# DAY 1 TASK

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

* **pom.xml (Project Object Model)** is the configuration file for **Maven**, an automation tool used for managing Java-based projects.
* It defines the **project dependencies**, build process, plugins, and execution goals.
* Maven uses pom.xml to ensure that all required libraries are downloaded and configured correctly.

**How do dependencies work in Maven?**

* Dependencies are external libraries required for the project to work.
* They are listed inside pom.xml under the <dependencies> section.
* Maven automatically downloads these dependencies from the **Maven Central Repository** or a specified repository and stores them in the local .m2 folder.

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-lang3</artifactId>

<version>3.12.0</version>

</dependency>

**How to check the Maven repository?**

* **Online Maven Repository**: <https://mvnrepository.com/> (for searching dependencies).
* **Local Maven Repository**: Windows: C:\Users\training\author\.m2\repository

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

* AEM projects usually follow a **multi-module Maven structure**.
* The parent pom.xml manages all the submodules (like ui.apps, core, ui.content, etc.).
* Running: mvn clean install
  + Cleans the project (clean), compiles and installs dependencies (install).
  + This builds all modules sequentially.

**Can we build a specific module?**

* to build only one module, use: mvn clean install -pl module-name -am
  + -pl → Specifies the module to build.
  + -am → Builds required dependencies as well.

**Role of ui.apps, ui.content, and ui.frontend folders?**

* **ui.apps** → Stores AEM **components, templates, and OSGi configurations**. It contains the essential functionalities of the AEM project.
* **ui.content** → Holds **actual content** like pages, assets, and content structures that are deployed in AEM.
* **ui.frontend** → Contains **front-end assets** (CSS, JS, React, Angular). It is responsible for the visual part of the website.

**Why are we using Run Modes?**

* Run Modes define **different environment configurations** (dev, stage, prod).
* This allows AEM to **load specific settings** based on the environment, without modifying the codebase.
* Example:
  + author mode → Used for content creation.
  + publish mode → Serves the content to end users.

**What is a Publish Environment?**

* The **Publish Instance** is responsible for delivering content to the end-users.
* It does **not allow authoring**; content comes from the **Author Instance**.
* **Example URLs:**
  + **Author:** http://localhost:4502
  + **Publish:** http://localhost:4503

**Why are we using a Dispatcher?**

* **AEM Dispatcher** is used for **caching, security, and load balancing**.
* It caches pages to improve performance and acts as a firewall to protect AEM instances.
* It prevents unnecessary hits to the AEM **Publish Instance**, reducing load and improving site speed.

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

* **URL:** http://localhost:4502/crx/de/index.jsp
* This is the **AEM CRXDE (Content Repository eXplorer and Development Environment)**.