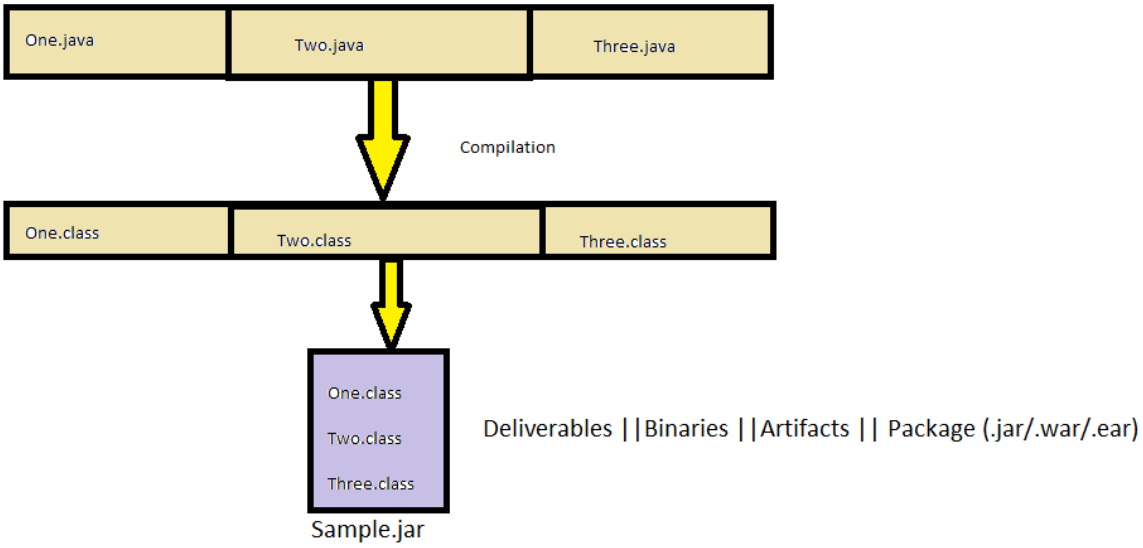

DEVOPS
Maven™

MAVEN

BUILD MANAGEMENT

- It is the process of compiling and assembling/grouping a software system.
- Build automation is the nothing but act of scripting.
    - Compiling source code.
    - Packaging binaries.
    - Running automated tests.
    - Deploying to production system.
    - Creating documentation.



BUILD MANAGEMENT TOOL

A build management tool takes care of everything for building a process. It does following:

- Generates source code (if auto-generated code is used).
- Generates documentation from source code.
- Compiles source code.
- Packages compiled code into JAR of ZIP file.
- Installs the packaged code in local repository, server repository, or central repository.

Ex: ANT, Maven and Gradle.

Maven (Apache Maven)

- Maven is a software project management tool. It is open source software from Apache software foundation.
- It is used for building, reporting and documenting a Software project. It is mainly based on POM (Project Object Model).
- Maven provides superset of features found in a build tools.
- It simplifies the build process like ANT. But it is too much advanced than ANT.
- Maven is used for only java applications.
- Maven runs on java.

Without Maven

There are many problems that we face during the project development.

Adding set of Jars in each project:

In case of struts, spring, hibernate frameworks, we need to add set of jar files in each project. It must include all the dependencies of jars also.

Creating the right project structure:

We must create the right project structure in servlet, struts etc, otherwise it will not be executed.

Building and deploying the project:

We must have to build and deploy the project so that it may work.

With Maven

Maven simplifies the above mentioned problems. It does mainly following tasks.

- It makes a project easy to build.
- It provides uniform build process (maven project can be shared by all the maven projects).
- It provides project information (log document, cross referenced sources, mailing list, dependency list, unit test reports etc.).
- It is easy to migrate for new features of Maven
- Apache maven helps to manage.
    - Builds
    - Documentation
    - Reporting (JUnit/Jacaco)
    - SCMs

- Releases
- Distribution
- Mailing list

Features of Maven

Simple:

Maven provides simple project setup that is based on best practices.

Fast:

We can get a new project or module started in a few seconds in Maven.

Easy to learn:

Maven usage and commands are easy to learn across all projects. Therefore ramp up time for new developers coming onto a project is very less.

Dependency management:

Maven provides superior dependency management including automatic updates and transitive dependencies.

Multiple Projects:

We can easily work with multiple projects at the same time by using Maven.

Large Library:

Maven has a large and growing repository of libraries and metadata to use out of the box.

Extensible:

Maven supports the ability to easily write plugins in Java or scripting languages for extending its core functionality.

Instant:

Maven is online and it provides instant access to new features with very less configuration.

Advantages of Maven

Maven has a long list of advantages for Software development. Below are the some of the main advantages.

Common Project Structure:

By using Maven, every developer has a common project structure that helps in understanding the code as well as developing new features in a new project.

Modular Design:

Maven promotes modular design that divides a complex project into multiple modules that are easier to manage. By using Maven, it is easier to manage multiple modules for build, test, release etc.

Centralized Dependency Management:

With Maven, each developer does not have to include the jars separately in each project or module. Maven provides a centralized dependency management that can help improve efficiency of software development.

Fewer Decisions:

With Maven a developer has to make fewer decisions about things unrelated to software development work. The project structure comes ready with Maven, dependency management is a uniform approach and build/release is handled by Maven. So a developer can focus on core work of developing software.

Disadvantages of Maven

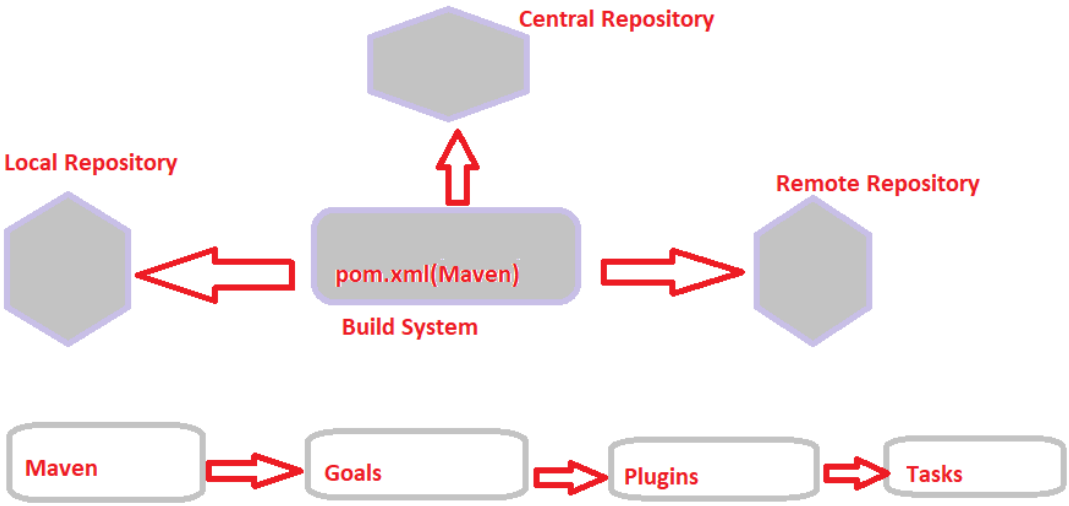
- Maven needs the maven installation in the system for working and maven plugin for the IDE (Eclipse and Netbeans).
- If the maven code for an existing dependency is not available, then one cannot add that dependency using maven.
- You need to know maven command line or use an ide that has maven integration, such as netbeans or eclipse.
- If you have a dependent jar that isn't mavenized, you might lose your mind before you figure out how to integrate it.

Ant Vs Maven

ANT	MAVEN
Ant is a Java library and command line toolbox for build process.	Maven is a framework for many aspects of software development like- project setup, compile, build, documentation etc.
Ant does not have any conventions for project structure or build processes.	Maven has conventions for setting up project structure as well as for build processes.
Ant is based on procedural programming. We have to write code	Maven is based on declarative programming. We have to just configure it

for compilation build, copy etc.	for our project setup and programming.
Ant does not impose any lifecycle. We need to create the sequence of tasks manually.	Maven has a lifecycle for software build processes. There are well-defined phases that we can use in Maven.
Ant scripts are not reusable in multiple projects.	Maven has plugins that are reusable across multiple projects.
It is less preferred than Maven.	It is more preferred than Ant.

Maven Architecture



Maven Repositories

- A maven repository is a place i.e. a directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and this can be used by Maven easy.
- Maven searches for dependencies in the repositories. There are 3 types of maven repository:
  1. Local Repository
  2. Central Repository
  3. Remote Repository
- Maven searches for the dependencies in the following order:

Local repository then Central repository then Remote repository.
(Local repository --> Central repository --> Remote repository).

Note: If dependency is not found in these repositories, maven stops processing and throws an error.

1. Local Repository

- A Maven local repository is located in your local system. It is created by the maven when you run any maven command.
- By default, maven local repository is %USER_HOME%/.m2 directory. C:\Users\<user-name>\.m2
- We can change the location of maven local repository by changing the settings.xml file. It is located in MAVEN_HOME/conf/settings.xml, for example: E:\apache-maven-3.1.1\conf\settings.xml.

Let's see the default code of settings.xml file.

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
 <!-- localRepository
  | The path to the local repository maven will use to store artifacts.
  |
  | Default: ${user.home}/.m2/repository
 <localRepository>/path/to/local/repo</localRepository>
 -->
 ...
</settings>
...
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository>e:/mavenlocalrepository</localRepository>

...
</settings>
```

2. *Central Repository*

- Maven central repository is located on the web. It has been created by the apache maven community itself.
- The path of central repository is: http://repo1.maven.org/maven2/.
- The central repository contains a lot of common libraries that can be viewed by this url http://search.maven.org/#browse.

3. *Remote Repository*

- Maven remote repository is located on the web.
- Most of libraries can be missing from the central repository such as JBoss library etc, so we need to define remote repository in pom.xml file.

MAVEN Lifecycle

When Maven starts building a project, it steps through a defined sequence of phases and executes goals, which are registered with each phase. Maven has the following three standard lifecycles.

- Clean
- Default (Build Lifecycle)
- Site

Maven clean: This phase clean the temporary and runtime files.

Maven Site: This phase is generally used to create fresh documentation to create reports, deploy site, etc.

Default OR Build Lifecycle

A Build Lifecycle is a well-defined sequence of phases, which define the order in which the goals are to be executed. Here phase represents a stage in life cycle. A typical Maven Build Lifecycle consists of the following sequence of phases.

- prepare-resources
- validate
- compile
- test
- package
- install
- deploy

prepare-resources --> resource copying --> Resource copying can be customized in this phase.
validate --> Validating the information --> Validates if the project is correct and if all necessary information is available.
compile --> compilation --> Source code compilation is done in this phase.
Test --> Testing --> Tests the compiled source code suitable for testing framework.
package --> packaging -->This phase creates the JAR/WAR package as mentioned in the packaging in POM.xml.
install --> installation -->This phase installs the package in local/remote maven repository.
deploy --> Deploying --> Copies the final package to the remote repository/server.

Directory Structure in Maven Project

src\
src\main
src\main\java\
1.java
2.java
src\test\
src\test\java\
test1.java
test2.java

Environment setup

System Requirement

JDK	:	1.7 or above.
Memory	:	No minimum requirement.
Disk Space	:	No minimum requirement.
OS	:	No minimum requirement.

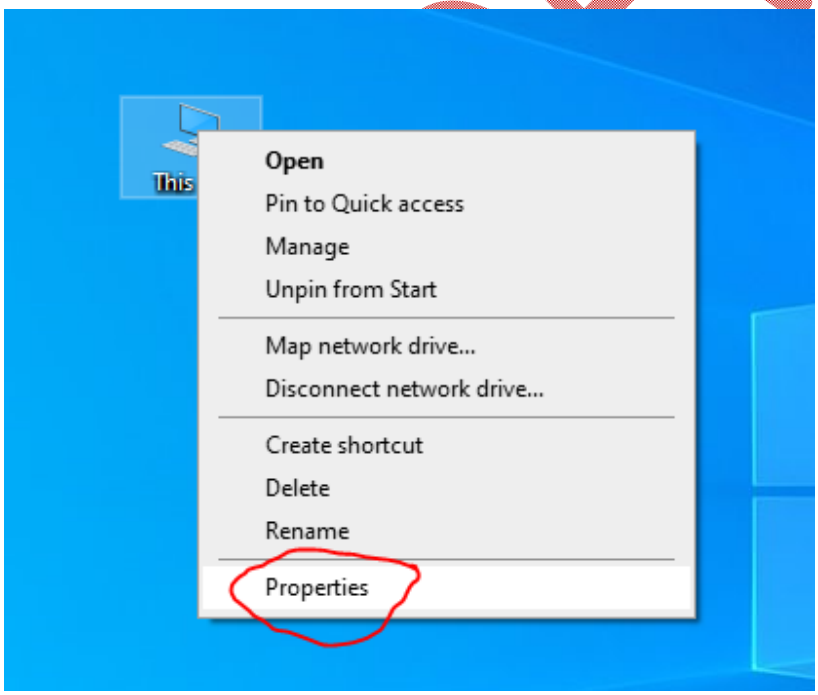
Windows

1. Download and install jdk below url
https://corretto.aws/downloads/latest/amazon-corretto-11-x64-windows-jdk.msi

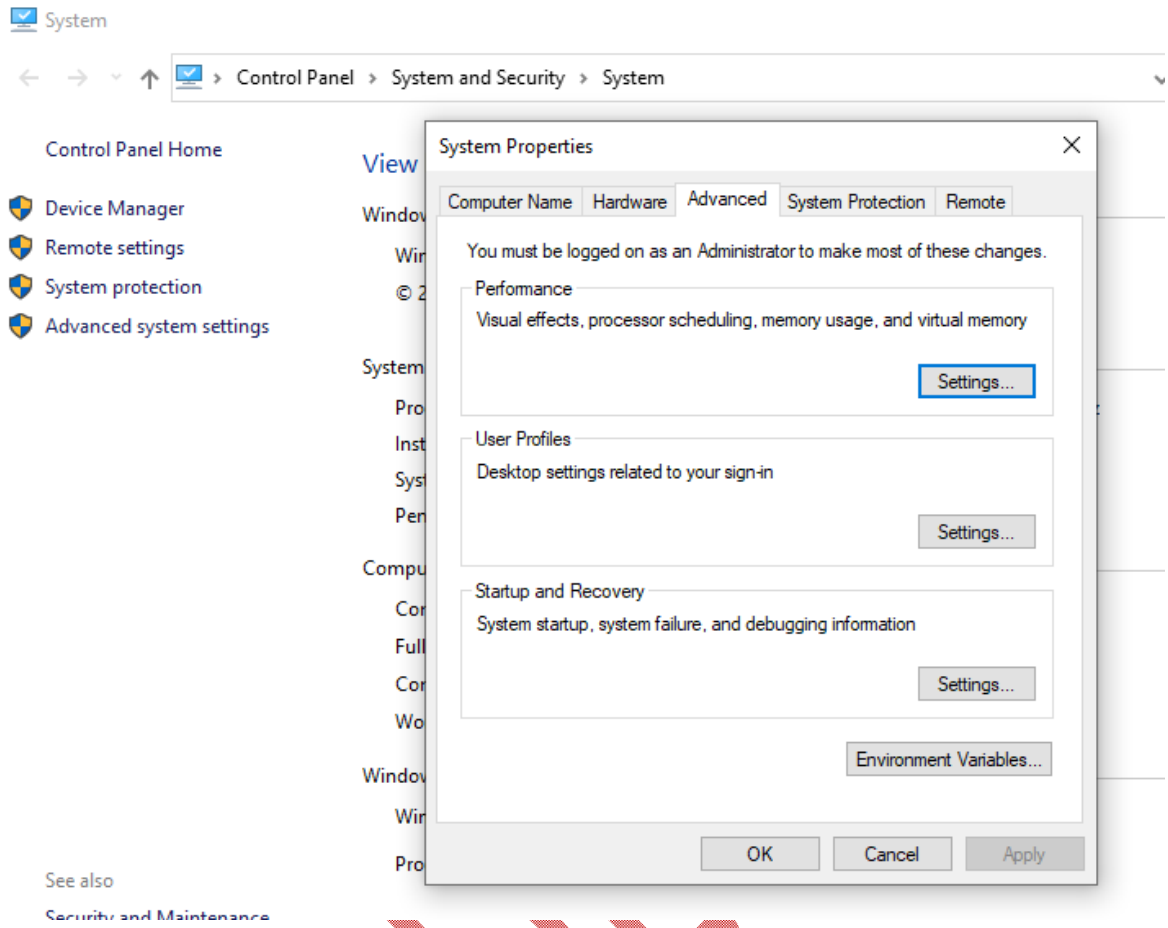
2. Download and unzip the maven.
https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.zip

3. Setup the environment variable for java and maven.
JAVA_HOME=C:\Program Files\Java\jdk1.8.0_65\
MAVEN_HOME=D:\Java\maven\
PATH=C:\ProgramFiles\Intel\WiFi\bin\;C:\ProgramFiles\Common Files\Intel\WirelessCommon\;%JAVA_HOME%\bin\;%MAVEN_HOME%\bin\;

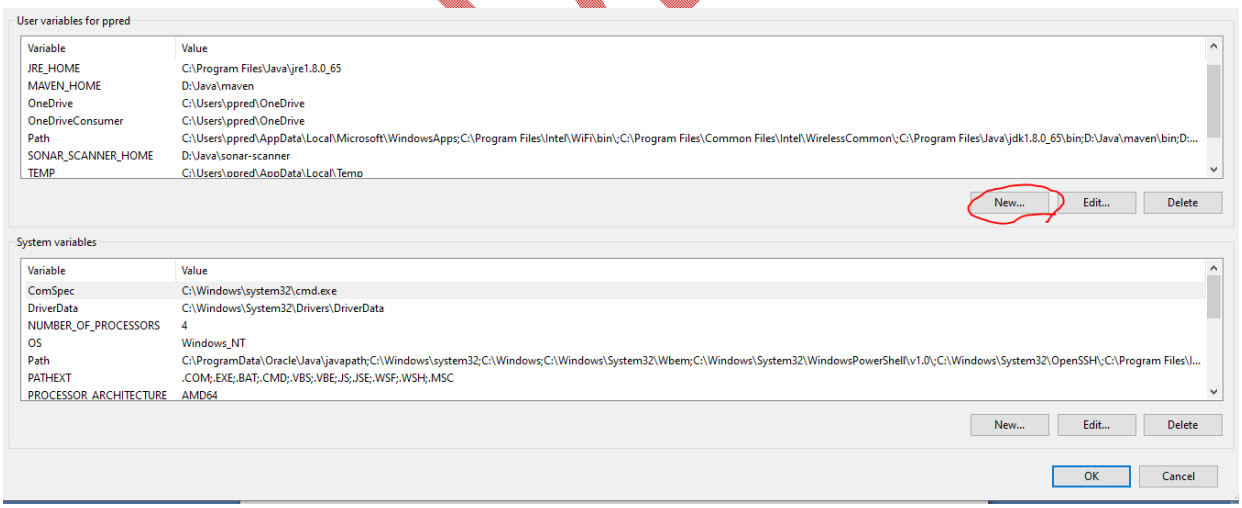
Go to this PC → Properties →

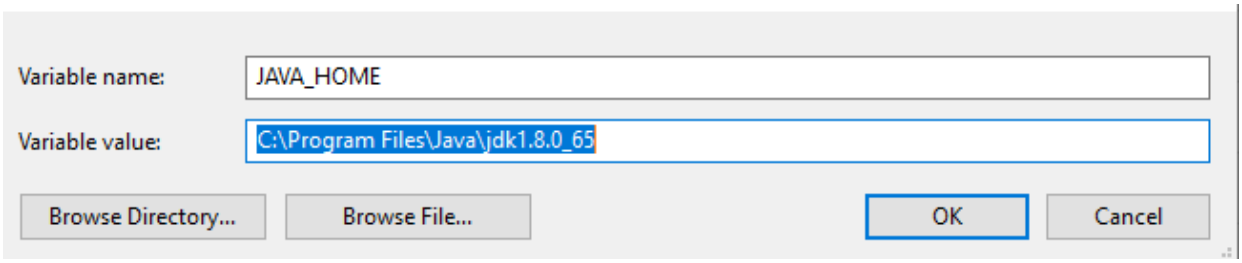


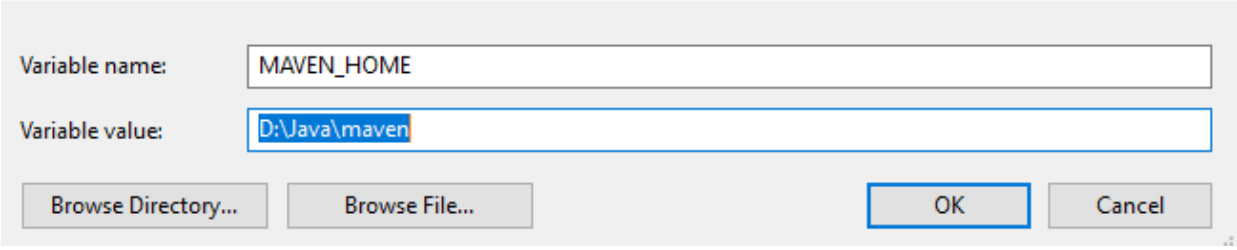
Click on Advanced system settings → Environment Variables

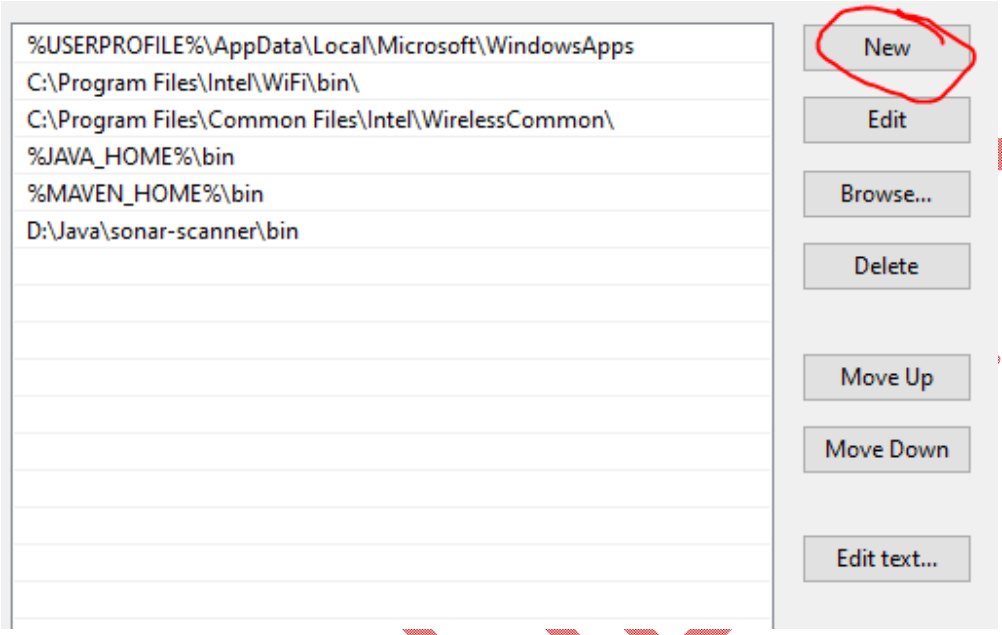


Click on new Add JAVA_HOME and MAVEN_HOME and update the PATH







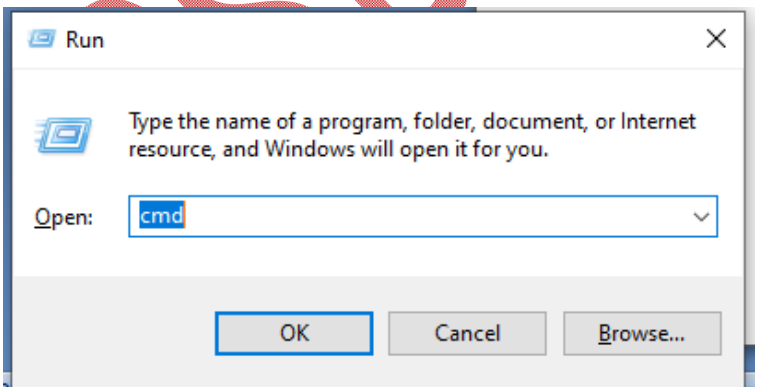


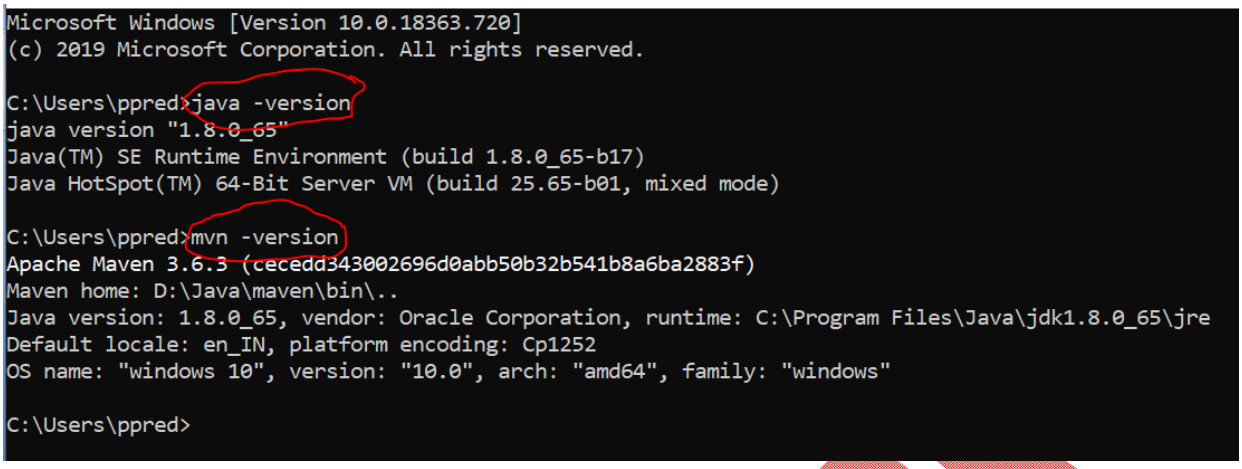
4. Verify the java and maven version in command prompt.

Click windows+R and enter cmd → run the below commands

java –version

mvn –version





Maven installation on Ubuntu

1. Update and install below java commands

sudo apt-get update

sudo add-apt-repository ppa:openjdk-r/ppa

sudo apt-get install openjdk-8-jdk

2. Download and configure maven

cd /home/ubuntu

sudo wget https://mirrors.estointernet.in/apache/maven/maven-3/3.6.3/binaries/apache-maven-3.6.3-bin.tar.gz

sudo tar -xvf apache-maven-3.6.3-bin.tar.gz

sudo mv apache-maven-3.6.3 maven

sudo chmod 777 -R maven

3. Update the .profile and add the below entries

JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64

export JAVA_HOME

MAVEN_HOME=/home/ubuntu/maven

export MAVEN_HOME

M2=/home/ubuntu/maven/bin

export M2

PATH=$PATH:$M2:$JAVA_HOME/bin

export PATH

1. Verifying java and maven versions

```
ubuntu@ip-172-31-23-216:~$ java -version
openjdk version "11.0.7" 2020-04-14 LTS
OpenJDK Runtime Environment Corretto-11.0.7.10.1 (build 11.0.7+10-LTS)
OpenJDK 64-Bit Server VM Corretto-11.0.7.10.1 (build 11.0.7+10-LTS, mixed mode)
ubuntu@ip-172-31-23-216:~$ mvn -version
Apache Maven 3.6.3 (cecedd343002696d0abb50b32b541b8a6ba2883f)
Maven home: /opt/maven
Java version: 11.0.7, vendor: Amazon.com Inc., runtime: /opt/java
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "5.4.0-1015-aws", arch: "amd64", family: "unix"
ubuntu@ip-172-31-23-216:~$
```

Creating java project using maven

1. Run the below command in command prompt.

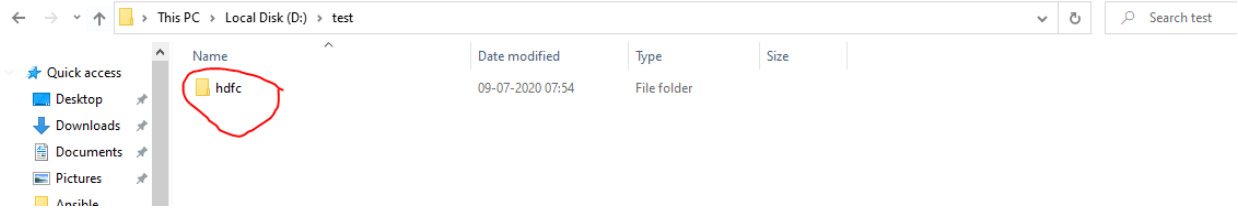
mvn archetype:generate -DgroupId=retail -DartifactId=hdfc -Dversion=1.0-SNAPSHOT -Dpackaging=jar -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false

archetypeArtifactId: archetypeArtifactId is nothing but the template used for creating this Java project.

```
D:\test>mvn archetype:generate -DgroupId=retail -DartifactId=hdfc -Dversion=1.0-SNAPSHOT -Dpackaging=jar -DarchetypeArtifactId=maven-archetype-quicks
tart -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-quickstart:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: D:\test
[INFO] Parameter: package, Value: retail
[INFO] Parameter: groupId, Value: retail
[INFO] Parameter: artifactId, Value: hdfc
[INFO] Parameter: packageName, Value: retail
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: D:\test\hdfc
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.946 s
[INFO] Finished at: 2020-07-09T07:54:34+05:30
[INFO] ------------------------------------------------------------------------

D:\test>
```

2. Verify the project created OR not.



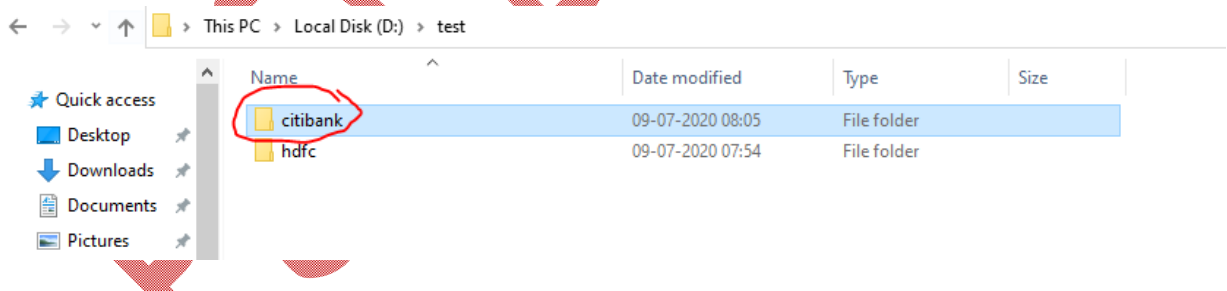
Generating web project using maven

1. Run the below command in cmd prompt

mvn archetype:generate -DgroupId=com.citi -DartifactId=citibank -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false

```
D:\test>mvn archetype:generate -DgroupId=com.citi -DartifactId=citibank -DarchetypeArtifactId=maven-archetype-webapp -DinteractiveMode=false
[INFO] Scanning for projects...
[INFO]
[INFO] ------------------< org.apache.maven:standalone-pom >-------------------
[INFO] Building Maven Stub Project (No POM) 1
[INFO] --------------------------------[ pom ]---------------------------------
[INFO]
[INFO] >>> maven-archetype-plugin:3.1.2:generate (default-cli) > generate-sources @ standalone-pom >>>
[INFO]
[INFO] <<< maven-archetype-plugin:3.1.2:generate (default-cli) < generate-sources @ standalone-pom <<<
[INFO]
[INFO]
[INFO] --- maven-archetype-plugin:3.1.2:generate (default-cli) @ standalone-pom ---
[INFO] Generating project in Batch mode
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-webapp/1.0/maven-archetype-webapp-1.0.pom (
533 B at 554 B/s)
Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-webapp/1.0/maven-archetype-webapp-1.0.jar (
3.9 kB at 5.0 kB/s)
[INFO] ----------------------------------------------------------------------------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0
[INFO] ----------------------------------------------------------------------------
[INFO] Parameter: basedir, Value: D:\test
[INFO] Parameter: package, Value: com.citi
[INFO] Parameter: groupId, Value: com.citi
[INFO] Parameter: artifactId, Value: citibank
[INFO] Parameter: packageName, Value: com.citi
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: D:\test\citibank
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  12.824 s
```

2. Verify the project



pom.xml (Project Object Model)

- A Project Object Model or POM is the fundamental unit of work in Maven.
- It is an XML file that contains information about the project and configuration details used by Maven to build the project.

Sample pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/maven-v4_0_0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>retail</groupId>
  <artifactId>hdfc</artifactId>
  <version>1.0-SNAPSHOT</version>
  <packaging>jar</packaging>

  <name>hdfc</name>
  <url>http://maven.apache.org</url>
  <dependencies>
    <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>3.8.1</version>
      <scope>test</scope>
    </dependency>

  </dependencies>
</project>
```


Project root:

This is project root tag. You need to specify the basic schema settings such as apache schema and w3.org specification

modelVersion:

- It is the sub element of project.
- modelVersion is the version of project descriptor your POM conforms to.
- It needs to be included and is set. The value 4.0.0 just indicated that it is compatible Maven 3.
- modelVersion should be 4.0.0 for maven 3.0 +

groupId:

- This is an Id of project's group.
- groupId uniquely identifies your project across all projects. A group ID should follow Java's package name rules. This means it starts with a reversed domain name you control.

Ex: org.apache.maven, org.apache.commons
com.company.bank

artifactId:

- This is an Id of the project. This is generally name of the project/jar.
- artifactId is the name of the jar without version. If you created it, then you can choose whatever name you want with lowercase letters and no strange symbols.

Ex: maven, commons-math
consumer-banking

version:

This is the version of the project. Along with the groupId, It is used within an artifact's repository to separate versions from each other.

Ex: 1.0.0
1.0.1

Project Name (GAV):

Ex: com.company.bank:consumer-banking:1.0.0
com.company.bank:consumer-banking:1.1.1
retail:hdfc:1.0-SNAPSHOT

Goal:

A goal represents a specific task that contributes to the building and managing of a project.

prepare-resources --> mvn process-resources

validate --> mvn validate

compile --> mvn compile

test --> mvn test

package --> mvn package

install --> mvn install

deploy --> mvn deploy

List of plugins: mvn fr.jcgay.maven.plugins:buildplan-maven-plugin:list

Plugins path: .m2\repository\org\apache\maven\plugins\

package/applications installation path: .m2\repository\retail\hdfc\1.0-SNAPSHOT\

Maven Repositories

1. *Local Repository*

A Maven local repository is located in your local system. It is created by the maven when you run any maven command.
By default, maven local repository is %USER_HOME%/.m2 directory. For example: C:\Users\<user-name>\.m2.

Update location of Local Repository
We can change the location of maven local repository by changing the settings.xml file. It is located in MAVEN_HOME/conf/settings.xml, for example: E:\apache-maven-3.1.1\conf\settings.xml.

Let's see the default code of settings.xml file

```
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
```

```
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
 <!-- localRepository
  | The path to the local repository maven will use to store artifacts.
  |
  | Default: ${user.home}/.m2/repository
 <localRepository>/path/to/local/repo</localRepository>
 -->

...
</settings>

<localRepository>/home/ubuntu/hdfcrepo</localRepository>

...
<settings xmlns="http://maven.apache.org/SETTINGS/1.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
http://maven.apache.org/xsd/settings-1.0.0.xsd">
  <localRepository>e:/mavenlocalrepository</localRepository>

...
</settings>
```

As you can see, now the path of local repository is e:/mavenlocalrepository.

2. *Maven Central Repository*

- Maven central repository is located on the web. It has been created by the apache maven community itself.
- The path of central repository is: https://repo.maven.apache.org/maven2
- The central repository contains a lot of common libraries that can be viewed by this url http://search.maven.org/#browse.

```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
  <groupId>com.companyname.projectgroup</groupId>
  <artifactId>project</artifactId>
  <version>1.0</version>
  <dependencies>
    <dependency>
      <groupId>com.companyname.common-lib</groupId>
      <artifactId>common-lib</artifactId>
      <version>1.0.0</version>
    </dependency>
  <dependencies>
        <repositories>
                <repository>
                <id>central</id>
                <name>Maven Repository Switchboard</name>
                <url>http://repo1.maven.org/maven2</url>
                </repository>
        </repositories>
        </project>
```

3. *Maven Remote Repository*

- Maven remote repository is located on the web.
- Maven does not find a mentioned dependency in central repository as well. It then stops the build process and output error message to console. To prevent such situation, Maven provides concept of Remote Repository, which is developer's own custom repository containing required libraries or other project jars.
- Maven will download dependency (not available in central repository) from Remote Repositories mentioned in the same pom.xml.

```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
  xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
  http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.companyname.projectgroup</groupId>
  <artifactId>project</artifactId>
  <version>1.0</version>
```

```
  <dependencies>
    <dependency>
      <groupId>com.companyname.common-lib</groupId>
      <artifactId>common-lib</artifactId>
      <version>1.0.0</version>
    </dependency>
  <dependencies>
  <repositories>
    <repository>
      <id>companyname.lib1</id>
      <url>http://download.companyname.org/maven2/lib1</url>
    </repository>
    <repository>
      <id>companyname.lib2</id>
      <url>http://download.companyname.org/maven2/lib2</url>
    </repository>
  </repositories>
</project>
```


Maven Dependency Search Sequence

When we execute Maven build commands, Maven starts looking for dependency libraries in the following sequence.

Step 1 − Search dependency in local repository, if not found, move to step 2 else perform the further processing.

Step 2 − Search dependency in central repository, if not found and remote repository/repositories is/are mentioned then move to step 4. Else it is downloaded to local repository for future reference.

Step 3 − If a remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency).

Step 4 − Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference. Otherwise, Maven stops processing and throws error (Unable to find dependency).

Note:

- Dependencies are downloaded from repositories Via http
- Downloaded dependencies are cached in a local repositories under .m2 location.(${user.home}/.m2/repository)

Maven Plugins

- The maven plugins are central part of maven framework, it is used to perform specific goal.
- Maven is actually a plugin execution framework where every task is actually done by plugins.
- Maven Plugins are generally used to –
  - create jar file
  - create war file
  - compile code files
  - unit testing of code
  - create project documentation
  - create project reports
- A plugin generally provides a set of goals, which can be executed using the following syntax –

  mvn [plugin-name]:[goal-name]

Ex: A Java project can be compiled with the maven-compiler-plugin's compile-goal by running the following command.

mvn compiler:compile

Types of Plugins:

Maven provided the following two types of Plugins

1. Build Plugins
2. Reporting Plugins

Build Plugins:

These plugins are executed at the time of build. These plugins should be declared inside the <build> element.

Reporting Plugins:

They execute during the site generation process and they should be configured in the <reporting> element of the pom.xml.

Maven Core/Common plugins:

clean --> clean up after build.
compiler --> compiles java source code.
deploy --> deploys the artifact to the remote repository.
failsafe --> runs the JUnit integration tests in an isolated classloader.
install --> installs the built artifact into the local repository.
resources --> copies the resources to the output directory for including in the JAR.
site --> generates a site for the current project.
surefire --> runs the JUnit unit tests in an isolated classloader.
verifier --> verifies the existence of certain conditions. It is useful for integration tests.

Note: To see the list of maven plugins, you may visit apache maven official website http://repo.maven.apache.org/maven2/org/apache/maven/plugins/.
Maven plugins are also available outside the maven at codehaus.org and code.google.com.

Maven plugin examples

- Compiler plugin (Specify a specific version of a compiler to use)
- Docker plugin (Used to create docker images)
- tomcat server plugin (configuring a web server to deploy over code on tomcat web server)
- Maven sonatype nexus plugin (Nexus is a Maven remote repository)
- ANT plugin
- exec plugin

During plugin configuration three things we need to remember

1. What is the plugin
2. When to use the plugin
3. What that plugin exactly do

mvn <goals> --> Plugins --> task
mvn compile --> Compiler plugin -->run --> task

Ex: 1

```
<project xmlns = "http://maven.apache.org/POM/4.0.0"
xmlns:xsi = "http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation = "http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>com.companyname.projectgroup</groupId>
<artifactId>project</artifactId>
<version>1.0</version>
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-antrun-plugin</artifactId>
      <version>1.1</version>
      <executions>
        <execution>
          <id>id.clean</id>
          <phase>compile</phase>
          <goals>
            <goal>run</goal>
          </goals>
          <configuration>
            <tasks>
              <echo>clean phase</echo>
            </tasks>
          </configuration>
        </execution>
      </executions>
    </plugin>
  </plugins>
</build>
</project>
```


Ex: 2

```
<build>
    <plugins>
            <plugin>
```

```
      <groupId>org.codehaus.mojo</groupId>
      <artifactId>exec-maven-plugin</artifactId>
      <version>1.6.0</version>
         <configuration>
          <executable>git</executable>
                        <arguments>
                         <argument> --version</argument>
                        </arguments>
         </configuration>
       </plugin>

     </plugins>

</build>
```

Multi module projects

- Maven supports multi module projects.
- A parent pom is used group of all modules.

Syntax:

Root POM/Super POM/ Parent POM:

```
    <groupId>com.retail</groupId>
    <artifactId>RBS</artifactId>
    <version>1.0-SNAPSHOT</version>
    <packaging>pom</packaging>
        <modules>
         <module>child1</module>
         <module>child2</module>
          <module>child3</module>
        </modules>
```

Child POM

```
<dependency>
   <groupId>com.retail</groupId>
   <artifactId>child3</artifactId>
   <version>1.0-SNAPSHOT</version>
</dependency>
<parent>
```

```
   <groupId>com.retail</groupId>
    <artifactId>RBS</artifactId>
    <version>1.0-SNAPSHOT</version>
</parent>
```

Reactor:

- The reactor determines the correct build order from the dependencies stated by each project in their respective project descriptors, and will then execute a stated set of goals.
- It can be used for both building projects and other goals, such as site generation.

Snapshot And Release versions

Snapshot Version

- Snapshot version is an version which is currently under development.
- If our project depends on snapshot version, every time we build maven get latest copy from central/remote repository.

Note: If a version is ending with -SNAPSHOT then it is snapshot version otherwise it is RELEASE version.

Example:

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>4.3.10-SNAPSHOT</version>
</dependency>

  <groupId>retail</groupId>
  <artifactId>hdfc</artifactId>
  <version>1.0.1-SNAPSHOT</version>
  <packaging>war</packaging>
```

Release Version:

- A version whose development is completed is called as release version.

Note: If a version is not ending with -SNAPSHOT then it is RELEASE version.

```
<dependency>
   <groupId>org.springframework</groupId>
   <artifactId>spring-core</artifactId>
   <version>4.3.10</version>
</dependency>

<groupId>retail</groupId>
 <artifactId>hdfc</artifactId>
 <version>1.0.1</version>
 <packaging>war</packaging>
```

External Dependency concept

Maven dependency management uses the concept of Maven Repositories (Local, Central, Remote). Suppose dependency is not present in any of remote repositories and central repository then in such case Maven uses the concept of External Dependency.

Configuration of external dependency

External dependencies (library jar location) can be configured in pom.xml in same way as other dependencies are configured.

- First, specify groupId the same as the name of the library.
- Then specify artifactId the same as the name of the library.
- Thirdly, specify scope as a system.
- Lastly, specify the system path relative to the project location

Interview Questions

1. What is Junit?

Junit is a framework for automating unit testing, whenever we add new features to the software we need to retested with all functionalities, developers write Junit test cases.

2. What is integration testing?

Integration test cases are written by QA team, integrations testing is testing end to end flow.QA teams use tools like selenium, QAT for automating integration testing. This can be integrated with build tool like maven.

3. What is war/ear/jar ?

War stands WebArchive, it a format to package web applications.Ear stands for Enterprise Archive, this format is for EJB based applications.Jar stands for Java Archive.In pom.xml we can declare packaging tag.

Ex:

```
<groupId>retail</groupId>
<artifactId>hdfc</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>jar</packaging>

<groupId>retail</groupId>
<artifactId>hdfc</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>war</packaging>

<groupId>retail</groupId>
<artifactId>hdfc</artifactId>
<version>1.0-SNAPSHOT</version>
<packaging>ear</packaging>
```

4. *What is dependency management?*

Dependency:

We our project wants to use a framework, frameworks come as a jar file, this jar is our project dependency.

Transitive Dependency:

A dependency on which our dependency depends on

Ourproject -----> a.jar ---> b.jar

a.jar is our dependency and b.jar is our transitive dependency

Dependency Management:

Maven automatically manages dependencies and transitive dependencies by downloading them from maven repositories.

5. *What is Maven target directory?*

Target directory is created by maven to organise project out files

6. *How to Skip test cases in maven ?*

mvn package -DskipTests

7. *What is Maven settings.xml?*

This file contains certain information about maven settings, for example

- Local repository path
- Remote repository credentials
- Remote server details like credentials.

Ex:

```
<servers>
    <server>
        <id>TomcatServer</id>
        <username>tomcat</username>
        <password>tomcat</password>
    </server>
</servers>
```

- Details about central repositories
- Proxy settings, etc

There are two locations where a settings.xml file may live:

The Maven install: ${maven.home}/conf/settings.xml
A user's install: ${user.home}/.m2/settings.xml

8. In Maven what are the two setting files called and what are their location?

In Maven, the setting files are called settings.xml, and the two setting files are located at

- Maven installation directory: $M2_Home/conf/settings.xml
- User's home directory: ${ user.home }/ .m2 / settings.xml

9. List out the build, source and test source directory for POM in Maven?

Build = Target
Source = src/main/java
Test = src/main/test

10. For POM what are the minimum required elements?

The minimum required elements for POM are project root, modelVersion, groupID, artifactID and version.

11. Explain how you can produce execution debug output or error messages?

To produce execution debug output you could call Maven with X parameter or e parameter
Ex: maven -e clean install,maven -X clean install

11. Explain how to run test classes in Maven?

To run test classes in Maven, you need surefire plugin, check and configure your settings in setting.xml and pom.xml for a property named "test."

12. Mention how profiles are specified in Maven?

Profiles are specified in Maven by using a subset of the elements existing in the POM itself.

13. What does it mean when you say Maven uses Convention over Configuration?

In the case of Configuration, developers have to create the build processes manually and they have to specify each and every configuration in detail. But,

Maven uses convention where the developers need not create the build processes manually.
Also, for convention, users do not need to specify the configuration in detail. Once a developer creates a project in Maven then Maven will automatically create a structure. Developers just have to place the files appropriately. There is no need to specify any configuration details in pom.xml file.

14. What is Build Profile?

A Build profile is a set of configuration values that can be used to set or override default values of Maven build.
Using a build profile, you can customize build for different environments such as Production and the Development environments.

15. How can you activate profiles?

A Maven Build Profile can be activated in the following ways –

. Explicitly using command console input.
. Through maven settings.
. Based on environment variables (User/System variables).
. OS Settings (for example, Windows family).
. Present/missing files.

16. What is optional dependency in maven?

You can mark any transitive dependency as optional using the “optional” element.
As an example, A depends upon B and B depends upon C. Now B marked C as optional. Then A will not use C.

17. What is dependency exclusion?

You can exclude any transitive dependency using the “exclusion” element.
Suppose if A depends upon B and B depends upon C then A can be marked C as excluded.

18. How many project types available in Maven to choose from?

There is more than thousand Java project as there are templates, skeleton provided to you by Maven so that you do not have to remember a basic configuration detail or a basic setup of that particular type of project which Maven

is going to give it to you.
It includes examples like basic Java project, Spring Project, Spring MVC, Spring Web Flow, and Spring Boot.

19. What phases does a Clean Lifecycle consist?

The clean lifecycle consists of the following phases –

. pre-clean
. clean
. post-clean

20. What phases does a Site Lifecycle consist?

The phases in Site Lifecycle are –

. pre-site
. site
. post-site
. site-deploy

21. What is a MOJO?

A MOJO stands for Maven plain Old Java Object. Each MOJO is an executable goal in Maven, and a plugin is a distribution of one or more related MOJOs.

22. What is a Repository in Maven?

A repository is a location on file system where build artifacts, jars, dependencies and pom.xml files are stored.

23. Why we should not store jars in CVS or any other version control system instead of Maven repository?

Maven recommends storing jars in local repository instead of CVS or any other version control system. There are following advantages of storing it in Maven repo vs. CVS:

- **Less Storage:** A repository is very large, but it takes less space because each JAR is stored only in one place. E.g. If we have 10 modules dependent on Spring jar, then they all refer to same Spring jar stored in local repository.

- **Quicker Checkout:** Project checkout is quicker from local repository, since there is not need to checkout jars if they are already present in repo.
- **No need for versioning:** There is no need to version JARS since external dependencies do not change so often.

24. Can anyone upload JARS or artifacts to Central Repository?

No, we need special permissions to upload JARS and artifacts to Central Maven Repository.

25. How can we exclude a dependency in Maven?

To exclude a dependency we can add the tag under the section of the pom.

Ex:

```
        <dependencies>
<dependency>
 <groupId>test.ProjectX</groupId>
 <artifactId>ProjectX</artifactId>
 <version>1.0</version>
 <scope>compile</scope>
 <exclusions>
  <exclusion> <!-- exclusion is mentioned here -->
  <groupId>test.ProjectY</groupId>
  <artifactId>ProjectY</artifactId>
  </exclusion>
 </exclusions>
</dependency>
</dependencies>
```

26. What are Excluded dependencies in Maven?

- Let say a project A depends on project B, and project B depends on project C. The developers of project A can explicitly exclude project C as a dependency. We can use the "exclusion" element to exclude it.
- Such dependencies are called Excluded dependencies in Maven.

27. What are the default locations for source, test and build directories in Maven?

The default locations are as follows:

Source: src/main/java

Test: src/main/test

Build: Target

28. How will you run test classes in Maven?

We need Surefire plugin to run the test classes in Maven.

To run a single test we can call following command:

```
mvn -Dtest=TestCaseA test
```

We can also use patterns to run multiple test cases:

```
mvn -Dtest=TestCase* test
mvn -Dtest=TestCaseA,TestCaseB,TestImportant* test
```

29. Can we create our own directory structure for a project in Maven?

Yes, Maven gives us the flexibility of creating our own directory structure. We just need to configure the elements like , etc. in the section of pom.xml.

30. What are the differences between Gradle and Maven?

- Gradle is nowadays getting more popular. Google uses it for Android development and release. Companies like LinkedIn also use Gradle.
- Gradle is based on Domain Specific Language (DSL). Maven is based on XML.
- Gradle gives more flexibility to do custom tasks similar to ANT. Maven scripts have predefined structure. So it is less flexible.
- Maven is mainly used for Java based systems. Gradle is used for a variety of languages. It is a Polyglot build tool.

31.List out what are the Maven's order of inheritance?

The maven's order of inheritance is

- Parent Pom
- Project Pom
- Settings
- CLI parameters (Running the goals)