**Maven Lifecycle**

Maven follows a structured lifecycle with phases such as clean, validate, compile, test, package, verify, install, and deploy. Each phase executes specific goals, ensuring the build process is systematic and repeatable.

**What is pom.xml and why do we use it?**

pom.xml (Project Object Model) is Maven's configuration file that defines project structure, dependencies, plugins, and build configurations. It acts as a blueprint for Maven to manage the build, test, and deployment processes.

**How dependencies work?**

Maven dependencies are defined in pom.xml under the <dependencies> section. When a dependency is declared, Maven downloads the required JAR files from the Maven Central Repository (or other repositories) and manages transitive dependencies automatically.

**Check the Maven Repository.**

The Maven Central Repository (https://mvnrepository.com/) is the default source for dependencies. Additionally, Maven maintains a local repository (.m2/repository) where it caches downloaded dependencies for faster future builds.

**How are all modules built using Maven?**

Maven uses a multi-module build approach where the parent pom.xml defines shared configurations, and child modules inherit them. Running mvn install at the root level compiles and builds all modules sequentially.

**Can we build a specific module?**

Yes, a specific module can be built by navigating to its directory and running:

mvn clean install

Alternatively, from the root project, specify the module:

mvn install -pl <module-name> -am

-pl (projects list) specifies the module, and -am (also make) ensures dependencies are built.

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

ui.apps: Contains AEM-specific configurations, templates, and OSGi bundles.

ui.content: Holds sample content and content package configurations.

ui.frontend: Manages the frontend assets (CSS, JS, client-side libraries) built with tools like Webpack or Node.js.

**Why are we using run modes?**

Run modes in AEM allow different configurations for different environments (e.g., development, staging, production). They enable modular configuration and prevent unnecessary settings from being loaded.

**What is a Publish Environment?**

A Publish Environment is where content is made publicly available. AEM distinguishes between Author (where content is created) and Publish (where content is delivered to end users). The publish environment serves optimized content to visitors.

**Why are we using Dispatcher?**

The Dispatcher is AEM's caching and security tool, improving performance by serving cached pages and protecting against unauthorized access. It reduces load on AEM servers by handling requests efficiently.

**From where can we access crx/de?**

The CRXDE (Content Repository Extreme Development Environment) can be accessed via:  
http://localhost:4502/crx/de (for Author)  
http://localhost:4503/crx/de (for Publish)  
It allows developers to view and modify JCR repository content directly.