AEM-TASK 1 (Padmesh G)

1. Maven Life Cycle:

- **validate**: Checks if the project is correct.
- **compile**: Compiles the source code.
- **test**: Runs unit tests.
- **package**: Packages the compiled code into JAR/WAR.
- **verify**: Validates integration tests.
- **install**: Installs the package in the local repository.
- **deploy**: Deploys the built package to a remote repository.

2. What is pom.xml File and Why We Use It?

pom.xml (Project Object Model) is the fundamental configuration file in a Maven project. It contains metadata, dependencies, build configurations, plugins, and project structure details. Key benefits of using pom.xml include:

- Managing dependencies centrally.
- Automating the build process.
- Defining project structure and configurations.
- Integrating plugins for various tasks.

3. How Dependencies Work in Maven?

Dependencies in Maven are managed through the <dependencies> section of pom.xml. Maven downloads required dependencies from repositories automatically and resolves transitive dependencies (dependencies of dependencies).

Example:

```
<dependencies>
    <dependency>
        <groupId>org.apache.commons</groupId>
        <artifactId>commons-lang3</artifactId>
        <version>3.12.0</version>
    </dependency>
</dependencies>
```

4. Maven repositories:

- **Local Repository (~/.m2/repository)**: Cached dependencies.
- **Central Repository (Maven Central)**: Default public repository.
- **Remote Repositories**: Custom repositories for enterprise projects.

5. Checking the Maven Repository

You can search for dependencies on Maven Central Repository. Use mvn dependency:tree to view the project's dependency tree.

6. How All Modules Build Using Maven

Maven allows multi-module builds using a parent pom.xml. The parent POM specifies module references:

```
<modules>
    <module>core</module>
    <module>ui.apps</module>
    <module>ui.content</module>
</modules>
```

To build all modules:

```
mvn clean install
```

7. Can We Build a Specific Module?

Yes, we can build a specific module using:

mvn clean install -pl module-name -am

-pl: Specifies the module. -am: Builds dependencies of the module.

8. Role of ui.apps, ui.content, and ui.frontend Folder

- **ui.apps**: Contains the main application configurations, templates, and components.
- **ui.content**: Stores content packages, pages, and assets.
- **ui.frontend**: Includes front-end code (React, JavaScript, CSS) for UI development.

9. Why Are We Using Run Modes?

Run modes allow AEM to adapt configurations based on environments (dev, stage, prod). They help in:

- Defining environment-specific settings.
- Optimizing performance and security.

Example:

- author mode for content authors.
- publish mode for live website access.

10. What is Publish Environment?

The **publish environment** is where the final AEM content is made available to end users. It fetches content from the author environment and serves it to visitors.

11. Why Are We Using Dispatcher?

The **Dispatcher** is AEM's caching and load-balancing tool. It:

- Improves performance by caching pages.
- Protects AEM instances from high traffic.

- Secures AEM by blocking unwanted requests.

12. From Where Can We Access crx/de?

CRX/DE (Content Repository eXtreme Developer Environment) can be accessed at:

http://localhost:4502/crx/de/index.jsp

It allows developers to explore, edit, and manage AEM content and configurations.