1. Maven life cycle

Maven has three key build lifecycles:

- **Clean**: Cleans the project by deleting the target/ folder.
- **Default (Build)**: Manages the complete build process, including compile, test, package, etc.
- **Site**: Generates project documentation.

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

The **pom.xml** (Project Object Model) is the core configuration file in a Maven project. It defines the project's structure, dependencies, plugins, and build configurations.

Key Elements in pom.xml:

- <groupId> – Identifies the organization or company (e.g., com.example).
- <artifactId> – Identifies the project or module (e.g., my-app).
- <version> – The project's version (e.g., 1.0.0).
- <dependencies> – Manages external libraries required for the project.
- <build> – Specifies plugins, goals, and profiles.
- <modules> – Lists submodules in a multi-module project.

3.How dependencies work?

Dependencies are external libraries (like JAR files) that your project needs. Maven simplifies dependency management by automatically fetching them from remote repositories.

Steps in Dependency Resolution:

1. Maven checks the **local repository** (~/.m2/repository/).
2. If the dependency is unavailable locally, Maven downloads it from the configured repositories (e.g., Maven Central).
3. Maven stores the downloaded dependency in the .m2 folder for future use.

4. How all modules build using maven?

In a multi-module project:Running mvn install from the **parent** project builds all modules in the correct order.

5.Can we build specific module?

To build a specific module:

```
cpp
CopyEdit
mvn install -pl <module-name> -am
```

- -pl: Specifies the module you want to build.
- -am: Builds required dependencies as well.

6. Role of ui.apps and ui.content and ui.frontend folder?

These folders are part of **AEM Maven Project Structure**.

ui.apps Folder:
- Contains code, servlets, and OSGi configurations.
- Holds components, templates, and dialogs.

ui.content Folder:
- Stores sample content, site structure, and pages.
- Useful for creating demo content for deployment.

ui.frontend Folder:
- Contains CSS, JavaScript, and other frontend resources.
- Uses modern frontend tools like Node.js, Webpack, etc.

7. Why we are using run mode?

Run Modes help manage different configurations for various environments (e.g., Development, Staging, Production).

Key Run Modes in AEM:

- author – Used for content creation and authoring.
- publish – Used for delivering content to end-users.
- dev, stage, prod – Custom modes for environment-specific configurations.

8. What is publish env?

The **Publish Environment** is responsible for serving published content to end-users.

Key Features:

- Users cannot modify content here (only view it).
- Content is replicated from the **Author Environment** to the **Publish Environment**.
- Typically secured behind a **Dispatcher** for caching and protection.

9. Why we are using dispatcher?

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

Improves performance by caching static content.
Reduces load on AEM instances by serving cached content.
Protects AEM instances by filtering malicious requests.

Dispatcher Configuration Files:

- dispatcher.any – Main configuration file.
- cache.rules – Defines what content should be cached.

10. From where can access the crx/de?

The **CRXDE Lite** interface is the development environment in AEM where you can:

View the JCR repository structure.
Create and modify nodes, properties, and files.
Manage components, templates, and dialogs.