

1. mvn clean: Deletes the target directory and any files generated during the build process.
    - Example: mvn clean
2. mvn compile: Compiles the source code of the project.
    - Example: mvn compile
3. mvn test: Runs tests for the project.
    - Example: mvn test
4. mvn package: Packages the compiled code into a distributable format (e.g., JAR, WAR).
    - Example: mvn package
5. mvn install: Installs the packaged artifact into the local Maven repository.
    - Example: mvn install
6. mvn clean install: Combines the clean and install phases, ensuring a clean build and then installing the artifact.
    - Example: mvn clean install
7. mvn clean package: Combines the clean and package phases, ensuring a clean build and then packaging the artifact.
    - Example: mvn clean package

Run Compiled JAR

```
java -cp target/hello-world-1.0-SNAPSHOT.jar com.mycompany.app.App
```

Read Properties From pom.xml

Please add the following section to the pom xml and let’s learn how to read them from pom xml

```
<properties>
    <app.name>HelloWorldApp</app.name>
    <app.version>1.0.0</app.version>
</properties>
```

Following is the updated code

```
public class App {
    public static void main(String[] args) {
        String appName = System.getProperty("app.name");
        String appVersion = System.getProperty("app.version");

        System.out.println("App Name: " + appName);
        System.out.println("App Version: " + appVersion);
    }
}
```

Run Jar File

We need to provide the main class information when building to run the JAR file without any additional information.

Please add the following to the pom.xml file.

```
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-jar-plugin</artifactId>
            <version>3.2.0</version>
            <configuration>
                <archive>
                    <manifest>
                        <mainClass>com.mycompany.app.App</mainClass>
                    </manifest>
                </archive>
            </configuration>
        </plugin>
```

Then Build the project and run the following command

```
Java -jar <JAR_FILE>
```