**MAVEN**

**Q) What is maven?**

Maven is an application build tool. Help to build a java application.

**Q) What is application build tool?**

Which will help to develop an application.

**Q) Why maven need?**

To build a java application.

Egg:

Compile

.java --------> .class

Source target/executable

**Q) What is compiler?**

It is a software, which will help to convert from high level language to low level language (or) machine language.

1 java file ---------> 1 .class file

10 java files ---------> 10 .class files

100 java files ----------> 100 .class files

.java -----> .class (bundle) ----> .jar (bundle) ---->.war

Group of .class files is called .jar file (java archive)

Group of .jar files is called .war file (web archive)

.ear --> enterprise web archive

**Pre-requisites:**

---------------

Java ---> yum -y install java-1.8.0-openjdk

Java ---> yum -y install java-1.8.0-openjdk-devel

**Debian, ubuntu:**

---------------

- Sudo apt-get install openjdk-8-jre

Maven uses pom.xml (project object modules) to define the project and its dependencies. It will help to build the structure of the project, associate the files and help to build the application.

**Q) How to download and install maven?**

1) Go to maven site, and download .tar.gz file

2) Extract the file (tar xfs zpzxhe-mzven-xxx.tar.gz -C /opt/)

**3) Configure it for a user**

M2\_HOME=/opt/apache-maven-xxx

PATH=$PATH: $M2\_HOME/bin

configure on command line and make them permanent so place in user home\_dir ~/user/.bash\_profile

**4) mvn --version**

**Maven commands:**

---------------

1)mvn archetype:generate -> it will create a project structure

2)mvn validate ->it will validate project structure

3)mvn compile ->it will convert .java files into .class files

4)mvn test ->it will test .class files

5)mvn package ->it will help to create .jar,.war or .ear

6)mvn install ->it will install the package locally

7)mvn deploy ->it will install package remotely.

**Create a .jar file using maven:**

-------------------------------

**1) mvn archetype: generate**

project number :

group id : unique project name (technical adda)

artifact id : out put file name (simplejar.jar)

packaage id : same as group id

version : 1.0-SNAPSHOT

2)mvn validate

3)mvn compile

4)mvn test

5)mvn package

To see the content in the .jar file

jar -tvf target/simplejar-1.0-SNAPSHOT.jar

java -cp target/simplejar-1.0-SNAPSHOT.jar technicaladda.App

**Create a .war file using maven:**

-------------------------------

**1) mvn archetype: generate**

Project number:

group id : unique project name (techmine)

artifact id : out put file name (simpleweb.war)

packaage id : same as group id

version : 1.0-SNAPSHOT

2)mvn validate

3)mvn compile

4)mvn test

5) mvn package

**mvn deploy to nexus:**

-------------------

a)Nexus quickly becomes the only real and mature option for hosting third party artifacts, as well as for reusing internal artifacts across development streams.

- This will show how to deploy the artifacts of a project to Nexus, with Maven.

**b) Nexus Requirements in the pom.xml:**

-----------------------------------

- In order for Maven to be able to deploy the artifacts it creates in the package phase of the build, it needs to define the repository information where - the packaged artifacts will be deployed, via the distributionManagement element: write in <project> element not in <build> element

<distributionManagement>

<snapshotRepository>

<id>nexus-snapshots</id>

<url>http://localhost:8081/repository/maven-snapshots/</url>

</snapshotRepository>

</distributionManagement>

- A hosted, public Snapshots repository comes out of the box on Nexus, so there’s no need to create or configure anything further. Nexus makes it easy to determine the URLs of its hosted repositories

**c) Plugins:**

---------

By default, Maven handles the deployment mechanism via the maven-deploy-plugin – this mapped to the deployment phase of the default Maven lifecycle:

<plugin>

<artifactId>maven-deploy-plugin</artifactId>

<version>2.8.1</version>

<executions>

<execution>

<id>default-deploy</id>

<phase>deploy</phase>

<goals>

<goal>deploy</goal>

</goals>

</execution>

</executions>

</plugin>

- The maven-deploy-plugin is a viable option to handle the task of deploying to artifacts of a project to Nexus, but it was not built to take full advantage of what Nexus has to offer.

- Because of that fact, Sonatype built a Nexus specific plugin – the nexus-staging-maven-plugin – that is actually designed to take full advantage of the more advanced functionality that Nexus has to offer – functionality such as staging.

- Although for a simple deployment process we do not require staging functionality, we will go forward with this custom Nexus plugin since it was built with the clear purpose to talk to Nexus well.

- The only reason to use the maven-deploy-plugin is to keep open the option of using an alternative to Nexus in the future – for example, an Artifactory repository.

- However, unlike other components that may actually change throughout the lifecycle of a project, the Maven Repository Manager is highly unlikely to change, so that flexibility is not required.

- So, the first step in using another deployment plugin in the deploy phase is to disable the existing, default mapping:

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-deploy-plugin</artifactId>

<version>${maven-deploy-plugin.version}</version>

<configuration>

<skip>true</skip>

</configuration>

</plugin>

Now, we can define:

<plugin>

<groupId>org.sonatype.plugins</groupId>

<artifactId>nexus-staging-maven-plugin</artifactId>

<version>1.5.1</version>

<executions>

<execution>

<id>default-deploy</id>

<phase>deploy</phase>

<goals>

<goal>deploy</goal>

</goals>

</execution>

</executions>

<configuration>

<serverId>nexus</serverId>

<nexusUrl>http://localhost:8081/nexus/</nexusUrl>

<skipStaging>true</skipStaging>

</configuration>

</plugin>

- The deploy goal of the plugin is mapped to the deploy phase of the Maven build.

- Also notice that, as discussed, we do not need staging functionality in a simple deployment of -SNAPSHOT artifacts to Nexus, so that is fully disabled via the <skipStaging> element.

**d) The Global settings.xml:**

-------------------------

- Deployment to Nexus is a secured operation – and a deployment user exists for this purpose out of the box on any Nexus instance.

- Configuring Maven with the credentials of this deployment user, so that it can interact correctly with Nexus, cannot be done in the pom.xml of the project.

- This is because the syntax of the pom doesn’t allow it, not to mention the fact that the pom may be a public artifact, so not well suited to hold credential information.

- The credentials of the server have to be defined in the global Maven setting.xml:

<servers>

<server>

<id>nexus-snapshots</id>

<username>deployment</username>

<password>the\_pass\_for\_the\_deployment\_user</password>

</server>

</servers>

- The server can also be configured to use key based security instead of raw and plaintext credentials.

Maven deploy to sonarqube for codequality:

-----------------------------------------

**1) Settings.xml:**

--------------

- edit the global settings file "/opt/maven-3.6.1/conf/settings.xml"

- place the below commands in <settings> element

<pluginGroups>

<pluginGroup>org.sonarsource.scanner.maven</pluginGroup>

</pluginGroups>

and place below also

<profiles>

<profile>

<id>sonar</id>

<activation>

<activeByDefault>true</activeByDefault>

</activation>

<properties>

<!-- Optional URL to server. Default value is http://localhost:9000 -->

<sonar.host.url>

http://(sonar URL):9000/sonar

</sonar.host.url>

</properties>

</profile>

</profiles>

2)edit the pom.xml:

------------------

place the plugin in "pom.xml" file

<plugins>

<plugin>

<groupId>org.sonarsource.scanner.maven</groupId>

<artifactId>sonar-maven-plugin</artifactId>

<version>3.6.0.1398</version>

</plugin>

</plugins>

**3) Deploy to sonarqube:**

---------------------

- to deploy to sonarqube

mvn soanr:soanr -Dsonar.login=admin -Dsoanr.password=admin

or

mvn org.sonarsource.scanner.maven:sonar-maven-plugin:3.2:sonar