Maven Lifecycle

Maven has **three built-in lifecycles**:

- **Clean**: Cleans up the project (mvn clean)
- **Default (Build)**: Main lifecycle (compile, test, package, install, deploy)
- **Site**: Generates project documentation

Key phases in the **default lifecycle**:

- validate – check if project is correct
- compile – compile the source code
- test – run unit tests
- package – package into a JAR/WAR
- install – install into local .m2 repo
- deploy – deploy to remote repo

What is pom.xml and Why We Use It?

- pom.xml stands for **Project Object Model**.
- It is the core file of a Maven project.
- Used to define:
  - **Project dependencies**
  - **Build plugins**
  - **Project information**
  - **Module structure** (for multi-module projects

How Dependencies Work?

When you define a dependency in pom.xml, Maven:

- Checks in the **local repository (~/.m2/repository)**

- If not found, downloads from **central repository** or other configured repos (e.g., Adobe's Nexus)

Maven manages **transitive dependencies** (dependencies of dependencies)

Example:

```
<dependency>
  <groupId>com.adobe.aem</groupId>
  <artifactId>uber-jar</artifactId>
  <version>6.5.0</version>
  <scope>provided</scope>
</dependency>
```

Check the maven repository

How all modules build using maven?

To build everything:

mvn clean install

This builds everything in order:

- core compiles Java classes
- ui.apps packages components
- ui.content packages content
- all (if present) builds the AEM package with the others included

Can we build specific module?

Yes.

Example:
cd core
mvn clean install

Role of ui.apps and ui.content and ui.frontend folder:

- ui.apps: Contains AEM-specific code like components, templates, clientlibs (installable under /apps)
- ui.content: Contains sample or live content (installable under /content)
- ui.frontend: Manages frontend code (JS, CSS, webpack, etc.); compiles to clientlib format for AEM
- Run separately using npm run build

Why we are using run mode?

Run Modes in AEM allow different configurations based on the environment:

- Examples: author, publish, dev, prod
- Used in OSGi config: config.author/com.example.Config.xml, config.publish/...
- Helps manage environment-specific settings (e.g., dispatcher configs, analytics, etc.)

What is Publish Environment?

Author: Content creation and editing

Publish: Live site delivery

  - Content replicated (activated) from Author to Publish
  - No editing allowed on Publish

Why we are using dispatcher?

Dispatcher is AEM’s caching & security layer.

Roles:

1. Cache HTML/Assets on Apache web server

2. Filter requests and block direct access to /bin, /apps
3. Load balancer if multiple publish instances

It sits between user and AEM Publish:

User → Dispatcher → Publish Instance

From where can access the crx/de?

You can access CRXDE Lite in Author (or Publish if enabled) via:

http://localhost:4502/crx/de