1 MAVEN LIFECYCLE

Maven follows a structured lifecycle that defines different phases for building a project. The three main lifecycles in Maven are default, clean, and site.

- **Clean Lifecycle:** This phase cleans the project by removing previously compiled files. The command mvn clean deletes the target directory to ensure a fresh build.
- **Default Lifecycle:** This is the primary lifecycle that compiles, tests, packages, and deploys the project. Common phases include:
    - validate (checks project structure)
    - compile (compiles source code)
    - test (runs unit tests)
    - package (creates a JAR/WAR file)
    - install (adds the package to the local repository)
    - deploy (pushes the package to a remote repository)
- **Site Lifecycle:** Generates project documentation using mvn site.

Each phase is executed in order, meaning running mvn package also runs compile and test automatically.

2 WHAT IS POM.XML FILE AND WHY WE USE IT?

The Project Object Model (POM) file (pom.xml) is the core configuration file in Maven. It defines the project structure, dependencies, plugins, and build settings.

- It contains metadata such as project name, version, and description.
- Developers use pom.xml to manage dependencies, ensuring the correct versions are downloaded.
- It allows automation of the build process with minimal manual intervention.
- It supports multi-module projects where a parent POM manages multiple submodules.

3 HOW DEPENDENCIES WORK IN MAVEN?

Dependencies in Maven are external libraries required for a project. These dependencies are defined inside pom.xml, and Maven fetches them from the Maven repository.

- When a dependency is added, Maven looks for it in the local repository (~/.m2).

- If not found locally, Maven downloads it from Maven Central (https://repo.maven.apache.org/maven2).
- Organizations can set up private repositories like Nexus or Artifactory for internal dependencies.
- Dependencies are versioned, and conflicts can be resolved using dependency management.

4 CHECKING THE MAVEN REPOSITORY

The Maven repository is a storage location for all dependencies. There are three types of repositories:

- **Local Repository (.m2 folder):** Stores dependencies downloaded on the developer's machine.
- **Remote Repository (Maven Central):** Hosted online to provide dependencies for projects.
- **Private Repository (Nexus, Artifactory):** Used by companies for internal dependency management.

5 HOW ALL MODULES BUILD USING MAVEN?

Maven supports multi-module projects, allowing multiple submodules to be built under a parent POM.

- The parent module contains a pom.xml that defines shared dependencies and configurations.
- Each child module has its own pom.xml and inherits properties from the parent.
- Running mvn clean install at the root builds all modules in the correct order.
- The reactor mechanism ensures that dependencies within the modules are built first.

6 CAN WE BUILD A SPECIFIC MODULE?

Yes, instead of building the entire project, a specific module can be built using:

- Navigating to the module directory and running mvn clean install.
- Running from the root directory using mvn -pl module-name clean install.
- Skipping tests with mvn clean install -DskipTests.

This speeds up development by compiling only the required part of the project.

7 Role of ui.apps, ui.content, and ui.frontend Folders in AEM

In an Adobe Experience Manager (AEM) project, these folders serve different purposes:

- **ui.apps:** Contains AEM components, templates, and dialogs. It defines the structure and logic of content.
- **ui.content:** Stores actual website content, including pages, assets, and configuration settings.
- **ui.frontend:** Manages frontend resources like JavaScript, CSS, and client-side logic.

8 Why Are We Using Run Mode in AEM?

Run modes in AEM allow different configurations for different environments without changing code.

- **Author Mode:** For content authors to create and manage content.
- **Publish Mode:** For serving content to end users.
- **Development Mode:** Enables debugging and additional logs.
- **Production Mode:** Optimized for performance and security.

9 What is the Publish Environment?

The publish environment in AEM is where content is delivered to the end users.

- It is read-only, ensuring that published content is not modified.
- It works with the dispatcher to improve caching and security.
- Authors push content from author to publish instance using replication agents.

The publish environment ensures that the final website content is optimized for fast and secure delivery.

10 Why Are We Using Dispatcher in AEM?

The Dispatcher in AEM is used for caching and security.

- **Caching:** It caches pages to reduce load on the AEM publish instance and improves performance.
- **Load Balancing:** It distributes traffic across multiple AEM instances.
- **Security:** It filters requests and prevents unauthorized access.

Dispatcher is configured using .any files to define caching rules, URL filters, and security settings.

11 From Where Can We Access CRX/DE?

CRX/DE is the AEM content repository explorer, which allows developers to interact with stored data.

- It is accessed via http://localhost:4502/crx/de in author mode.
- Developers can browse JCR nodes, modify properties, and upload files.
- It helps in debugging AEM applications by directly manipulating stored content.