Assignment

1)MAVEN LIFECYCLE:

Maven is a build automation and project management tool used primarily for Java projects. It simplifies dependency management, compilation, testing, packaging, and deployment.

Maven has three built-in lifecycles:

1. Clean - Cleans the project (mvn clean).
2. Default (Build) - Handles compilation, testing, packaging, and deployment (mvn install).
3. Site - Generates project documentation (mvn site).

Each lifecycle consists of phases, such as:

- validate → compile → test → package → verify → install → deploy.

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

- pom.xml (Project Object Model) is the core configuration file in Maven.
- It defines project dependencies, build settings, plugins, and properties.
- Used to manage project structure and automate builds.

3)**How dependencies work?**

In Maven, dependencies are external libraries required by a project, defined in the pom.xml file under the <dependencies> section. Each dependency includes a groupId, artifactId, and version. Maven resolves dependencies by first checking the local repository (~/.m2/repository/), then fetching them from remote repositories like Maven Central if not found locally. Dependencies have different scopes, such as compile (default), provided, runtime, test, and system, which determine when and where they are used. Maven also manages transitive dependencies, automatically resolving and downloading dependencies of dependencies. To maintain version consistency, dependencyManagement can be used in multi-module projects to specify versions centrally without including dependencies by default. Unwanted transitive dependencies can be excluded using the <exclusions> tag. Developers can analyze dependencies using commands like mvn dependency:tree to visualize the hierarchy. Maven simplifies dependency management, ensuring efficient and automated handling of external libraries for Java projects.

4)How All Modules Build Using Maven?

- A multi-module Maven project has a parent POM that manages multiple child modules.
- Running mvn install at the parent level builds all modules sequentially**.**

5)Can We Build a Specific Module?

We can build a specific module using the command

mvn install -pl module-name -am

where -pl (projects list) specifies the module, and -am (also-make) builds dependencies.

6)Role of ui.apps, ui.content, and ui.frontend Folders?

- ui.apps: Stores OSGi bundles and templates.
- ui.content: Holds site-specific content (pages, assets).
- ui.frontend: Contains frontend code (JS, CSS, React, etc.).

7)Why Are We Using Run Mode?

- Run modes in AEM control environment-specific configurations (e.g., dev, staging, production).
- Example:

  java -jar aem-author.jar -r author,dev
- Allows different settings without modifying the codebase.

8)What is Publish Environment?

- The publish environment in AEM serves content to end-users.
- It receives content from the author environment via replication.

9)Why Are We Using Dispatcher?

- Dispatcher is an AEM caching and load balancing tool.
- It improves performance by reducing requests to the AEM publish server.
- Filters requests to enhance security.

10)From Where Can We Access crx/de?

- crx/de is the AEM CRXDE Lite interface for managing content.
- Access via:

  http://localhost:4502/crx/de/