Component of pom.xml continue ....

```
3)  Dependency Mangement
->  in Application dependencies are required to compile, to test and
    run an application

->  in normal approach developer is responsible to download required jar files
    from internet and put in the project

->  in Maven approach developer is not responsible to download required jar files
    and put in the project.

->  Maven has given flexibility to developer to specify required dependencies
    in pom.xml. Maven will search for them in the repositary,
    download and put in the project.

->  If we need any dependency in Maven based application then we have to specify
    in pom.xml file like
```

root tag

```
<dependencies>
    <dependency>
        <groupId>org.seleniumhq.selenium</groupId>
        <artifactId>selenium-java</artifactId>
        <version>4.1.4</version>
    </dependency>

    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.8.2</version>
        <scope>test</scope>
    </dependency>
</dependencies>
```

If we specify dependency like this
then Maven searches for those dependency
by creating url like below

https://repo1.maven.org/maven2/org.seleniumhq.selenium/
selenium-java/4.1.4/

Note: maven following "Transitive Dependency
Mechanism"
i.e If our dependency requires any other
dependency then maven will download automatically

Dependency Scope

There are 6 scope available for dependency in Maven
i) compile
ii) provided
iii) Runtime
iv) test
v) system
vi) import

1) compile

-> compile is default scope. i.e If we are not specific scope then bydefault it is compile

-> compile scope means this dependency is available in all phases like compile, test and run

2) provided

-> If dependency scope is provided then that dependency is available upto compile and test but not for runtime

-> At run time either JVM or container will provide required dependency

Example : In web application , servlet api is required to compile and test but servlet api is provided by container at run time that's why servlet api dependency scope is provided

```
<dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>javax.servlet-api</artifactId>
    <version>4.0.1</version>
    <scope>provided</scope>
</dependency>
```

3) runtime :

If scope of dependency is runtime then that dependency is available at run time but not for compile time

4) test

If scope of dependency is test then that dependency is available for compile, test and run

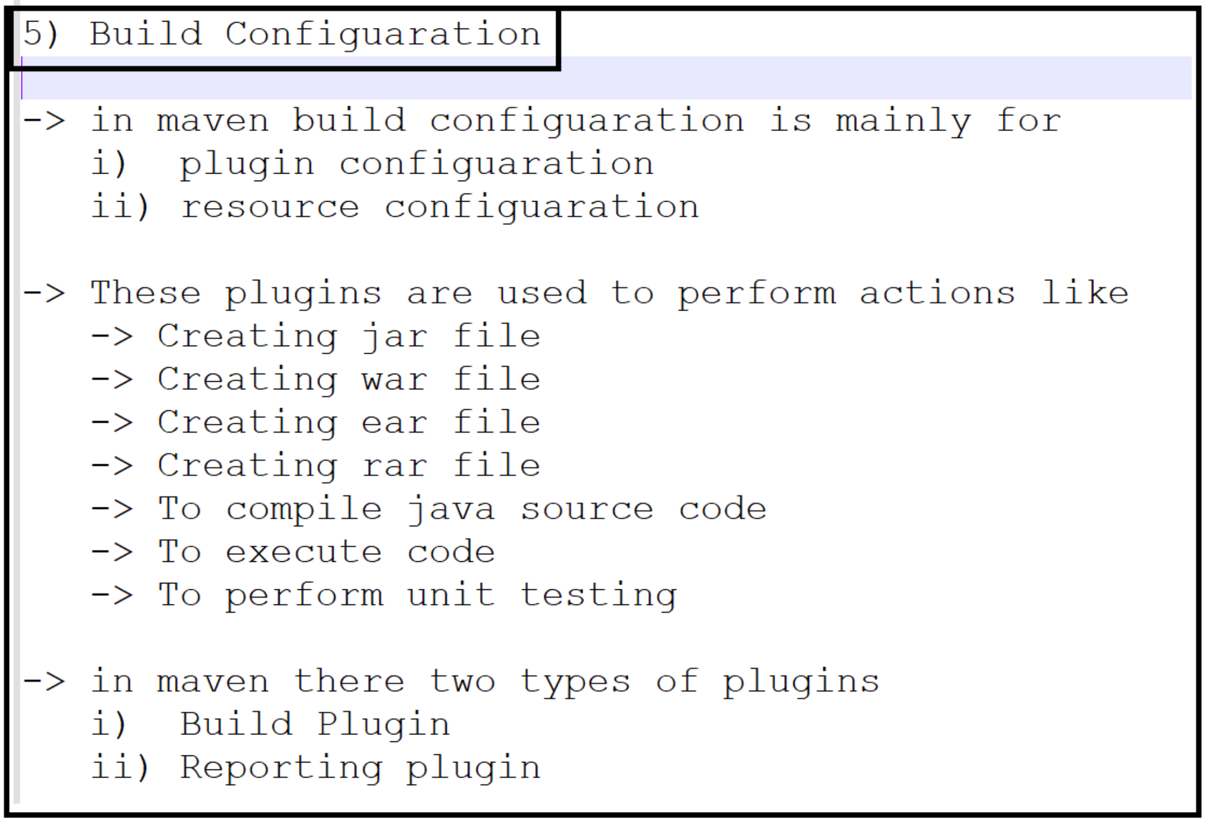
This scope is trasitive scope

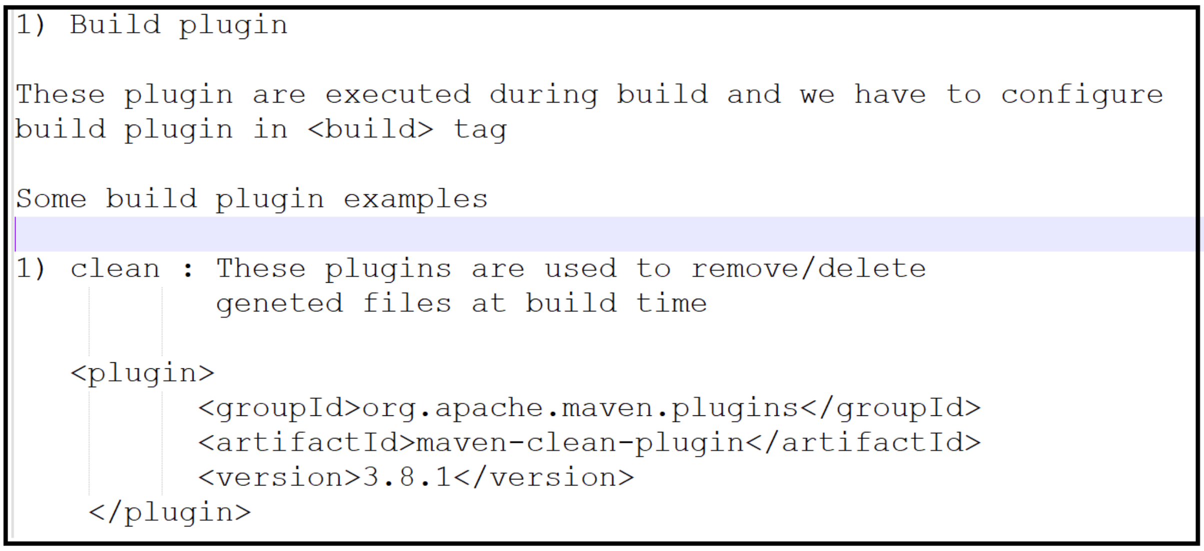
Example :

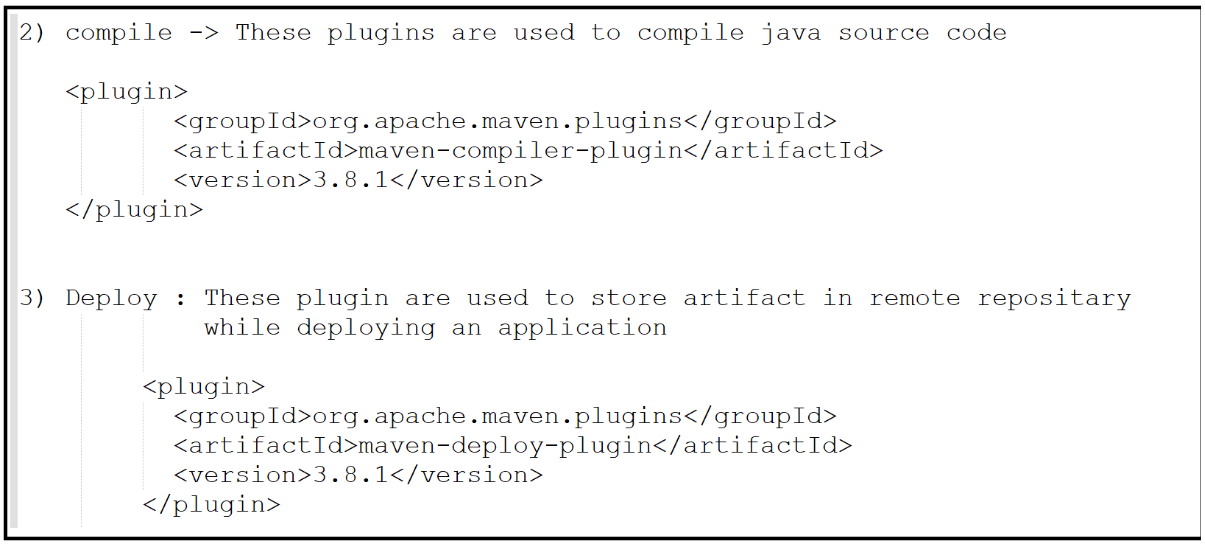
```
    <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-api</artifactId>
            <version>5.8.2</version>
            <scope>test</scope>
        </dependency>
```

```
5) system
-> system scope is same as provided. The only difference is
   they are not downloaded from remote repositary. They are refered
   from system and for that we need to configure system path

   <dependency>
   <groupid>x.y.z</groupid>
   <artifactid>demo-project</artifactid>
   <version>1.0.0</version>
   <scope>system</scope>
   <systemPath>apps/app.war/WEB-INF/lib/demo.jar</systemPath>
   </dependency>

6) import
-> This scope is available in Maven 2.0.9 and later

-> import scope is only supported on a dependency of type pom
   in the dependencyManagment section.
```

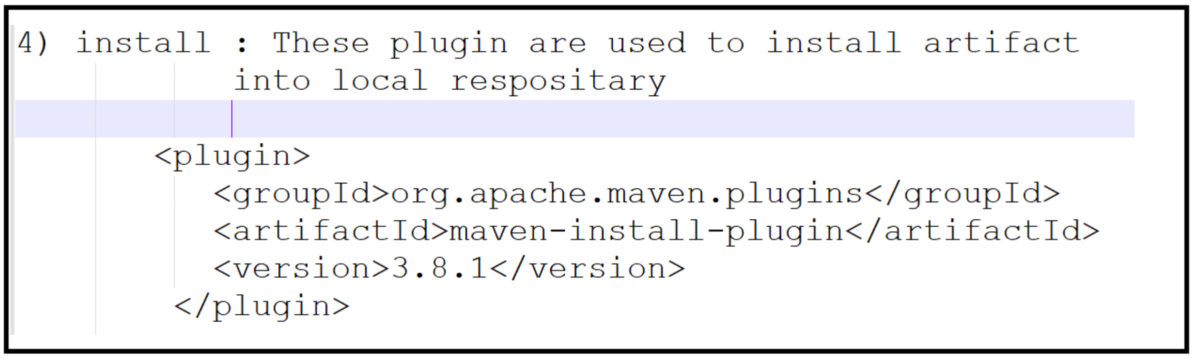
```
4) Project Inheritance

-> in maven based application it is possible to inherit
   configuaration from one pom file to another pom file

-> When we want to inherit configuaration from one pom to another pom file
   then parent pom file packaging tag value must be pom like below

   <packaging>pom</packaging>

-> If we want to inherit parent pom configuaration details in child pom
   then in child pom file we need to use <parent> tag
   like below

   <parent>
     <groupId>com.jipl</groupId>
     <artifactId>JIPL-CLM-APP</artifactId>
     <version>0.0.1-SNAPSHOT</version>
   </parent>
```


```
5) Build Configuaration

-> in maven build configuaration is mainly for
   i)  plugin configuaration
   ii) resource configuaration

-> These plugins are used to perform actions like
   -> Creating jar file
   -> Creating war file
   -> Creating ear file
   -> Creating rar file
   -> To compile java source code
   -> To execute code
   -> To perform unit testing

-> in maven there two types of plugins
   i)  Build Plugin
   ii) Reporting plugin
```

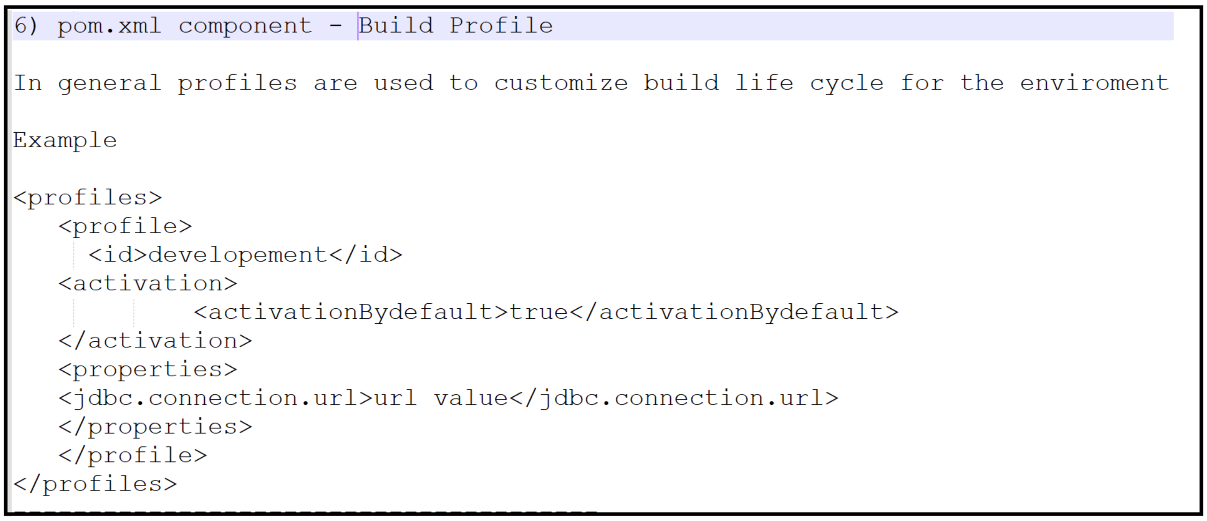

```
1) Build plugin

These plugin are executed during build and we have to configure
build plugin in <build> tag

Some build plugin examples

1) clean : These plugins are used to remove/delete
           geneted files at build time

   <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-clean-plugin</artifactId>
           <version>3.8.1</version>
    </plugin>
```



```
2) compile -> These plugins are used to compile java source code

   <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-compiler-plugin</artifactId>
          <version>3.8.1</version>
   </plugin>


3) Deploy : These plugin are used to store artifact in remote repositary
            while deploying an application

        <plugin>
          <groupId>org.apache.maven.plugins</groupId>
          <artifactId>maven-deploy-plugin</artifactId>
          <version>3.8.1</version>
        </plugin>
```



```
4) install : These plugin are used to install artifact
             into local respositary

        <plugin>
           <groupId>org.apache.maven.plugins</groupId>
           <artifactId>maven-install-plugin</artifactId>
           <version>3.8.1</version>
         </plugin>
```



```
6) pom.xml component - Build Profile

In general profiles are used to customize build life cycle for the enviroment

Example

<profiles>
   <profile>
     <id>developement</id>
   <activation>
            <activationBydefault>true</activationBydefault>
   </activation>
   <properties>
   <jdbc.connection.url>url value</jdbc.connection.url>
   </properties>
   </profile>
</profiles>
```

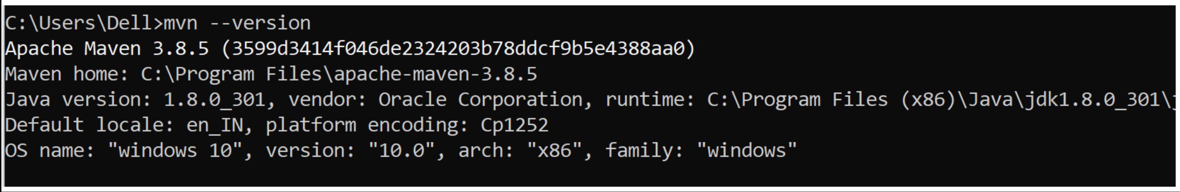
```
Install MAVEN Software:

Installation Process:

1. download apache-maven-3.5.4.zip file from
   internet[https://maven.apache.org/download.cgi]

2. Unzip apache-maven-3.5.4.zip file under C drive and we will get apache-maven-3.5.4 folder.

3. Set the following Environment Variables in System.
```

```
JAVA_HOME: C:\Program Files (x86)\Java\jdk1.8.0_301
```

```
path

1) jdk bin directory path as below

   C:\Program Files (x86)\Java\jdk1.8.0_301\bin

2) maven bin directory path as below
   C:\Program Files\apache-maven-3.8.5\bin
```

```
3) M2_HOME :C:\Program Files\apache-maven-3.8.5

4) MAVEN_HOME :C:\Program Files\apache-maven-3.8.5
```

```
4. Test MAVEN Installation:
 Open Command prompt and use the following command.
 C:\apache-maven-3.5.4\bin>mvn --version
```



```
C:\Users\Dell>mvn --version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: C:\Program Files\apache-maven-3.8.5
Java version: 1.8.0_301, vendor: Oracle Corporation, runtime: C:\Program Files (x86)\Java\jdk1.8.0_301\
Default locale: en_IN, platform encoding: Cp1252
OS name: "windows 10", version: "10.0", arch: "x86", family: "windows"
```

when above command executed without any error means maven configuration has done successfully