**1. What is Maven primarily used for?**

a) Database Management  
b) Project Management and Build Automation  
c) Web Design  
d) Data Analysis

**Answer:** b) Project Management and Build Automation

**2. What file is used to configure a Maven project?**

a) build.gradle  
b) pom.xml  
c) config.json  
d) maven.yml

**Answer:** b) pom.xml

**3. Which command is used to compile a Maven project?**

a) mvn clean install  
b) mvn compile  
c) mvn build  
d) mvn test

**Answer:** b) mvn compile

**4. What is the default local repository location for Maven?**

a) /usr/local/maven/repository  
b) C:/maven\_repo  
c) ~/.m2/repository  
d) C:/Program Files/Maven/repository

**Answer:** c) ~/.m2/repository

**5. What does the clean phase in Maven do?**

a) Deletes the pom.xml file  
b) Removes temporary files from the /target directory  
c) Installs dependencies  
d) Tests the project

**Answer:** b) Removes temporary files from the /target directory

**6. What is the purpose of a Maven plugin?**

a) To provide a GUI for Maven  
b) To extend Maven functionality  
c) To manage database connections  
d) To replace the pom.xml file

**Answer:** b) To extend Maven functionality

**7. Which lifecycle phase generates the JAR file for a Maven project?**

a) compile  
b) package  
c) install  
d) deploy

**Answer:** b) package

**8. What command is used to create a new Maven project?**

a) mvn project:create  
b) mvn archetype:generate  
c) mvn new:project  
d) mvn start

**Answer:** b) mvn archetype:generate

**9. What is the purpose of the dependencyManagement section in a pom.xml file?**

a) To list all required dependencies explicitly  
b) To manage transitive dependencies  
c) To provide a central location for defining dependency versions  
d) To download dependencies automatically

**Answer:** c) To provide a central location for defining dependency versions

**10. Which Maven scope is used for dependencies required only during testing?**

a) provided  
b) compile  
c) test  
d) runtime

**Answer:** c) test

**1. Which phase of the Maven build lifecycle is not executed by default when running mvn install?**

a) validate  
b) test  
c) deploy  
d) package

**Answer:** c) deploy  
**Explanation:** The deploy phase is executed only when explicitly called or during a deployment process, not during mvn install.

**2. What happens if a dependency version is not specified in the pom.xml file?**

a) Maven throws an error and stops execution.  
b) Maven downloads the latest available version.  
c) Maven uses the version specified in dependencyManagement, if available.  
d) Maven skips that dependency.

**Answer:** c) Maven uses the version specified in dependencyManagement, if available.  
**Explanation:** If no version is specified, Maven looks to the dependencyManagement section for the version. If no version is provided anywhere, it results in an error.

**3. If a parent POM specifies a plugin version, what happens when the child POM does not specify the plugin version?**

a) The child uses the default plugin version from Maven.  
b) The build fails with an error.  
c) The version defined in the parent POM is used.  
d) The plugin is ignored in the child project.

**Answer:** c) The version defined in the parent POM is used.  
**Explanation:** Maven allows child POMs to inherit plugin versions defined in the parent POM.

**4. What does the <scope> tag with provided indicate for a dependency?**

a) The dependency will be included in the JAR but not used at runtime.  
b) The dependency will be used for testing only.  
c) The dependency is provided by the runtime environment (e.g., servlet container).  
d) The dependency will not be downloaded automatically.

**Answer:** c) The dependency is provided by the runtime environment (e.g., servlet container).  
**Explanation:** The provided scope means the dependency is needed for compilation but is expected to be available in the runtime environment (e.g., a servlet API in a web server).

**5. What will happen if two dependencies with the same groupId and artifactId but different versions are declared in pom.xml?**

a) Maven downloads both versions and merges them.  
b) Maven uses the latest version specified in the pom.xml.  
c) Maven uses the first version declared in the pom.xml.  
d) Maven throws a build error.

**Answer:** b) Maven uses the latest version specified in the pom.xml.  
**Explanation:** Maven resolves dependency conflicts by using the version that appears last in the pom.xml or dependency tree, unless overridden by dependency management.

**6. In which situation is the optional dependency flag used?**

a) To ensure the dependency is downloaded only when explicitly requested.  
b) To exclude the dependency from transitive dependencies.  
c) To mark a dependency as required for testing only.  
d) To ensure the dependency is downloaded regardless of scope.

**Answer:** b) To exclude the dependency from transitive dependencies.  
**Explanation:** The optional flag marks a dependency so that it won't be included as a transitive dependency for other projects.

**7. What is the difference between <dependencyManagement> and <dependencies> in Maven?**

a) <dependencyManagement> is used to manage dependency scopes.  
b) <dependencyManagement> does not directly include dependencies in the build.  
c) <dependencies> is only used for transitive dependencies.  
d) There is no difference; they are interchangeable.

**Answer:** b) <dependencyManagement> does not directly include dependencies in the build.  
**Explanation:** Dependencies listed in <dependencyManagement> must be explicitly declared in child modules to be included, whereas dependencies in <dependencies> are directly included.

**8. Which repository is queried first when resolving a dependency in Maven?**

a) Central Repository  
b) Remote Repository  
c) Local Repository  
d) Snapshot Repository

**Answer:** c) Local Repository  
**Explanation:** Maven first checks the local repository (~/.m2/repository) before querying remote repositories.

**9. What happens when a dependency is declared with the system scope?**

a) Maven downloads it from the central repository.  
b) Maven searches for it in the system classpath.  
c) The dependency must be manually added and its path explicitly defined.  
d) Maven excludes it from the final build.

**Answer:** c) The dependency must be manually added and its path explicitly defined.  
**Explanation:** The system scope requires specifying the physical path of the dependency, which is discouraged due to poor portability.

**10. What will happen if a transitive dependency has a conflicting version with a directly declared dependency?**

a) Maven throws a build error.  
b) Maven uses the transitive dependency version.  
c) Maven uses the version declared directly in the project.  
d) Maven skips the conflicting dependency.

**Answer:** c) Maven uses the version declared directly in the project.  
**Explanation:** Direct dependencies take precedence over transitive dependencies when resolving conflicts.

**1. What happens if a dependencyManagement section in a parent POM specifies a dependency version, but the child POM overrides the version?**

a) The version in the parent POM is used.  
b) Maven uses the overridden version from the child POM.  
c) Both versions are downloaded and used simultaneously.  
d) Maven throws a build error due to the conflict.

**Answer:** b) Maven uses the overridden version from the child POM.  
**Explanation:** The dependencyManagement section only defines default versions. If a child POM explicitly specifies a version, it takes precedence.

**2. What is the purpose of the <exclusions> tag in a dependency declaration?**

a) To exclude the dependency from the build process entirely.  
b) To prevent specific transitive dependencies from being included.  
c) To exclude the dependency from testing only.  
d) To exclude dependencies from the dependencyManagement section.

**Answer:** b) To prevent specific transitive dependencies from being included.  
**Explanation:** The <exclusions> tag is used to exclude specific transitive dependencies brought in by another dependency.

**3. Which of the following is true about Maven profiles?**

a) Profiles are used to override dependencies in the pom.xml.  
b) Profiles can only be activated by specifying a specific Maven property.  
c) Profiles can be used to define different build configurations.  
d) Profiles are only supported in the settings.xml file.

**Answer:** c) Profiles can be used to define different build configurations.  
**Explanation:** Maven profiles allow you to customize build configurations for different environments, such as production or development. They can be activated by properties, JDK versions, or operating systems.

**4. What will happen if two transitive dependencies have the same groupId and artifactId but different versions?**

a) Maven uses the version closest to the root in the dependency tree.  
b) Maven includes both versions in the classpath.  
c) Maven uses the version farthest from the root in the dependency tree.  
d) Maven throws an error and stops the build.

**Answer:** a) Maven uses the version closest to the root in the dependency tree.  
**Explanation:** Maven resolves conflicts using a nearest-wins strategy in the dependency tree.

**5. What does the <repositories> tag in the pom.xml file do?**

a) Specifies local repository configurations.  
b) Defines remote repositories where Maven should search for dependencies.  
c) Specifies a backup repository for failed downloads.  
d) Overrides the central Maven repository.

**Answer:** b) Defines remote repositories where Maven should search for dependencies.  
**Explanation:** The <repositories> tag allows you to specify additional remote repositories for dependency resolution.

**6. What is the effect of using the pluginManagement section in the POM file?**

a) Plugins are directly added to the build.  
b) Plugins defined here must be used explicitly in child modules.  
c) Plugins are automatically inherited by child modules.  
d) Plugins defined here override all other plugin configurations.

**Answer:** b) Plugins defined here must be used explicitly in child modules.  
**Explanation:** The pluginManagement section defines plugin versions and configurations that must be explicitly declared in the child POM to be included.

**7. How does Maven determine the order of execution for lifecycle phases?**

a) Based on alphabetical order.  
b) Based on the order they are declared in the pom.xml.  
c) Based on a predefined order defined by Maven’s lifecycle.  
d) Based on the dependencies between lifecycle phases.

**Answer:** c) Based on a predefined order defined by Maven’s lifecycle.  
**Explanation:** Maven follows a well-defined lifecycle order (e.g., validate → compile → test → package, etc.).

**8. What does the <optional>true</optional> tag in a dependency declaration mean?**

a) The dependency will not be included in the build.  
b) The dependency will be included but not in transitive dependencies.  
c) The dependency is downloaded only for optional builds.  
d) The dependency is used only in testing phases.

**Answer:** b) The dependency will be included but not in transitive dependencies.  
**Explanation:** An optional dependency will not propagate to projects that depend on the current project.

**9. What happens if a dependency is declared twice with different scopes in the same POM?**

a) Maven uses the dependency with the narrower scope.  
b) Maven uses the dependency with the broader scope.  
c) Maven throws a build error.  
d) Maven includes both dependencies in the classpath.

**Answer:** b) Maven uses the dependency with the broader scope.a  
**Explanation:** Maven resolves scope conflicts by using the broader scope (e.g., compile > test > provided).

**10. What does Maven's "inheritance mechanism" allow for in multi-module projects?**

a) Sharing the target directory between modules.  
b) Automatically executing all child modules when the parent is built.  
c) Sharing configurations, dependencies, and plugins between parent and child POMs.  
d) Combining all child modules into a single executable file.

**Answer:** c) Sharing configurations, dependencies, and plugins between parent and child POMs.  
**Explanation:** Maven’s inheritance mechanism allows child POMs to inherit configurations, dependencies, and plugins from the parent POM.

