AEM ASSIGNMENT

Maven Life Cycle

Maven follows a set of predefined steps (phases) to build and manage projects. The main life cycle includes:

1. **Clean** – Deletes old build files.
2. **Compile** – Compiles the source code.
3. **Test** – Runs unit tests.
4. **Package** – Bundles the compiled code into JAR/WAR.
5. **Install** – Saves the package in the local repository.
6. **Deploy** – Sends the package to a remote repository.

Each phase depends on the previous one, meaning if you run mvn package, it automatically compiles and tests before packaging.

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

pom.xml (Project Object Model) is like the heart of a Maven project. It's an XML file where you define everything about the project—dependencies, plugins, build instructions, and configurations.

Why use it? Because it keeps everything organized. Instead of manually downloading libraries, Maven reads pom.xml, grabs what's needed, and sets up the project.

How Dependencies Work?

Dependencies are the external libraries your project needs. In pom.xml, you just add:

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>5.3.20</version>

</dependency>

Maven then downloads it from the **Maven repository** and keeps it in your system's .m2 folder. It even handles version conflicts and transitive dependencies (dependencies of dependencies).

How All Modules Build Using Maven?

In multi-module projects, there's a parent POM that links all modules. Running mvn install from the parent directory builds everything in order. If a module depends on another, Maven handles the sequence automatically.

Can We Build a Specific Module?

Yes, you don't need to build the whole project every time. You can build a specific module by running:

"mvn install -pl module-name -am"

This builds only the module you want while resolving dependencies.

Role of ui.apps, ui.content, and ui.frontend Folder?

- **ui.apps** – Contains all AEM code (components, templates, configs).
- **ui.content** – Stores content packages like pages and assets.
- **ui.frontend** – Handles frontend stuff (CSS, JS, client libraries).

Each has its own role, but together, they make up an AEM project structure.

Why Are We Using Run Mode?

Run modes allow AEM to behave differently based on the environment (dev, staging, production). You can configure settings like logging, users, or features differently for each environment.

Example: author.dev, author.prod – same AEM, different configs.

What is Publish Env?

The **publish environment** is where content goes live for users. Unlike the author environment (where you create and edit content), publish is optimized for performance and security.

Why Are We Using Dispatcher?

Dispatcher is like a **security guard + performance booster** for AEM. It caches content (so pages load faster) and filters requests (so bad traffic stays out). It sits in front of the AEM publish instance.

From Where Can We Access crx/de?

You can access it by going to:

http://localhost:4502/crx/de

It’s the **AEM CRXDE Lite** where you manage JCR content, nodes, and configurations. If you're on a publish instance, replace 4502 with the correct port (e.g., 4503).