

**Default (or Build) Lifecycle**

* **validate** - validate the project is correct and all necessary information is available
* **compile** - compile the source code of the project
* **test** - test the compiled source code using a suitable unit testing framework. These tests should not require the code be packaged or deployed
* **package** - take the compiled code and package it in its distributable format, such as a JAR.
* **verify** - run any checks on results of integration tests to ensure quality criteria are met
* **install** - install the package into the local repository, for use as a dependency in other projects locally
* **deploy** - done in the build environment, copies the final package to the remote repository for sharing with other developers and projects.

#### Build Life Cycle of clean phase

1. mvn pre-clean
2. mvn clean
3. mvn post-clean

#### Site Lifecycle (site)

* pre-site
* site
* post-site
* site-deploy
* The command used in maven to generate javadocs for a given project is '*mvn site*'. Basically, when this command is invoked, maven calls '*Doxia*' document generation and other report generating plugins.
* **Doxia** is basically a framework used for content generation by maven. This generates contents both in static and dynamic ways.

**Q1. What is Maven ?**  
Ans. Maven is a build automation tool used primarily for Java projects.  
  
**Q2. Difference between Maven and Ant ?**  
Ans. Ant is procedural, we need to provide information about what to do and when to do through code.   
  
Maven is declarative, everything is defined in the pom file.  
 **Q3.  What is the best practice configuration usage for files - pom.xml or settings.xml ?**  
Ans. The best practice guideline between settings.xml and pom.xml is that configurations in settings.xml must be specific to the current user and that pom.xml configurations are specific to the project.  
 **Q4.  How can I change the default location of the generated jar when I command "mvn package"?**  
Ans. By default, the location of the generated jar is in ${project.build.directory} or in your target directory. We can change this by configuring the outputDirectory of maven-jar-plugin.  
 **Q5.  What is Maven's order of inheritance?**  
Ans.   
  
1. parent pom  
2. project pom  
3. settings  
4. CLI parameters  
 **Q6.  What is a Mojo?**  
Ans. A mojo is a 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.  
 **Q7.  How do I determine which POM contains missing transitive dependency?**  
Ans. run "mvn -X"  
  
**Q8. How would you see the version of Maven ?**  
  
Ans. mvn --version  
  
**Q9. Where do we configure repositories in Maven ?**  
  
Ans. Within settings.xml in either MAVEN\_HOME or .M2 directory.  
  
**Q10. What are different type of repositories in Maven ?**  
  
Ans. Local and Remote.   
  
**Q11. How does Maven looks for a dependency or resource ?**   
  
Ans. It refers to the settings.xml to look for the repositories to look for the resource. First It looks into the configured local repository, then it looks into the configured Remote repositories. If the resource is still not found , it looks it within maven repository central i.e repo1.maven.org. If its still not found, it throws the exception saying "Unable to find resource in repository central".  
  
**Q12. What is maven repository central ?**  
  
Ans. Its the repository provided by Maven. In case your POM specify the dependencies and its not available in the configured local and the remote repository. It then looks for the resource in Maven Central. Maven provides most of the generic dependency resources at this remote location.   
  
**Q13. What would you do if you have to add a jar to the project using Maven ?**  
  
Ans. If it’s already there in Maven local repository, we can add that as a dependency in the project pom file with its Group Id, Artifact Id and version.  
  
We can provide additional attribute SystemPath if its unable to locate the jar in the local repository.  
  
If it’s not there in the local repository, we can install it first in the local repository and then can add it as dependency.

**Q14. Have you ever had problem getting your projects in eclipse refreshed after you made changes in the POM files?**

Ans. Yes, It happens many times but I would usually perform mvn eclipse:eclipse and this would resolve the project refresh problems.

**Q15. What is the difference between compile and install?**

Ans. Compile compiles the source code of the project

whereas  
  
Install installs the package into the local repository, for use as a dependency in other projects locally  
  
**Q16.  How can we see Dependencies for the project and where exactly they are defined?**  
  
Ans. Using    
  
mvn dependency:tree

**Q17.  What is a transitive dependency? Can we override Transitive Dependency version and If Yes, how?**

Ans. Transitive dependency is the dependencies not defined directly in the current POM but the POM of the dependent projects.  Transitive dependencies allow to avoid specifying the libraries that are required by the project which are specified in other dependent projects - Remote or Local.

Yes we can override transitive dependency version by specifying the dependency in the current POM.

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

suppose we have two projects, "mytest" and "mytest2".

The "mytest" project has a dependency on the commons-lang jar file

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>commons-lang</groupId>

<artifactId>commons-lang</artifactId>

<version>2.3</version>

<scope>compile</scope>

</dependency>

</dependencies>

### [pom.xml of mytest2](http://www.avajava.com/tutorials/maven/how-do-i-add-a-transitive-dependency-to-my-projects-classpath/mytest2-pom.xml)

<dependencies>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>com.maventest</groupId>

<artifactId>mytest</artifactId>

<version>1.0-SNAPSHOT</version>

<scope>compile</scope>

</dependency>

</dependencies>

**Q18.  What is a cyclic dependency?**

Ans. A has dependency of B, B has dependency of C and C has dependency of A.

With Maven 2 , came transitive dependency wherein in above scenario, C will acts as a dependency of A as if this dependency has been defined directly in A but the negative side is that if it leads to cyclic dependency , it creates problems.

**Q19. What are the benefits of transitive depency in Maven ?**  
Ans. Transitive dependencies allows to avoid specifying the libraries that are required by the project which are specified in other dependent projects - Remote or Local.  
  
**Q20. What is the difference between Maven, Ant and Jenkins ?**  
Ans. Maven and Ant are Build Technologies whereas Jenkins is a continuous integration tool.  
  
**Q21. Have you heard of Ban Duplicate Classes Maven enforcer plugin ? What is its use ?**  
Ans. Yes , we have been using this plugin with our projects and its **purpose is to warn and stop the Build if there are duplicates of the same package and class are being carried either directly or through transitive dependencies**. the duplicate could be coming through different types of dependencies or through different versions of the same dependency. Its purpose is to make sure that there is only one copy that’s being used at compile time and runtime and hence shouldn’t later result in runtime problems.  
  
**Q22. How to tackle duplicate classes in maven build?**  
Ans. The simplest way is to ignore them if Maven enforcer plugin is complaining about it but it may lead to runtime problems later.  
  
We can do the dependency:tree to see from where these duplicate ones are coming and hence can exclude the duplicate one.  
  
**Q23. What are different dependency scopes in Maven ?**  
Ans. Compile  
  
This is the default scope. Compile dependencies are available in all classpaths of a project. Moreover, these dependencies are propagated to dependent projects.  
  
Provided  
  
This is much like compile, but indicates you expect the JDK or a container to provide the dependency at runtime.   
  
Runtime  
  
This scope indicates that the dependency is not required for compilation, but is for execution.   
  
Test  
  
This scope indicates that the dependency is only available for the test compilation and execution phases. This scope is not transitive.  
  
System  
  
This scope is similar to provide except that you have to provide the JAR which contains it explicitly.  
  
**Q24. How do you resolve Maven Dependencies issue while upgrading dependencies versions?**  
  
Ans. Step 1 - Upgrade the required dependency , perform build and check build errors  
  
Step 2.1 - If the error is of missing transitive dependency ( which is rare and means that the previous version has a dependency which is missing in the later version ), I would look for the dependency in google and hence will include it as direct dependency in Pom file.  
  
Step 2.2. If the error is for Duplicate dependencies, and the choice is between transitive and direct dependency, I usually remove the direct dependency.  
  
Step 2.3 - If the error is for Duplicate dependencies , and both are transitive dependencies. I first make a choice ( usually later version ) and then ignore the previous version dependency.  
  
Step 2.4 - If the error is for Duplicate dependencies , and there are more than 2 duplicates, I usually ignore it by specifying within maven-enforcer-plugin config.  
  
Step 3 - Perform a Clean Build.  
  
Step 4 - Check Maven Dependency Tree to make sure that Duplicates have been removed or dependency is there in case of missing dependency.  
  
Step 5 - Perform tests and make sure that there are no runtime problems.  
  
Step 6 - If there are runtime problems ( which very likely occurs if you have different version dependencies , very likely by doing 2.4 , you will have to remove step 2.4 and alternately perform 2.2 or 2.3 )  
  
**Q25. What is a Test Dependency Scope in Maven?**  
Ans. This scope indicates that the dependency is not required for normal use of the application, and is only available for the test compilation and execution phases. This scope is not transitive.  
  
**Q26. What is the POM packaging in Maven?**  
Ans. POM packaging is simply a specification that states the primary artifact is not a war or jar, but the pom.xml itself.  
  
**Q27. Difference between Maven Project and Module?**  
Ans. Maven Module has a Parent whereas Project doesn’t. when we add the parent section to the pom file, it adds the module section to the parent project pom file. When we execute mvn compile / install, it basically checks that module section of the parent to identify all the modules that needs to be compiled first.  
  
**Q28. How to specify the sequence in which sub modules needs to be built?**  
Ans. By specifying the modules in the same sequence in the parent POM.  
  
  
**Multiple Choice Questions**  
 **Q29.  What is Maven's order of inheritance?**  
  
 a. parent pom -> project pom -> settings -> CLI parameters  
 b. project pom -> parent pom -> settings -> CLI parameters  
 c. settings -> parent pom -> project pom -> CLI parameters  
 d. CLI parameters -> settings -> parent pom -> project pom  
  
Ans. parent pom -> project pom -> settings -> CLI parameters  
  
**Q30.  How do I determine which POM contains missing transitive dependency?**  
  
 a. mvn -A  
 b. mvn -M  
 c. mvn -R  
 d. mvn -X  
  
Ans. mvn -X  
  
**Q31.  What is the sequence in which Maven looks for the resources?**  
  
 a. Remote-> Maven Central - > Local  
 b. Local -> Remote - Maven Central  
 c. Remote-> Local -> Maven Central  
 d. Maven Central -> Local -> Remote  
  
Ans. Local -> Remote - Maven Central  
  
**Q32.  POM stands for ...**   
  
 a. Project Object Model  
 b. Project Oriented Model  
 c. Prospective Object Model  
 d. Prospective Objective Model  
  
Ans. Project Object Model  
  
**Q33.  Which of the following is not type of Maven Repository?**  
  
 a. Local  
 b. Remote  
 c. Maven Central  
 d. Maven Local  
  
Ans. Maven Local  
  
**Q34.  What is the default location of local repository?**  
  
 a. ~/.m2/repo  
 b. ~/m2./home/repository  
 c. ~/m2./repository  
 d. ~/m2./home/repo  
  
Ans. ~/m2./repository  
  
**Q35.  Which of the following is not type of Maven Plugin?**  
  
 a. Build  
 b. Reporting  
 c. Remote  
 d. All are valid Maven Plugin types  
  
Ans. Remote  
  
**Q36.  Which of the following is not a dependency scope in Maven?**  
  
 a. Compile  
 b. Test  
 c. System  
 d. Export  
  
Ans. Export  
  
**Q37.  What is a project's fully qualified artifact name?**  
  
 a. <groupId>:<artifactId>:<version>  
 b. <groupId>:<artifactId>  
 c. <artifactId>:<groupId>:<version>  
 d. <artifactId>:<version>  
  
Ans. <groupId>:<artifactId>:<version>  
  
**Q38. Which of the following is dependency exclusion?**  
  
a. A doesn't depend on C and then A marks C as excluded.  
b. A Depends on B and B depends on C and then A can mark B as excluded.  
c. A and B depends on C and then they can mark C as excluded.  
d. A depends upon B and B depends upon C then A marks C as excluded.  
  
Ans. A depends upon B and B depends upon C then A marks C as excluded.