**What is the purpose of the core module in AEM?**

* The core module in AEM is used to create reusable components with backend logic. It helps to keep the code well-organized, making it easier to maintain and update.
* This module improves performance and makes AEM projects more efficient.

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

The core folder in AEM has Java files that control how components work including Sling Models Servlets and helper classes to get data and handle content and it also has OSGi services that help different parts of AEM communicate these files make sure that the website run smoothly and is easy to manage.

**Explain the role of ui.apps in AEM projects**.

The ui.apps folder in AEM is used to store the front-end-related codes such as components templates dialogs and client libraries it contains everything needed for rendering content in the AEM site this folder is deployed to the AEM instance, it also includes resource files like XML HTML CSS JS and images it helps in defining how content appears and interacts with users.

**How are components structured in the ui.apps folder?**

In the ui.apps folder, each component has its own folder with different files such as, the HTML file defines how the component looks, while XML files store settings. If the component needs logic, it may be having a .js file, and CSS for styles. Some components also have a dialog folder for user inputs.

**Hello World Component:**

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

It is present in two modules:

1. In the core module, it is in /core/src/main/java/com/project/models.
2. In the ui.apps module, it is in /apps/project-name/components/helloworld."

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

It is a Sling Model that fetches and process the data. It helps in providing dynamic content to the component using backend logic.

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

The HTL script displays data from the backend using expressions like ${properties.text}. It makes the component dynamic and interactive.

* **How are properties and dialogs defined for this component?**

They are set in dialog.xml, where fields like text and title are added. Authors can enter values, which are stored in the JCR and displayed on the page.

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

It has different AEM modules such as the,

* The core module contains backend logic using Sling Models and OSGi services.
* The ui.apps module holds frontend components, dialogs, and client libraries.
* The ui.content module stores site content, templates, and initial data for pages.
* Other modules like ui.config manage OSGi configurations, and dispatcher modules handle caching and request handling.

**How does Maven build these modules**?

Maven builds these modules using the multi-module project structure, where it follows the parent POM file, which defines dependencies and plugins. Each module (core, ui.apps, ui.content, etc.) has its own POM file specifying how it should be built. When running “mvn clean install -PautoInstallPackage”, Maven compiles the code, packages the modules, and deploys them to the AEM instance.

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

Clean-> Deletes previous build files.

Validate-> Checks if the project is correct and all dependencies are available.

Compile-> Converts Java files into bytecode.

Test-> Runs unit tests.

Package-> Creates JAR or ZIP files for deployment.

Install-> Places the built package into the local repository.

Deploy-> Installs the package into the AEM instance.

**How are dependencies managed in pom.xml?**

Dependencies in AEM are managed in the pom.xml file. Under the <dependencies> there are list that has the required libraries, and Maven automatically downloads them. The <dependencyManagement> section makes sure all modules use the same versions. Plugins in <plugin> help with tasks like building and deploying AEM packages.

**Why is Maven used instead of other build tools?**

Maven is used because it simplifies project structure, and it also manages dependencies.

**What advantages does Maven offer for AEM development?**

* Simplifies project creation.
* Automatically manages dependencies.
* Ensures consistency across projects.

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

Maven manages by :

* pom.xml which defines project dependencies and plugins.
* Automatically downloads and manages dependencies.
* Configures and executes plugins.

**What does mvn clean install do in an AEM project?**

The command mvn clean install is used to build the AEM project. clean deletes old build files, and install compiles the code, runs tests, and packages the project.

**How to deploy packages directly to AEM using Maven commands?**

To deploy packages to AEM using Maven:

* Configure the pom.xml file
* Run the **mvn clean install** command to build and package the project.
* Use the **mvn deploy** command to deploy the package to AEM.

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

* The purpose of **autoInstallPackage** is to automatically installs packages in AEM.
* The purpose of **autoInstallBundle** is to automatically installs bundles in AEM.

**What is the purpose of dumplibs in AEM?**

It’s main purpose is to help developers by allowing them to inspect client libraries like JavaScript and CSS files, debug client-side issues, and optimize client library performance

**How can you view client libraries using dumplibs?**

* Add **?debugClientLibs=true** to the AEM URL.
* Access the AEM page using the client library.
* View client library files via the dumplibs URL.

**Explain how client libraries are structured in AEM.**

Client libraries are structured by:

* Category
* Embedding
* Dependencies
* Compression
* Storage in clientlibs.