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. 1. 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. 2. Default Lifecycle: This is the primary lifecycle that compiles, tests, packages, and deploys the project. Common phases include: o validate (checks project structure) o compile (compiles source code) o test (runs unit tests) o package (creates a JAR/WAR file) o install (adds the package to the local repository) o deploy (pushes the package to a remote repository) 3. Site Lifecycle: Generates project documentation using mvn site. Each phase is executed in order, meaning running mvn package also runs compile and test automatically.

Checking the Maven Repository

The Maven repository is a storage location for all dependencies. There are three types of repositories: • Local Repository (.m2️ 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.

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.

## Maven Lifecycle

Maven follows a lifecycle with predefined phases such as `clean`, `validate`, `compile`, `test`, `package`, `verify`, `install`, and `deploy`.

## What is pom.xml and Why Do We Use It?

The `pom.xml` (Project Object Model) file is the core of a Maven project. It defines dependencies, plugins, goals, and project metadata. It helps in managing the build process and dependencies in a structured way.

## How Dependencies Work?

Dependencies in Maven are libraries or modules required by the project. These are defined in `pom.xml`, and Maven fetches them from repositories (like Maven Central).

## Check the Maven Repository

Maven repositories are storage locations for dependencies. You can browse dependencies at https://mvnrepository.com/

## How All Modules Build Using Maven?

Maven can build all modules in a multi-module project using the `mvn install` command at the parent module level. It builds each module based on dependency hierarchy.

## Can We Build a Specific Module?

Yes, you can build a specific module by navigating to its folder and running `mvn install`. Alternatively, use `mvn install -pl module-name -am`.

1. **Role of ui.apps, ui.content, and ui.frontend Folders**

- ui.apps: Contains AEM-specific components, templates, and configurations.  
- ui.content: Stores AEM content such as pages and sample content.  
- ui.frontend: Manages the front-end (CSS, JavaScript, clientlibs).

## Why We Use Run Mode?

Run modes in AEM help define environment-specific configurations (e.g., development, staging, production). They allow customization without modifying the core codebase.

## What Is the Publish Environment?

The Publish environment in AEM serves final content to end users. It ensures that only approved content is accessible on the live website.

## Why We Use Dispatcher?

The Dispatcher is a caching and security tool for AEM. It helps improve performance and protects against unauthorized access by caching content and filtering requests.

## From Where Can We Access crx/de?

The `crx/de` (CRX Development Environment) can be accessed at:  
- `http://localhost:4502/crx/de (Author)  
- `http://localhost:4503/crx/de (Publish)