AEM Assignment - 01

**DATE : 18-03-2025**

**1. Maven Lifecycle**

Maven is a build automation tool used primarily for Java projects. It follows a predefined lifecycle to manage project build processes. The three key lifecycles are:

- Clean : Removes previous build artifacts.

- Default (Build) : Compiles, tests, packages, and installs the project.

- Site : Generates project documentation.

**Maven Build Phases**

1. validate – Checks if the project structure is correct.

2. compile – Compiles the source code.

3. test – Runs unit tests.

4. package – Packages the compiled code into a distributable format (e.g., JAR, WAR).

5. verify – Runs integration tests.

6. install – Installs the package into the local repository.

7. deploy – Deploys the final build to a remote repository.

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

The pom.xml (Project Object Model) file is the configuration file for Maven projects. It defines:

- Project information

- Dependencies

- Plugins

- Build settings

- Profiles

Example pom.xml structure:

xml

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>sample-project</artifactId>

<version>1.0.0</version>

<dependencies>

<!-- Dependencies here -->

</dependencies>

</project>

**3. How Dependencies Work?**

Dependencies define external libraries required for a project. They are managed through pom.xml under the <dependencies> section.

Example:

xml

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.12.0</version>

</dependency>

Maven automatically downloads dependencies from repositories and places them in the target directory.

**4. Checking the Maven Repository**

Maven repositories store libraries and dependencies. The main types are:

- Local Repository (~/.m2/repository)

- Central Repository (https://repo.maven.apache.org/maven2/)

- Remote Repository (Configured in pom.xml for custom dependencies)

**5. How All Modules Build Using Maven**

In a multi-module project, the parent POM manages submodules. The pom.xml includes:

xml

<modules>

<module>ui.apps</module>

<module>ui.content</module>

<module>ui.frontend</module>

</modules>

Running mvn install at the root level builds all modules.

**6. Can We Build a Specific Module?**

Yes, using:

sh

mvn install -pl ui.apps -am

- -pl: Specifies the module to build.

- -am: Builds required dependencies.

**7. Role of ui.apps, ui.content, and ui.frontend**

- ui.apps : Stores AEM OSGi configurations and component code.

- ui.content : Stores content packages and templates.

- ui.frontend : Stores front-end resources like JavaScript, CSS, and React code.

**8. Why We Use Run Mode?**

Run modes configure different environments (e.g., development, production). AEM supports:

- author (Admin environment for content management)

- publish (User-facing site environment)

- dev, stage, prod (Custom modes for environments)

Set run modes in sling.properties or environment variables.

**9. What is Publish Environment?**

The publish environment serves live content to users. It is optimized for:

- Fast content delivery

- User access management

- Integration with the dispatcher

**10. Why We Use Dispatcher?**

AEM Dispatcher is a caching and load-balancing tool used to:

- Improve performance by caching pages.

- Protect AEM from high traffic loads.

- Restrict access based on security rules.

**11. From Where Can We Access crx/de?**

The CRX/DE (Content Repository eXtreme/Developer Edition) is accessible at:

http://localhost:4502/crx/de

It allows:

- Managing AEM JCR nodes.

- Editing repository content.

- Inspecting components and configurations.