AEM Assignment - 01

DATE : 18-03-2025

1) Maven Life Cycle

The Maven life cycle is a sequence of phases that manage the build process of a project. It consists of three main life cycles: `default`, `clean`, and `site`. The `default` life cycle includes key phases like `validate`, `compile`, `test`, `package`, `install`, and `deploy`. Each phase executes a specific task, and Maven automatically runs all preceding phases when a particular phase is invoked. For example, running `mvn install` triggers all phases up to `install`.

2) What is pom.xml file and why we use it?

The `pom.xml` (Project Object Model) file is the core configuration file in Maven. It defines the project's structure, dependencies, build process, and plugins. We use it to manage project metadata (like version, name, and group ID), specify external libraries (dependencies), and configure build settings, ensuring consistency and automation in the build process.

3) How dependencies work?

Dependencies in Maven are external libraries or modules required by a project, defined in the `<dependencies>` section of `pom.xml`. Maven downloads these from repositories (like Maven Central) based on the group ID, artifact ID, and version. It resolves transitive dependencies (dependencies of dependencies) automatically and stores them in the local repository (`.m2` folder) for reuse.

4) Check the Maven repository.

The Maven repository is a storage location for libraries, dependencies, and plugins. There are three types:

1. **Local Repository :** On the user's machine (default: `~/.m2/repository`), storing downloaded artifacts.

2. **Central Repository :** A remote public repository (e.g., Maven Central) for common libraries.

3. **Remote Repository :** Custom repositories defined in `pom.xml`. Maven checks the local repository first, then remote ones if needed.

5) How all modules build using Maven?

In a multi-module Maven project, a parent `pom.xml` defines all modules (sub-projects) under the `<modules>` tag. Running `mvn clean install` on the parent directory builds all modules in the order specified or based on their interdependencies. Each module has its own `pom.xml`, and Maven compiles, tests, and packages them systematically.

6) Can we build a specific module?

Yes, we can build a specific module by navigating to its directory and running a Maven command (e.g., `mvn clean install`), or from the parent directory using the `-pl` flag (e.g., `mvn clean install -pl module-name`). This builds only the specified module and its dependencies.

7) Role of ui.apps, ui.content, and ui.frontend folder?

In Adobe Experience Manager (AEM) projects:

- ui.apps : Contains JCR-based code like components, templates, and configurations deployed to `/apps`.

- ui.content : Holds content-related assets like pages and policies deployed to `/content`.

- ui.frontend : Manages front-end assets (e.g., CSS, JS) compiled via tools like Webpack and synced to `ui.apps`. These folders modularize AEM development.

8) Why we are using run mode?

Run modes in AEM allow environment-specific configurations (e.g., `author`, `publish`, `dev`, `prod`). They enable the system to load different settings or content based on the environment by matching run mode-specific config files (e.g., `config.author` or `config.publish`), ensuring flexibility and separation of concerns.

9) What is publish env?

The publish environment in AEM is a server instance where content is made publicly accessible after approval. It hosts the final, visitor-facing website, unlike the author environment, which is used for content creation and editing. Content is replicated from author to publish via workflows.

10) Why we are using dispatcher?

The dispatcher in AEM acts as a caching and load-balancing layer between the publish instance and end users. It caches static content to improve performance, handles security by filtering requests, and distributes load across multiple publish instances, ensuring scalability and speed.

11) From where can we access the crx/de?

CRX/DE (Content Repository Explorer/Developer) can be accessed in AEM at `http://<host>:<port>/crx/de` (e.g., `http://localhost:4502/crx/de`) on the author instance. It requires admin credentials and is used for managing repository content, nodes, and properties directly.