What is the purpose of the core module in AEM?

**Ans:** The core module in AEM contains backend logic, written in Java, that provides various components. It includes Java classes that define how data is processed and passed to the frontend.

What kind of files and code can be found in the core folder?

**Ans:** Java Classes, Sling Models, Unit Test Files, OSGi Services

Explain the role of ui.apps in AEM projects.

**Ans**: The ui.apps module contains all frontend-related files like components, templates, and client libraries. It defines how pages and components appear and function in AEM.

How are components structured in the ui.apps folder?

**Ans**: ui.apps/src/main/content/jcr\_root/apps/project-name/components/

**Hello World Component:**

* Where is the Hello World component located in both core and ui.apps?

**Ans:** In core: core/src/main/java/com/project/models/HelloWorldModel.java

In ui.apps: ui.apps/src/main/content/jcr\_root/apps/project/components/helloworld

* Explain the Java class (in core) for the Hello World component.

**Ans**: A Sling Model that fetches the "message" property and makes it available to the frontend. If the property is not set, it returns a predefined default message.

* How does the HTL script work in ui.apps for Hello World?

**Ans:** HTL pulls data from the model and displays it.

* How are properties and dialogs defined for this component?

**Ans:** In \_cq\_dialog.xml, the dialog includes a text field that allows authors to enter a custom message. This input is stored in the JCR and used by the component.

What are the different types of AEM modules (core, ui.apps, ui.content, etc.)?

**Ans:** core – Backend Java code (logic, models, services).

ui.apps – Frontend components (HTL, JS, CSS).

ui.content – Sample content (pages, templates).

dispatcher – Configurations for caching and load balancing**.**

How does Maven build these modules?

**Ans:** Maven compiles Java files, processes frontend files, and bundles everything into an AEM package.

Explain the build lifecycle of Maven in the context of AEM.

**Ans:** Maven follows these stages:

1. **validate** – Checks project structure.
2. **compile** – Compiles Java files.
3. **test** – Runs unit tests.
4. **package** – Bundles files into a .jar or .zip
5. **install** – Saves the package locally.
6. **deploy** – Uploads the package to AEM.

How are dependencies managed in pom.xml?

**Ans**: Maven handles dependencies in pom.xml using <dependencies> tags. It automatically fetches the required libraries from repositories and maintains version consistency.

Why is Maven used instead of other build tools?

**Ans:** Maven simplifies dependency management, automates builds, and integrates well with AEM.

What advantages does Maven offer for AEM development?

**Ans:** 1. Works the same on all systems.

2.Automatically downloads needed libraries.

3.Deploys directly to AEM.

How does Maven help in managing dependencies and plugins in AEM projects?

**Ans:** Maven downloads required dependencies and integrates plugins to handle different tasks, like unit testing and package deployment.

What does mvn clean install do in an AEM project?

**Ans**: 1. clean – Deletes old build files.

2. install – Builds and stores the package locally.

How to deploy packages directly to AEM using Maven commands?

**Ans:** mvn clean install -PautoInstallPackage

Explain the purpose of different Maven profiles in AEM (autoInstallPackage, autoInstallBundle).

**Ans:** 1. autoInstallPackage – Deploys the entire package that is both code and content.

2. autoInstallBundle – Deploys only the Java bundle.

What is the purpose of dumplibs in AEM?

**Ans:** It helps developers see which client libraries are loaded on a page.

How can you view client libraries using dumplibs?

**Ans:** 1.Open: http://localhost:4502/libs/granite/ui/content/dumplibs.html

2. Enter the page path to see all loaded client libraries

Explain how client libraries are structured in AEM.

**Ans:** ui.apps/src/main/content/jcr\_root/apps/project-name/clientlibs/