1. Explain what is Maven? How does it work?

Maven is a popular open source build tool for enterprise Java projects, designed to take much of the hard work out of the build process. Maven uses a declarative approach, where the project structure and contents are described, rather than the task-based approach used in Ant or in traditional make files, for example. This helps enforce company-wide development standards and reduces the time needed to write and maintain build scripts. The declarative, lifecycle-based approach used by Maven 1 is, for many, a radical departure from more traditional build techniques, and Maven 2 goes even further in this regard.

Maven’s primary goal is to allow a developer to comprehend the complete state of a development effort in the shortest period. In order to attain this goal, Maven deals with several areas of concern:

* Making the build process easy
* Providing a uniform build system
* Providing quality project information
* Encouraging better development practices

2. Explain what POM and its significance is.

POM stands for Project Object Model. It is fundamental unit of work in Maven. It is an XML file that resides in the base directory of the project as pom.xml. The POM contains information about the project and various configuration detail used by Maven to build the project(s). POM also contains the goals and plugins. While executing a task or goal, Maven looks for the POM in the current directory. It reads the POM, gets the needed configuration information, and then executes the goal.

Some of the configuration that can be specified in the POM are following:-

* project dependencies
* plugins
* goals
* build profiles
* project version
* developers
* mailing list

3. Explain what a Maven artifact is?

A Maven Artifact is a Java class that represents the kind of "name" that gets dereferenced by a repository manager into a repository manager artifact. When used in this sense, an Artifact is just a glorified name made up of such parts as groupId, artifactId, version, scope, classifier and so on.

Maven project probably depends on several Artifacts by way of its <dependency> elements. Maven interacts with a repository manager to resolve those Artifacts into files by instructing the repository manager to send it some repository manager artifacts that correspond to the internal Artifacts.

After resolution, Maven builds the project and produces a Maven artifact. We may choose to "turn this into" a repository manager artifact by, in turn, using whatever tool we like, sending it to the repository manager with enough coordinating information that other people can find it when they ask the repository manager for it.

4. List out the dependency scope in Maven?

Maven dependency scope attribute is used to specify the visibility of a dependency, relative to the different lifecycle phases.

Maven provides six scopes:-

1. Compile Scope

2. Provided Scope

3. Runtime Scope

4. Test Scope

5. System Scope

6. Import Scope

5. List out what are the build phases in Maven?

A Maven phase represents a stage in the Maven build lifecycle. Each phase is responsible for a specific task.

Some of the most important phases in the default build lifecycle are:

* Validate: Check if all information necessary for the build is available.
* Compile: Compile the source code.
* Test-compile: Compile the test source code.
* Test: Run unit tests.
* Package: Package compiled source code into the distributable format (jar, war, …).
* Integration-test: Process and deploy the package if needed to run integration tests.
* Install: Install the package to a local repository.
* Deploy: Copy the package to the remote repository.

6. Mention the three-build lifecycle of Maven?

Maven is based around the central concept of a build lifecycle. What this means is that the process for building and distributing a particular artifact (project) is clearly defined.

For the person building a project, this means that it is only necessary to learn a small set of commands to build any Maven project, and the POM will ensure they get the results they desired.

* There are three built-in build lifecycles: default, clean and site.
* The default lifecycle handles your project deployment.
* The clean lifecycle handles project cleaning.
* The site lifecycle handles the creation of your project's site documentation.

7. List out what are the aspects does Maven Manages?

The aspects managed by Maven are: -

* Builds
* Documentation
* Reporting
* SCMs
* Releases
* Distribution

8. Explain what a Maven Repository is? What are their types?

A Maven Repository is a location, generally on a filesystem (either remote or local), where maven artifacts are stored and managed. Once artifacts have been stored in a maven repository, they are available for retrieval and inclusion in other maven projects. Just like artifacts, repositories can be called by many different names: Artifact Repositories, Package Repositories, Package Managers, Repository Managers, Binary Repositories. Remote Maven Repositories are web servers which provide simple HTTP and HTTPs endpoints which allow GET and PUT requests for publishing and retrieving Maven Artifacts.

The three Maven repository are: -

* Local Repository: - Maven local repository is a folder location on your machine. It gets created when you run any maven command for the first time. Maven local repository keeps your project's all dependencies (library jars, plugin jars etc.). When you run a Maven build, then Maven automatically downloads all the dependency jars into the local repository. It helps to avoid references to dependencies stored on remote machine every time a project is build. Maven local repository by default get created by Maven in %USER\_HOME% directory. To override the default location, mention another path in Maven settings.xml file available at %M2\_HOME%\conf directory. When you run Maven command, Maven will download dependencies to your custom path.
* Central Repository: - Maven central repository is repository provided by Maven community. It contains many commonly used libraries.
* Remote Repository: - Sometimes, Maven does not find a mentioned dependency in central repository as well. It then stops the build process and output error message to console. To prevent such situation, Maven provides concept of Remote Repository, which is developer's own custom repository containing required libraries or other project jars.

9. Explain how you can exclude dependency?

Maven allows you to exclude specific dependencies. Exclusions are set on a specific dependency in your POM and are targeted at a specific groupId and artifactId.

Since Maven resolves dependencies transitively, it is possible for unwanted dependencies to be included in your project's classpath.

For example, a certain older jar may have security issues or be incompatible with the Java version you're using. To address this, Maven allows you to exclude specific dependencies.

Exclusions are set on a specific dependency in your POM and are targeted at a specific groupId and artifactId. When you build your project, that artifact will not be added to your project's classpath by way of the dependency in which the exclusion was declared.

10. For POM what are the minimum required elements?

The minimum requirement for a POM are:

* project root.
* modelVersion - should be set to 4.0. ...
* groupId - the id of the project's group.
* artifactId - the id of the artifact (project)
* version - the version of the artifact under the specified group.