1. When should one use Maven?

The Maven Build Tool can be used in the following conditions:

- When the project has a large number of dependencies. Then, using Maven, you can easily manage those dependencies.
- When the version of a dependency changes frequently. To update dependencies, simply update the version ID in the pom file.
- Maven makes it simple to handle continuous builds, integration, and testing.
- When you need a quick way to generate documentation from source code, this is the tool you use. It helps in compiling source code, and then packaging it into JAR or ZIP files.

2. Discuss the core concepts of Maven.

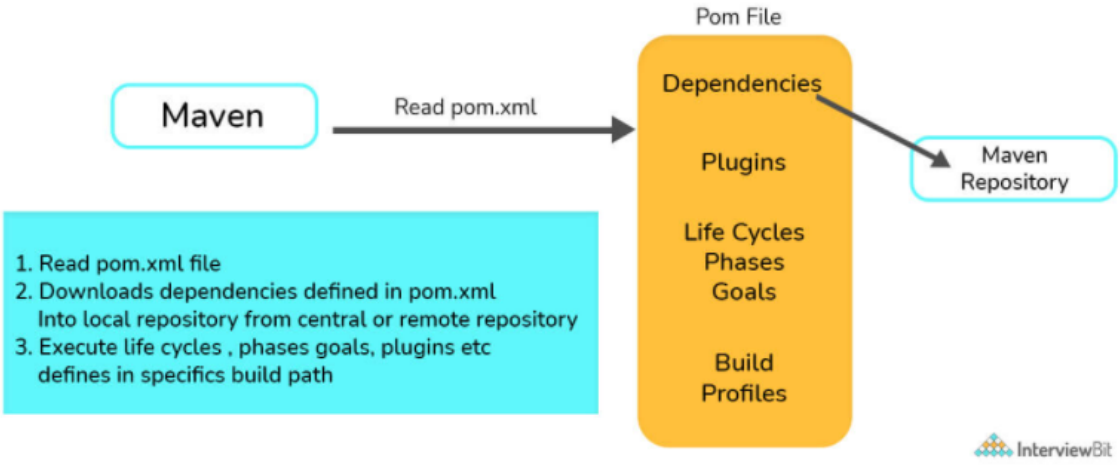
The core concepts of Maven are:

- POM Files: Project Object Model (POM) files are XML files that include information about the project and configuration information used by Maven to construct the project, such as dependencies, source directory, plugin, goals, and so on. When you want to run a maven command, you provide it with a POM file to run. To complete its configuration and functions, Maven reads the pom.xml file.
- Dependencies and Repositories: Repositories are folders containing bundled JAR files, and dependencies are external Java libraries necessary for Project. The local repository is simply a folder on your computer's hard drive. Maven retrieves dependencies from a central Maven repository and places them in your local repository if they aren't found in the local Maven repository.
- Build Life Cycles, Phases, and Goals: A build life cycle is made up of a series of build phases, each of which contains a set of goals. A build lifecycle, phase, or goal is referred to as a Maven command. When a lifecycle is asked to be run using the maven command, all of the build steps in that life cycle are likewise

run. When a build phase is requested to be executed, it is followed by all build phases in the given sequence.

- Build Profiles: Build Profiles are a set of configuration parameters that allow you to build your project using a variety of setups. For example, you might need to develop and test your project on your local computer. You can add different build profiles to your POM files using its profile elements to enable different builds, which can be triggered in a variety of ways.
- Build Plugins: Build Plugins are used to accomplish a certain task. A plugin can be added to the POM file. Maven comes with various pre-installed plugins, but you can also write your own in Java.

3. How does Maven work?



5. What elements are used for creating a pom.xml file?

The following elements are necessary for creating a pom.xml file:

- project- The root element of the pom.xml file is the project.
- modelVersion- It identifies which version of the POM model you're working with. For Maven 2 and Maven 3, use version 4.0.0.
- groupId- groupId is the project group's identifier. It is unique, and you will most likely use a group ID that is similar to the project's root Java package name.
- artifactId- It is used for naming the project you're working on.

- version- The version number of the project is contained in the version element. If your project has been released in multiple versions, it is helpful to list the versions.

Other Pom.xml File Elements

- dependencies- This element is used to establish a project's dependency list.
- dependency- dependency is used inside the dependencies tag to define a dependency. The groupId, artifactId, and version of each dependency are listed.
- name- This element is used to give our Maven project a name.
- scope- This element is used to specify the scope of this maven project, which can include compile, runtime, test, among other things.
- packaging- The packaging element is used to package our project into a JAR, WAR, and other output formats

6. What are the different types of Maven repositories? Discuss.

The three types of repositories of Maven are:

- Local repository
- Central repository
- Remote repository

Maven scans these repositories for dependencies. Maven looks in the Local repository first, then the Central repository, and finally the Remote repository if the Remote repository is defined in the POM.

- Local Repository: Local repository is a directory on the developer's device. The local repository contains all of Maven's dependencies. Even though several projects rely on dependencies, Maven only needs to download them once.
- Central Repository: The Maven community has built the central Maven repository. Maven searches this central repository for any dependencies that aren't available

in your local repository. The dependencies are subsequently downloaded into your local repository by Maven.

- Remote Repository: Maven may download dependencies from a remote repository hosted on a web server. It is frequently used to host internal organization projects. The dependencies are subsequently downloaded into your local repository by Maven.

7. What command should one use to install JAR files in the Local Repository?

- JAR files are installed in the local repository using mvn install.
- The following plugin is used to manually install the JAR into the local Maven repository: install-file-Dfile = <file path>

8. In Maven, what do you mean by Clean, Default, and Site?

The three built-in build life cycles are:

- Clean: The clean lifecycle is in charge of project cleaning.
- Default: The project deployment is handled by the default lifecycle.
- Site: The creation of the project's site documentation is referred to as the site lifecycle.

9. What are the different phases of the default life cycle?

The different phases of the default lifecycle are:

- Validate: Make sure the project is correct and that you have all of the necessary information.
- Test: Test the compiled source code using an appropriate unit testing framework. These tests should not demand that the code be packed or deployed; instead, take the compiled code and package it in a manner that can be distributed, such as a JAR.
- Compile: Compile the project's source code.

- Verify: Perform any necessary checks on integration test findings to ensure that quality criteria are met.
- Install: Adds the package to the local repository, allowing it to be used as a dependency in other projects.
- Deploy: Copies the entire package to the remote repository for sharing with other developers and organizations, and is done in the build environment.

10. What are Maven plugins used for? What are the types of Maven plugins?

Maven Plugins are used for:

- Creating JAR files.
- Creating WAR files.
- Compiling the source code files.
- Unit testing of the code.
- Creating the project documentation.
- Creating project reports.

Maven plugins are divided into two categories:

- Build plugins: These plugins are used throughout the build process and are configured in the pom.xml file's <build/> element.
- Reporting plugins: These plugins are configured in the pom.xml's <reporting/> element and run during stage generation

12. What is Maven's inheritance order?

In Maven, the order of inheritance is:

- Settings
- CLI parameters
- Parent POM
- Project POM

13. In Maven, what is a snapshot?

A snapshot is a specific version of a project that shows the most recent development copy of the project being worked on. Maven always checks out a SNAPSHOT of the project in the remote repository for each build.

As a result, anytime Maven discovers a newer SNAPSHOT of the project, it downloads and replaces the project's older .jar file in the local repository.

14. What are the locations where Maven dependencies are stored?

Maven saves all of the JARs, dependency files, and other things it downloads in the Maven local repository. All of the artifacts are kept locally in the Maven local repository, which is a folder on the local machine.

15. What are the different types of Maven build profiles? In what ways can build profiles of maven be activated?

The different types of Maven build profiles are:

- Per-User: This is defined in the Maven settings.xml file in user’s home dir.
- Per Project: This is defined in the project’s pom.xml.
- Global: This is defined in the global Maven settings.xml file.

Maven build profiles can be activated or triggered in the following ways:

- Using explicit commands
- Maven settings
- On the basis of environment variables
- Configuration of the operating system
- Present/missing files

16. How would you refer to a property declared in your pom.xml file?

In order to refer to a property declared in your pom.xml, the property name makes use of the names of the XML components that designate the value, with "pom" being accepted as a synonym for the project (root) element.

So ${pom.name} is the project's name, ${pom.version} is the project's version, ${pom.build.finalName} is the final name of the file generated when the built project is packaged, and so on.

17. How to generate javadocs in Maven?

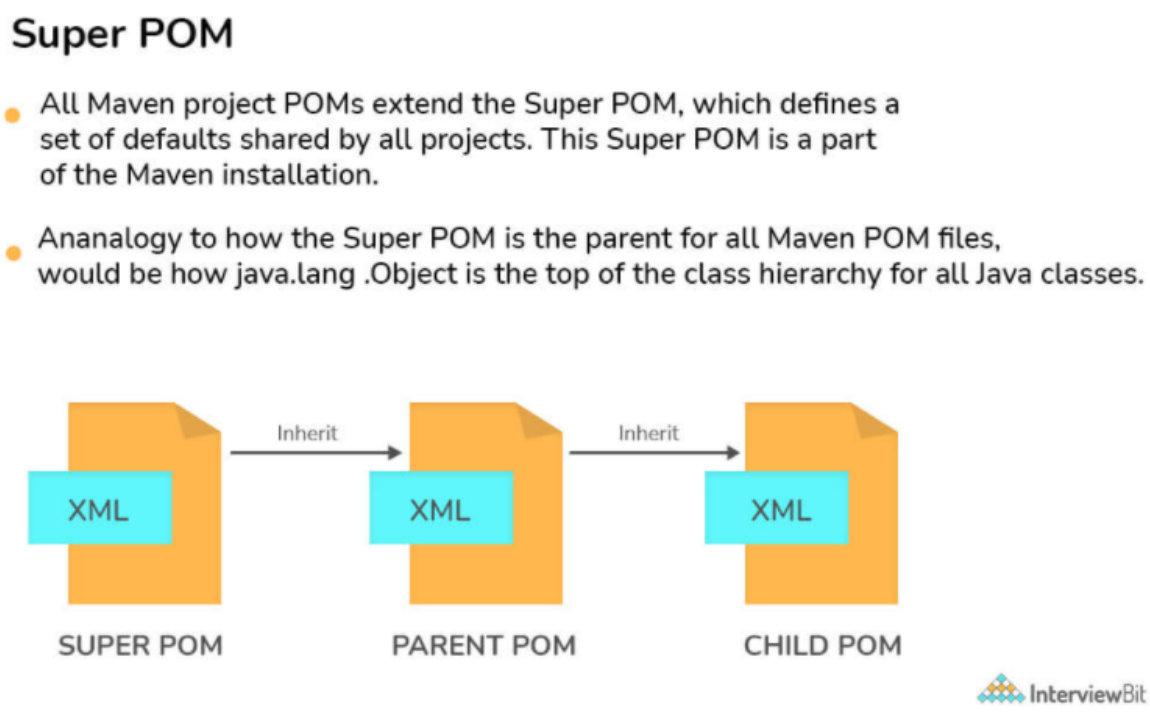
The maven-javadoc plugin is used by Maven to generate a project's javadocs. To create javadocs, this plugin internally uses the JDK\bin\javadoce.exe command. The javadocs for the project are generated when the project is deployed with the mvn install command.

18. What exactly is MOJO?

Every Maven plain Old Java Object (MOJO) is an executable goal, and a plugin pertains to the distribution of these MOJOs. MOJO allows Maven to add functionalities that it doesn't already have. In Maven, a MOJO is a single unit of the task.

19. What do you understand about the term ‘Super POM’?

Any POM file has the ability to point to its parent POM. There is a system-wide POM file that is automatically considered as the parent POM file if the parent POM element is absent. The super POM is the name given to this POM file. Finally, the super POM is used to extend all of the application POM files. The apex of the POM hierarchy is the super POM file. The super POM file contains all of the default configurations. All of the configurations defined in the super POM file will be inherited by even the simplest version of a POM file. You can alter any option you want by redefining the same section in your application POM file.



20. What is a 'Dependency Scope'? What are the different types of Dependency Scopes?

The dependency scope pertains to all dependencies related to the present stage of the build.

The following are the several sorts of dependence scopes:

- Compile- It's the default scope, and it shows which dependencies are available in the project's classpath.
- Provided- It denotes that the dependency is delivered at runtime by the JDK, web server, or container.
- Runtime- This indicates that the dependency is not required during compilation but is necessary during execution.
- Test- It claims that dependencies are only available during the test compilation and execution phases.
- System- It implies that you must specify the system path.

- Import- This means that the dependencies in that POM's section should be used instead of the identified or specified POM.

21. What do you mean by a Maven Archetype? How will you create a new project based on an Archetype?

Maven Archetype is a Maven plugin that makes it possible to create a project structure based on a template. These archetypes are essentially project templates that Maven generates when you create a new project. Archetype is a Maven project templating toolkit, in a nutshell.

After getting to the directory where the project is located, type the command: – mvn archetype: generate in the command prompt. This aids in creating a new project based on an archetype.

There are four steps for creating a project from an archetype:

- prepare a repository reference
- the choice of an archetype,
- that archetype's configuration,
- the efficient creation of the project using the data gathered

In most cases, an archetype is procured from a remote repository. You're ready to go if that repository can be reached using your Maven configuration. You must add the repository to your settings.xml if the repository is not managed and you wish to refer to it directly.

22. What command is used to create a new project from a hard drive?

The -mvn archetype: create is used to start a new project.

After reading the source and resource files, as well as the values of its parameters and other properties, the archetype is constructed.

23. What are the phases of the clean lifecycle?

The Maven clean lifecycle takes care of eliminating all temporary files from the output directory, including generated source files, compiled classes, and previous JAR files, among other things.

- pre-clean- performs tasks that are necessary prior to actual project cleaning.
- clean- delete all files created by the previous build.
- post-clean- performs tasks that are necessary to finalize project cleaning.

24. What are the phases of the site lifecycle?

Everything related to generating documentation for your project is handled by the Maven site lifecycle.

- pre-site- performs tasks that are necessary prior to actual project site generation.
- site- develop the project's site generation.
- post-site- performs tasks that are necessary to finalize project site generation, also prepares for site deployment.
- site-deploy- deploy the developed site documentation to the web server of your choice.

25. Explain the three commonly used plugins: clean, surefire, antrun.

Maven clean is a plugin that, as the name implies, attempts to clean the files and directories generated by Maven during the build process. The target folder, which contains all of the class files, documentation, and JAR files, is removed by the plugin.

The Surefire Plugin is used to run an application's unit tests during the test phase of the build lifecycle. It can generate reports in one of two file formats: plain text files or XML files.

The Antrun Plugin allows you to perform Ant tasks directly from within Maven. Your Ant scripts can even be embedded in the POM!

26. What is the settings.xml file in Maven?

A Maven installation is configured using the settings.xml file. It's comparable to a pom.xml file, but it's either global or user-specific. The Maven settings.xml file provides elements that define the values required to configure Maven's execution in several ways. These values include the location of the local repository, authentication information, and alternate remote repository servers among others.

Let's look at the elements in the settings.xml file that we can change. The settings.xml file's main element, settings, can include up to nine predefined child elements:

```
<settings xmlns = "http://maven.apache.org/SETTINGS/1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:schemaLocation="http://maven.apache.org/SETTINGS/1.0.0
https://maven.apache.org/xsd/settings-1.0.0.xsd">
   <localRepository/>
   <interactiveMode/>
   <offline/>
   <pluginGroups/>
   <servers/>
   <mirrors/>
   <proxies/>
   <profiles/>
   <activeProfiles/>
</settings>
```

The following configurations are included:

- Proxy configuration
- Local repository configuration
- Remote repository configuration
- Central repository configuration

27. What is dependency mediation and dependency management?

When multiple versions of an artifact are encountered, Maven determines which version of the dependency should be used. The earliest declared dependence will be used if two dependency versions are at the same depth in the dependency tree. This is referred to as "dependency mediation."

Dependency management allows project authors to declare the versions of artifacts that are to be utilized when they are discovered in transitive dependencies or dependencies that have no version specified.

28. What do you mean by the term “system dependency”?

The term "system dependency" refers to the scope system's dependency. These dependencies are typically used to let Maven know the dependencies the JDK or VM provides. System dependencies are typically used to resolve dependencies on JDK-provided artifacts. Some common examples are the Java Authentication and Authorization Service (JAAS) or the JDBC standard extensions.

29. What is the use of an Optional Dependency?

When splitting a project into submodules isn't practicable (for some reason), optional dependencies are employed. The concept is that some of the dependencies are just required for particular project features and will not be required if those features are not used. Such a feature should ideally be divided into a sub-module that is dependent on the project's main functionality. Only non-optional dependencies would be included in this new subproject, as you'd need them all if you wanted to use the subproject's features.

If a user wants to use functionality associated with an optional dependency, they must redeclare it in their own project. Optional dependencies save storage and memory. They prevent troublesome jars from being packed into a WAR, EAR, fat jar, or other formats if they violate a license agreement or cause classpath difficulties.

30. What do you understand about ‘Transitive Dependency’ in Maven? What is dependency exclusion?

By incorporating transitive dependencies automatically, Maven eliminates the need to discover and define libraries that the dependencies require. According to transitive dependency, if X is dependent on Y and Y is dependent on Z, then X is dependent on both Y and Z.

The "exclusion" element can be used to exclude any transitive dependency. If X is reliant on Y and Y is reliant on Z, then X can declare Z as excluded.

31. What are the elements that must be defined for each external dependency?

The Maven software relies heavily on external dependencies. It is an intrinsic component of the system without which it is impossible to find dependencies in a system. We'll need the following information to specify the external dependency:

- It necessitates a group ID that is identical to the library name.
- It necessitates an artifact ID that is identical to the library name.
- The system's dependency scope must be mentioned.
- The system path that corresponds to the project position must be mentioned.

32. What are user-defined properties?

You have the opportunity to define your own arbitrary properties in addition to the implicit properties. A POM or a Profile can be used to define properties. The properties defined in a POM or a Maven Profile can be referenced in Maven just like any other property. User-defined properties can be used to filter resources via the Maven Resource plugin, or they can be referenced in a POM. In a Maven POM, here's an example of defining some arbitrary properties.

```
<project>
```

```
    ...
    <properties>
        <arbitrary.property.x>Text</arbitrary.property.x>
        <hibernate.version>3.2.1.ga</hibernate.version>
    </properties>
    ...
    <dependencies>
        <dependency>
            <groupId>org.hibernate</groupId>
            <artifactId>hibernate</artifactId>
            <version>${hibernate.version}</version>
        </dependency>
    </dependencies>
    ...
</project>
```

arbitrary.property.x and hibernate.version are two properties defined in the preceding example. In a dependency declaration, hibernate.version is mentioned. It's usual practice in Maven POMs and Profiles to use the period character as a separator in property names. The following example demonstrates how to define a property in a Maven POM profile.

```
<project>
    ...
    <profiles>
        <profile>
            <id>random-profile</id>
            <properties>
                <arbitrary.property>Text</arbitrary.property>
            </properties>
        </profile>
    </profiles>
    ...
</project>
```

33. Discuss the profile element in settings.xml file.

The settings.xml profile element is a trimmed version of the pom.xml profile element. It is made up of the elements: activation, repositories, pluginRepositories, and properties.

These four components are the only ones included in the profile elements since they deal with the build system as a whole (which is what the settings.xml file is for), not individual project object model settings.

If a profile is activated from settings, its values will override any POM or profiles.xml profiles with the same ID.

- Activation: The strength of a profile, like that of the POM's profiles, comes from its capacity to modify specific values only under certain conditions, which are stated via an activation element.
- Repositories: Repositories are remote collections of projects that Maven utilizes to populate the build system's local repository.
- pluginRepositories: Maven plugins are a unique form of artifact in themselves. Plugin repositories may be segregated from other repositories as a result of this. Each of the pluginRepository components specifies a remote source where Maven can look for new plugins.
- Properties: Maven properties, like Ant properties, value placeholders. The notation ${X}, where X is the property, can be used to obtain their values anywhere within a POM.

34. What is maven-release plugin and how does it work?

The maven release plugin is used to automate the build and release process. When maven executes the maven-release-plugin, the following activities are performed:

- mvn release:clean - clears the workspace from the previous build and prepares it for a new one.
- mvn release:rollback - If the previous process failed, it rollbacks the workspace.
- mvn release:prepare - It performs the following tasks:
    - Checks the local workspace for any uncommitted files.
    - Checks for SNAPSHOT dependencies and verifies they aren't present.

    - Prepares the final version for release.
    - Updates the pom to SCM (SVN/Git/Mercurial/CVS).
    - Runs the test cases.
    - Executes the ultimate commit to the SCM.
    - Tags the script/code.
    - Increases the version number and includes the SNAPSHOT as part of the subsequent releases.
- mvn release:perform - fetches the code from the repository and executes the maven goal to develop and deploy the artifacts.

35. Why are exclusions made on a dependency-by-dependency basis instead of at the POM level?

This is primarily to ensure that the dependency graph is predictable, as well as to prevent inheritance effects from eliminating a dependent that should not be excluded. If you have to use the method of last resort and add an exclusion, make sure you know which of your dependencies is causing the undesirable transitive dependency.

The banned dependencies rule can be specified to fail the build if a troublesome dependency is identified, regardless of path. You'll need to add specific exclusions to each path the enforcer detects if the build fails

36. Explain the default and the advanced configuration inheritance.

The default behavior includes merging the content of the configuration element according to the element name. If a certain element exists in the child POM, that value becomes the effective value. The parent value becomes the effective value if the child POM does not have an element but the parent does. It's important to note that this is solely an XML operation, with no code or plugin settings involved. Only the elements are involved, not their values.

Advanced configuration inheritance includes adding attributes to the children of the configuration element to regulate how child POMs inherit configuration from parent POMs. Combine.children and combine.self are the two attributes. These attributes can be used in a child POM to regulate how Maven integrates the parent's plugin configuration with the child's explicit configuration.

37. Explain Project Aggregation.

Project Aggregation specifies the modules from the parent POM instead of specifying the parent POM from the module. As a result, the parent project is aware of its modules, and if a Maven command is issued against the parent project, the Maven command is also applied to the parent's modules. For Project Aggregation, you must accomplish the following:

- Change the packaging of the parent POMs to "pom."
- Specify the modules' directories in the parent POM (children POMs).

38. What is the use of the Maven Wagon plugin?

The Maven Wagon Plugin, as its name suggests, allows you to access numerous Maven Wagon functionalities. To transfer resources to and from Maven repositories, Maven Wagon offers a layer of abstraction over the core transport protocols. Maven Wagon's unified API includes implementations for seven transports.

The following picture depicts the architecture of the Maven Wagon:

39. How is Doxia used by Maven?

Doxia is a content creation framework that aims to give powerful approaches for creating static and dynamic content to its users: Doxia can be used to create static web pages in a web-based publication context, as well as in dynamic content creation systems such as blogs, wikis, and content management systems.

Maven makes substantial use of Doxia, which powers the project's complete documentation system. It enables Maven to take any Doxia-supported document and output it in any format.

For instance, 'mvn site' is the command used by Maven to produce javadocs for a specific project. Maven calls Doxia document generation and other report generating plugins when this command is run.

40. How will you run JUnit tests in parallel with a Maven build?

It is now possible to run tests in parallel without utilizing TestNG in junit 4.7. It's been feasible since 4.6, but 4.7 will include a number of improvements that will make it a realistic alternative. You may also use spring to execute parallel tests.

You can also use this maven plugin:

```
<build>
 <plugins>
   <plugin>
     <groupId>org.apache.maven.plugins</groupId>
     <artifactId>maven-surefire-plugin</artifactId>
     <version>2.6.0</version>
     <configuration>
       <parallel>classes</parallel>
       <threadCount>4</threadCount>
     </configuration>
   </plugin>
 </plugins>
</build>
```

41. How can you skip running the tests for a particular project?

Set the skipTests attribute to true to skip the tests for a certain project.

```
<groupId>org.apache.maven.plugins</groupId>
```

```
<artifactId>maven-surefire-plugin</artifactId>
<version>2.13.0</version>
<configuration>
          <skipTests>true</skipTests>
</configuration>
```

You may also skip the tests by using the following command from the command line:

```
mvn install -DskipTests
```

42. What is the difference between the maven package and the maven install?

package: converts the compiled code into a distributable format, such as a JAR.

install: adds the package to the local repository, allowing it to be used as a dependency in other projects.