**Maven install and configure step by step on ubuntu**

**Step 1: prerequisites - java**

$ sudo apt-get update

$ apt install default-jdk

$ apt install openjdk-11-jdk-headless

$ apt install ecj

$ apt install openjdk-8-jdk-headless

**Step 2: download and install Maven**

$ cd /opt/

$ sudo wget http://apachemirror.wuchna.com/maven/maven-3/3.6.2/binaries/apache-maven-3.6.2-bin.tar.gz

$ sudo tar -xvzf apache-maven-3.6.2-bin.tar.gz

$ sudo mv apache-maven-3.6.2/ apache-maven/

$ sudo update-alternatives --install /usr/bin/mvn maven /opt/apache-maven/bin/mvn 1001

**Step 3: verify if maven is installed**

$ mvn -v

or

$ mvn –version

**Download and unzip the source repository for this guide**

git clone https://github.com/spring-guides/gs-maven.git

### Create the directory structure

$ cd gs-maven

$ tree

$ cd gs-maven/initial

$ tree

$ cat src/main/java/hello/HelloWorld.java

$ cat src/main/java/hello/Greeter.java

$ mvn -v

## **Define a simple Maven build**

$ cat pom.xml

## **Build Java code**

$ mvn compile

$ mvn package

**To execute the JAR file run:**

$ java -jar target/gs-maven-0.1.0.jar

$ mvn install

## **Declare Dependencies**

$ cat src/main/java/hello/HelloWorld.java

package hello;

import org.joda.time.LocalTime;

public class HelloWorld {

public static void main(String[] args) {

LocalTime currentTime = new LocalTime();

System.out.println("The current local time is: " + currentTime);

Greeter greeter = new Greeter();

System.out.println(greeter.sayHello());

}

}

$

You can fix pom.xml by adding the following lines to pom.xml (within the <project> element):

<dependencies>

<dependency>

<groupId>joda-time</groupId>

<artifactId>joda-time</artifactId>

<version>2.9.2</version>

</dependency>

</dependencies>

This block of XML declares a list of dependencies for the project. Specifically, it declares a single dependency for the Joda Time library. Within the <dependency> element

## **Write a Test**

First add JUnit as a dependency to your pom.xml, in the test scope:

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>4.12</version>

<scope>test</scope>

</dependency>

Then create a test case like this:

$ mkdir -p src/test/java/hello/

$ cat src/test/java/hello/GreeterTest.java

package hello;

import static org.hamcrest.CoreMatchers.containsString;

import static org.junit.Assert.\*;

import org.junit.Test;

public class GreeterTest {

private Greeter greeter = new Greeter();

@Test

public void greeterSaysHello() {

assertThat(greeter.sayHello(), containsString("Hello"));

}

}

$ mvn test

## Summary

Congratulations! You’ve created a simple yet effective Maven project definition for building Java projects.

<https://spring.io/guides/gs/maven/>

**Step 4: Archetype is a Maven project templating toolkit that enables the creation of Maven project templates for users.**

$ mvn archetype:generate -DgroupId=com.javacodegeeks -DartifactId=SampleApplication -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

**Step 5: Navigate to the SampleApplication project folder and execute the command**

$ mvn package

**Step 6: The generated executable file can be executed using the following command:**

$ java –cptarget\SampleApplication-1.0-SNAPSHOT.jar com.javacodegeeks.App