Maven Notes:

Apache Maven is a build automation tool. Maven addresses two aspects of building software. First, it describes how a software is built and, second, it describes its dependencies. An XML file describes the software project being built, its dependencies on other external modules and components, the build order, directories, and required plugins. It comes with predefined targets to perform certain well-de ned tasks, such as code compilation and its packaging. Maven dynamically downloads Java libraries and Maven plugins from one or more repositories, such as the **Maven Central Repository**, and stores them locally.

Download Maven from http://maven.apache.org/

The list of directories contained in Maven folder and their brief description

- The *bin* folder contains the batch files and shell scripts to run maven on various platforms
- The *boot* folder contains the jars required for maven to start
- The *conf* folder contains the default settings.xml file used by Maven
- The *lib* folder contains the libraries used by Maven. It also contains an *ext* folder in which third party extension, which can extend or override the default Maven implementation can be placed.

Creating simple Maven project using command prompt

Run the following command:

```
mvn archetype:generate -DgroupId=com.banking.bank -DartifactId=simple-project -DarchetypeArtifactId=maven-archetype-quickstart -DinteractiveMode=false
```

The above command invoke the generate goal of the archetype plugin
The default Maven installation has minimal features. All features of Maven are available as plugins. When given a plugin name, Maven knows where to download if from and then run it. These plugins are place in local repository, which is a location in your system. Once downloaded, these are never downloaded again unless deleted.

Maven Repositories

There are three types of Maven repositories

- Local: This is the repository in your computer
- Remote: This is the repository where the required Maven files get downloaded
- Mirrors: These are the repository managers, such as nexus and Artifactory. That mirror various repositories.

  Maven stores the downloaded repositories in the following location of your computer
  C:\Users\username\.m2\repository

Maven Lifecycle:

Maven is implemented around the concept of build life cycle. This means there is a clearly defined process to build and distribute artifacts with Maven. The stages of lifecycle are called as phases. In each phase one or more goals can be executed.

Maven has three built in build lifecycles

- *default:* This lifecycle handles project build and deployment
- *clean:* This lifecycle cleans up the files and folders produced by Maven
- *site:* This lifecycle handles the creation of project documentation

The *clean* lifecycle phases:

clean phase removes all the files and folders created by Maven as part of the build

The *site* lifecycle phases:

site phase generates the project documentation, which can be published, as well as a template that can be customized further.

The *default* lifecycle phases:

validate phase validates that all project information is available and correct

compile phase compiles source code

test phase runs tests within a suitable framework

package phase packages the compiled code in the distribution format

verify phase runs checks to verify that package is validate

install phase installs the package in the local repository

deploy phase installs the final package in the configured repository

Each phase is made up of plugin goals. A plugin goal is a specific task that builds the project.

Here is the table of phases, plugins and goals:

Phase	**Plugin**	**Goals**
clean	Maven Clean plugin	clean
site	Maven Site plugin	site
compile	Maven Compiler plugin	compile
test	Maven Surefire plugin	test
package	Varies based on the packaging; for instance, the Maven JAR plugin	jar (in the case of a Maven JAR plugin)
Install	Maven install plugin	install
Deploy	Maven deploy plugin	deploy

Work with Maven surefire plugin:-

The surefire plugin is used during the test phase of the build lifecycle to execute the unit test of an application. The surefire plugin has only one goal "*surefire:test*".

```
<plugins>
    [...]
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-surefire-plugin</artifactId>
        <version>2.19.1</version>
        <configuration>
          <suiteXmlFiles>
            <suiteXmlFile>testng.xml</suiteXmlFile>
          </suiteXmlFiles>
        </configuration>
```

```
      </plugin>
    [...]
</plugins>
```

To add a particular suite.xml file which is in package level

```
<suiteXmlFiles>
    <file>src/test/resources/testng1.xml</file>
 </suiteXmlFiles>
```

To run multiple suite in parallel (TestNG 6.9.8 and JRE 1.7 only allows)

```
<configuration>
  <suiteXmlFiles>
    <file>src/test/resources/testng1.xml</file>
    <file>src/test/resources/testng2.xml</file>
  </suiteXmlFiles>
  <properties>
    <property>
      <name>suitethreadpoolsize</name>
      <value>2</value>
    </property>
  </properties>
</configuration>
```

To add the jar files which are not available in central repository as a dependency

- Add the dependency to the pom.xml

```
<dependency>
        <groupId>org.apache.tomcat</groupId>
        <artifactId>apache-tomcat</artifactId>
        <version>8.0.14</version>
        <type>tar.gz</type>
</dependency>
```

- Now run the following command from the project folder in command prompt

  mvn install:install-file -DgroupId=org.apache.tomcat -DartifcatId=apache-tomcat -Dversion=8.0.14 -Dtype=tar.gz -Dfile=location of the jar file

Work offline mode

Go to the project folder in command prompt and run the following command

```
mvn dependeny:go-offline
```

After successful message run the following command

```
mvn -o clean package
```

Another way to run the maven in offline mode is to specify the offline parameters as true in settings.xml

```
<offline>true</offline>
```