1. **What is Maven?**

Maven is a project management and comprehension tool. Maven provides developers a complete build lifecycle framework. Development team can automate the project's build infrastructure in almost no time as Maven uses a standard directory layout and a default build lifecycle.

1. **What does it mean when you say Maven uses Convention over Configuration?**

“Convention over Configuration” means that developers are not required to deal with the build processes themselves. Developers do not need to know each and every configuration details.

1. **What are the aspects Maven manages?**

Maven provides developers ways to manage

* Dependencies
* Compiling
* Testing
* Building
* Documentation / Site Generation
* Releases
* Distribution

1. **What is POM?**

POM stands for Project Object Model. It is an XML file. It always resides in the base directory of the project as pom.xml. It contains information about the project and various configuration details used by Maven to build the project(s).

1. **What is Maven artifact?**

An artifact is a file, usually a JAR that gets deployed to a Maven repository. A Maven build produces one or more artifacts, such as a compiled JAR and a "sources" JAR. That **sources jar** is a **jar** that contains only the **source** code (the . java files) corresponding to the **compiled** artifact. It is useful to add it as a **source** attachment in your IDE, to publish the **sources**, etc. As it only contains **sources**, not **compiled** classes.

Each artifact has a group ID (usually a reversed domain name, like com.example.foo), an artifact ID (just a name), and a version string. The three together uniquely identify the artifact. A project's dependencies are also specified as artifacts.

1. **What is Maven Build Lifecycle?**

A Build Lifecycle is a well defined set of phases which are defined in an order. Phases are composed of goals. A goal is just a unit of work. A phase in this context represents a stage in life cycle.

1. **What are the build lifecycles of Maven?**

The three build lifecycles are

* Clean: It cleans up artifacts created by prior builds.
* Default (Build): This is used to build the application.
* Site: It generates site documentation for the project.

1. **What does the “mvn clean” command do?**

It cleans the artifacts created by prior builds.

1. **What is a goal in Maven terminology?**

A goal represents a specific task which contributes to the building and managing of a project. It may be bound to zero or more build phases. A goal not bound to any build phase could be executed outside of the build lifecycle by direct invocation.

1. **What would this command do “*mvn clean dependency:copy-dependencies package”* ?**

This command will clean the project, copy the dependencies and package the project (executing all phases up to *package*).

1. **What is Build Profile?**

A Build profile is a set of configuration values which can be used to set or override default values of Maven build. Using a build profile, you can customize builds for different environments such as Production v/s Development environments.

1. **What are the three types of Build Profiles?**

Build profiles:

* Per Project − Defined in the project POM file, pom.xml.
* Per User − Defined in Maven settings xml file (<USER\_HOME>/.m2/settings.xml)
* Global − Defined in Maven global settings xml file (<M2\_HOME>/conf/settings.xml)

1. **How can you activate profiles?**

A Maven Build Profile can be activated by;

* Explicitly using command console input.
* Through maven settings.
* Based on environment variables (User/System variables).
* OS Settings (for example, Windows family).
* Present/missing files.

1. **What is a Maven Repository?**

A repository is a place i.e. directory where all the project jars, library jar, plugins or any other project specific artifacts are stored and can be used by Maven easily.

1. **What are the types of Maven repository?**

Maven repository are of three types: **local, central, remote** →(in some resources central and remote repositories are shown under the same node as remote repositories with central and third party repositories)

1. **What is the local repository?**

Maven local repository is a folder location on your local computer. It gets created when you run any maven command for the first time. Maven local repository keeps your project's dependencies (library jars, plugin jars etc).

1. **What is the default location for your local repository?**

**~/.m2/repository or <USER HOME>/.m2/repository**

1. **What is the command to install a JAR file into your local repository?**

**“mvn install”**

1. **What is the Central Repository?**

It is the repository provided by Maven Community. It contains a large number of commonly used libraries. When Maven does not find any dependency in the local repository, it starts searching in the central repository using the following URL: <http://repo1.maven.org/maven2/>.

1. **What is a Remote Repository?**

Sometimes, Maven does not find a dependency in the local or central repository. Then it stops the build process and outputs an error message to the console. To prevent such situations, Maven provides the concept of Remote Repository which is a developer's or a company’s own custom repository containing required libraries or other project jars.

1. **What is the sequence in which Maven searches for dependency libraries?**

Following is the search pattern −

* Step 1 − Search dependency in local repository, if not found, move to step 2 else if found then do the further processing.
* Step 2 − Search dependency in central repository, if not found and remote repository/repositories is/are mentioned then move to step 4 else if found, then it is downloaded to local repository for future reference.
* Step 3 − If a remote repository has not been mentioned, Maven simply stops the processing and throws error (Unable to find dependency).
* Step 4 − Search dependency in remote repository or repositories, if found then it is downloaded to local repository for future reference otherwise Maven as expected stops processing and throws error (Unable to find dependency).

1. **Why are Maven Plugins used?**

Maven Plugins are used to;

* create jar file.
* create war file.
* compile code files.
* unit testing of code.
* create project documentation.
* create project reports.

1. **What are the types of Maven Plugins?**

Maven provides following two types of Plugins:

* **Build plugins:** They execute during the build and should be configured under the <build> element of pom.xml file.
* **Reporting plugins:** They execute during the site generation and they should be configured in the <reporting> element of the pom.xml file.

1. **When does Maven use the External Dependency concept?**

If the dependency is not provided via local, remote or central repository External Dependency is used.

1. **What are the things you need to define for each external dependency?**

External dependencies (library jar location) can be configured in pom.xml in same way as other dependencies.

* Specify groupId same as name of the library.
* Specify artifactId same as name of the library.
* Specify scope as system.
* Specify system path relative to project location

Ex:

<dependency>

<groupId>ldapjdk</groupId>

<artifactId>ldapjdk</artifactId>

<scope>system</scope>

<version>1.0</version>

<systemPath>${basedir}\src\lib\ldapjdk.jar</systemPath>

</dependency>

1. **What is Archetype?**

Archetype is a Maven plugin whose task is to create a project structure as per its template.

1. **What is the command to create a new project based on an archetype?**

**“mvn archetype:generate”**

1. **What is SNAPSHOT in Maven?**

SNAPSHOT is a special version that indicates a current development copy. Unlike regular versions, Maven checks for the latest SNAPSHOT version in a remote repository for every build.

1. **What is the difference between Snapshot and Version?**

In case of **Version**, if Maven once downloaded the mentioned version say data-service:1.0, it will never try to download a newer 1.0 available in the repository. To download the updated code, the data-service version is to be upgraded to 1.1.

In case of **SNAPSHOT**, Maven will automatically fetch the latest SNAPSHOT (data-service:1.0-SNAPSHOT) everytime app-ui team builds their project.

1. **What is transitive dependency in Maven?**

Transitive dependency means to avoid needing to discover and specify the libraries that your own dependencies require, and including them automatically.

1. **What is dependency scope? Name all the dependency scope.**

Dependency scope includes dependencies according to the current stage of the build. Various Dependency Scopes are:

* compile: This scope indicates that dependency is available in the classpath of the project. It is the default scope.
* provided: This scope indicates that dependency is to be provided by JDK or web-Server/Container at runtime.
* runtime: This scope indicates that dependency is not required for compilation, but is required during execution.
* test: This scope indicates that the dependency is only available for the test compilation and execution phases.
* system: This scope indicates that you have to provide the system path.
* import: This scope is only used when dependency is of type pom. This scope indicates that the specified POM should be replaced with the dependencies in that POM's <dependencyManagement> section.

1. **What are the default values for packaging elements? If there is no packaging element defined? What is the default value for that?**

Some of the valid packaging values are jar, war, ear and pom. If no packaging value has been specified, it will default to **jar**.

1. **What is the packaging type for a project that is purely meta-data?**

pom

33. **What is the use of the execution element in a pom file?**

The <execution> element contains information required for the execution of a plugin.

1. **What is a project's fully qualified artifact name?**

<groupId>:<artifactId>:<version>

1. **If you do not define any information, where does your pom inherits that information from?**

All POMs inherit from a parent (despite explicitly defined or not). This base POM is known as the Super POM, and contains values inherited by default.

1. **Why is the profile used in Maven?**

To give portability to projects ( e.g. windows, linux etc).

1. **Which command is used to build your project offline?**

**“mvn -o package”**

1. **How do you exclude dependency?**

Using the *exclusion* element under the related dependency.

1. **What is a system dependency?**

Dependency with a scope value of “system” is always available and it is not looked up in a repository. They are usually used to tell Maven about dependencies which are provided by a local folder path, JDK or the VM.

1. **What is the use of optional dependency?**

Any transitive dependency can be marked as optional using the "optional" element. As an example, A depends upon B and B depends upon C. Now B marked C as optional. Then A will not use C.

1. **What is dependency exclusion?**

Any transitive dependency can be excluded using the "exclusion" element. As an example, A depends upon B and B depends upon C then A can mark C as excluded.

1. **How can you run the clean plugin automatically during the build?**

You can put the clean plugin inside the execution tag in the pom.xml file.

1. **How to stop the propagation of plugins to child POMs?**

Set <inherited> element to false.

1. **What does the "You cannot have two plugin executions with the same (or missing) elements" message mean?**

It means that you have executed a plugin multiple times with the same <id>. Provide each <execution> with a unique <id> then it would be ok.

1. **What is a Mojo?**

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.

1. **What is the difference between Apache Ant and Maven?**

|  |  |
| --- | --- |
| Ant | Maven |
| Ant doesn’t have formal conventions, so we need to provide information about the project structure in the build.xml file. | Maven has a convention to place source code, compiled code, etc. So we don’t need to provide information about the project structure in the pom.xml file. |
| Ant is procedural, you need to provide information about what to do and when to do through code. You need to provide an order. | Maven is declarative, everything you define in the pom.xml file. |
| There is no life cycle in Ant. | There is a life cycle in Maven. |
| Ant is a toolbox. | Maven is a framework. |
| It is mainly a build tool. | It is mainly a project management tool. |
| The ant scripts are not reusable. | The maven plugins are reusable. |
| It is less preferred than Maven. | It is more preferred than Ant. |