

1. Maven Lifecycle

Maven automates the build process by following a structured lifecycle composed of predefined phases. The primary lifecycles are:

- **Clean Lifecycle:** Removes previous build artifacts.
- **Default (Build) Lifecycle:** Handles compilation, testing, packaging, and deployment.
- **Site Lifecycle:** Generates project documentation.

2. What is pom.xml and Why is it Important?

The **POM (Project Object Model) file** is the central configuration file in Maven projects. It defines dependencies, build plugins, and project metadata.

Why is pom.xml Critical?

- Manages dependencies centrally.
- Standardizes the build across environments.
- Allows plugin integration for custom builds.
- Supports multi-module project structures.

3. How Dependencies Work in Maven

Maven automates dependency management by fetching libraries from repositories like Maven Central.

Example of Adding a Dependency:

```
<dependency>  
  <groupId>org.springframework.boot</groupId>  
  <artifactId>spring-boot-starter-web</artifactId>  
  <version>2.7.0</version>  
</dependency>
```

To view project dependencies, run:

```
mvn dependency:tree
```

4. Checking the Maven Repository

Locally, Maven stores dependencies in:

- **Linux/macOS:** ~/.m2/repository/
- **Windows:** C:\Users\YourUsername\.m2\repository\

5. Building All Modules in a Multi-Module Project

In multi-module projects, a parent POM manages submodules.

Parent pom.xml:

```
<modules>

  <module>module1</module>

  <module>module2</module>

</modules>
```

To Build All Modules:

mvn clean install

6. Building a Specific Module

To build only a specific module:

mvn clean install -pl module-name -am

Flags:

- -pl – Specifies the module.
- -am – Builds dependencies automatically.

7. Role of ui.apps, ui.content, and ui.frontend in AEM

AEM projects use a structured folder setup:

ui.apps (Code and Components)

- Stores templates, components, and OSGi configurations.
- Includes /apps/ and /etc/ content.

ui.content (Site Content)

- Contains actual website content deployed to /content/.
- Stores pages, DAM assets, and site structure.

ui.frontend (CSS & JS Management)

- Handles client-side assets.
- Uses Webpack for asset bundling.

8. Why Are Run Modes Used in AEM?

Run modes allow **environment-specific configurations**, ensuring flexibility across different deployment setups.

Example Configuration:

```
<config>
  <property name="run.mode" value="author"/>
</config>
```

9. What is a Publish Environment in AEM?

A **publish environment**:

- Serves content to end-users.
- Stores only published pages.
- Works with Dispatcher for caching and security.

10. Why Use Dispatcher in AEM?

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

Benefits:

Improves performance via caching.

Enhances security by restricting access.

Reduces load on publish instances.

Dispatcher configs are stored in `/etc/httpd/conf.dispatcher.d/`.

11. How to Access CRX/DE?

CRX/DE (Content Repository Explorer) manages JCR content.

To access:

- **Author:** <http://localhost:4502/crx/de/index.jsp>
- **Publish:** <http://localhost:4503/crx/de/index.jsp>