Video No 3: Introduction to Maven and its Lifecycle

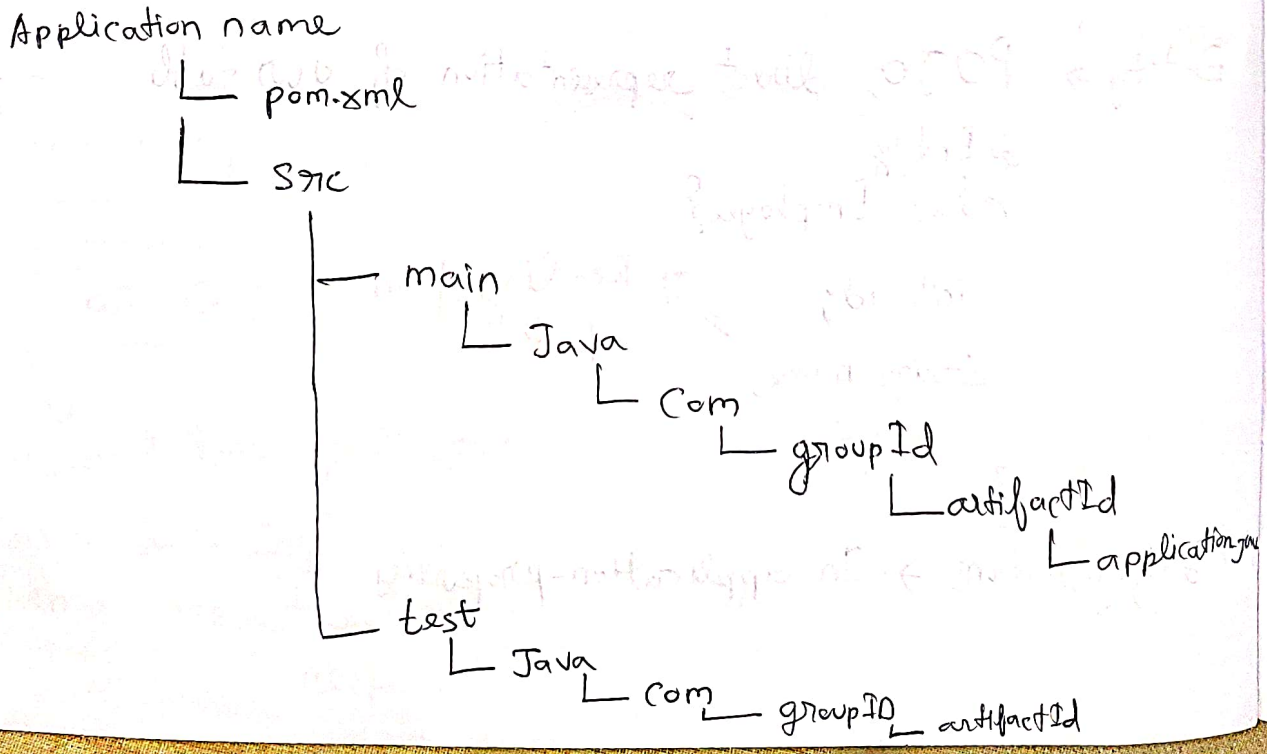
What is Maven ?

* Build generation
* Dependency Resolution
* Documentation.

Maven uses POM → (Project Object Model)

when "maven" command is given, it looks for "pom.xml" in the current directory & get the needed configuration.

(Structure of Pom.xml.)



Now we will decifer the pom.xml file

<?xml version ="1.0" encoding ="UTF-8"?>
< project xmlns ="http://maven ------" xmlns:xsi ="------"
xsi: schemaLocation ="http://------ ">
⇒ Location of the Schema
↳ It specifies the XML schema, also to make sure XML adheres to correct structure.

<parent>
<groupId> org.springframework.boot
<artifactId> spring-boot-starter-parent
<version> 3.2.3
<relativePath/>
</parent>

⇒ Used to define parent Pom.

⇒ Every com parent has a parent Pom.

And every application that you create has a <parent> (spring-boot-starter-parent) ⇒ and this parent has super Pom.xml as the parent Pom.

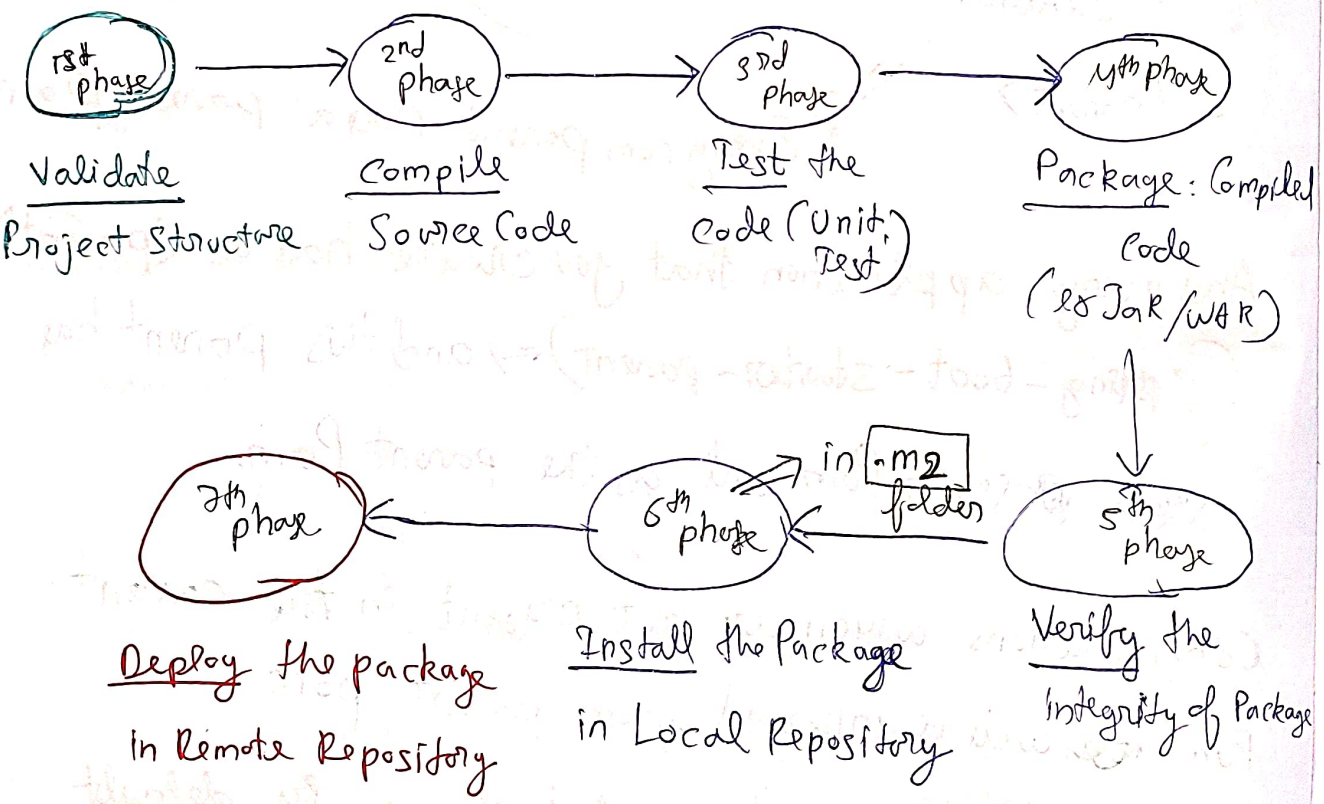
Configurations which are not present in the current Pom will be inherited from parent pom.
If <parent> tag is not specified Maven by default picks the config of superPom.xml

```
<repositories>
    <repository>
        <id> central
        <url> https://repo-maven.org/maven2
    </repository>
</repositories>
```

This is where Maven will look for dependency and download it

if value is not here, it will inherit it from parent pom or SuperPom.xml

Maven Build Lifecycle



** It is sequential. What we mean by that is if we give a command to run "Package" Phase i.e. 4th phase, then all 1st, 2nd and 3rd Phase will get executed first.

Each phase has multiple goals, we can say tasks as well.

- Phase 1
  - Goal 1 (task 1)
  - Goal 2 (task 2)
  - Goal 3 (task 3)
  - Goal 3. (task 4).

If you want to run the Goal 3 of phase 4, then all the goals of previous phases and their tasks will get executed sequentially and then the current task.

Phase 1 → all tasks, Phase 2 → all tasks, Phase 3 → all tasks,
Phase 4 → Goal 1, Goal 2, Goal 3.

Imp: Inside the pom.xml there is a <build> tag.

```
<build>
  <plugin>
    <groupId>
    <artifactId>
  </plugin>
</build>
```

This tag helps us to add a new task inside any of the phase required.

if I want to add a new task to any of the phases.
then I have to add in the <build> → in pom.xml

validate:

mvn validate

```
<build>
  <plugin>
    <groupId> org.maven.apache.plugins </groupId>
    <artifactId> maven-checkstyle-plugin </artifactId>
    <version> 3.1.2 </version>
    <executions>
      <execution> [illegible]
        <id> validate-checkstyle </id>
        <phase> validate </phase>
        <goals>
          <goal> check </goal>
        </goals>
      </execution>
  </plugin>
</build>
```

phase in which you want to add the goal

Actual goal.

This is the new task.

mvn compile:→ java classes → byte code.

.class files ⇒ Put it under target/classes folder for each component

mvn test → Compile and then run test classes in our project.

mvn package → First Validates → compile → test → and then creates a JAR/WAR file out of it

↳ this gets generated in target folder.

ex. In each component we have.

target ↳ promise-cache-0.1.0-SNAPSHOT.jar

mvn verify → Ex. pmd analysis.

Static code analysis.

- Find unused variable
- Find unused imports.
- Empty Catch Block
- No usage of Object
- Find duplicate code etc.

```
<executions>
  <execution>
    <id> pmd-analysis
    <phase> verify
    <goals>
      <goal> pmd.
    <goals>
  </execution>
</executions>
```

mvn install

» You want to install the packaged JAR file into your local repository. ~~.m2~~ /.m2 folder.

mvn package did the JAR creation, how to be used as a dependency you have to ~~use Maven~~ upload in local repository i.e. mvn install → uploading to /.m2 folder.

In settings.xml → the path is provided for local repository. You can change this path as well.

~~Note~~

Whenever we will have to download anything. Maven will give first preference to local repository [/.m2] then it will check remote repository → Because of pom.xml.

mvn deploy → this will deploy to remote repository

pom.xml

```
<project>
  <distributionManagement>
    <repository>
      <id> remote-repository
      <url> https://: remote-repository-url
    </repository>
  </distributionManagement>
</project>
```

(Example walmart central)

settings.xml.

```
<server>
  <id>
  <username>
  <password>
</server>
```