1. What is the purpose of the core module in AEM?

Ans: The core module in AEM contains the backend logic, including OSGi components, Sling Models, and Servlets. It handles business logic, data processing, and interactions with the JCR (Java Content Repository).

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

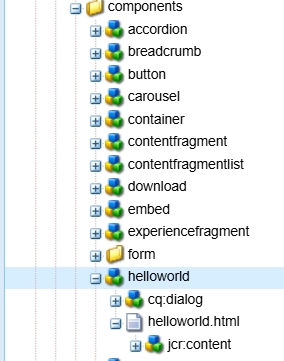
Ans: The core folder in AEM contains all the backend code that powers the project. It includes Sling Models for fetching content, OSGi services for business logic, and Servlets for handling HTTP requests. It also has scheduled tasks, configuration files like pom.xml for dependencies, and test files to ensure everything works correctly. This module plays a key role in providing the right data and functionality to the front end.

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

Ans: The ui.apps module in AEM stores the frontend components, including HTL templates, dialogs, client libraries (CSS & JS), and content structures. It defines how components look and interact in the AEM editor, making it essential for building the user interface.

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

Ans: In the ui.apps folder, components are organized under the /components directory, with each component having its own folder. Inside, you'll find HTL templates (.html) for rendering, cq:dialog files for configuring authoring options, and clientlibs for CSS and JavaScript. This structure keeps components modular, reusable, and easy to manage in AEM.

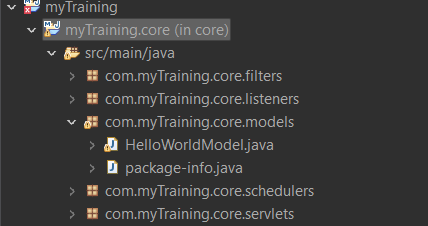


5. Hello World Component:

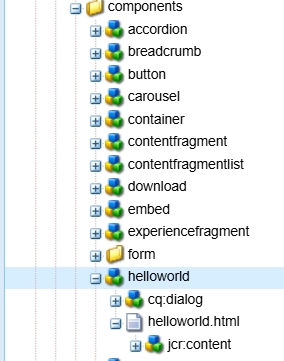
* Where is the Hello World component located in both core and ui.apps?

Location of Hello World Component:

1. In core: core/src/main/java/com/myproject/core/models/HelloWorldModel.java



2. In ui.apps: ui.apps/src/main/content/jcr\_root/apps/myproject/components/helloWorld



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

The HelloWorldModel class is a Sling Model that helps fetch data for the component. It uses @Model(adaptables = Resource.class) to access AEM resources, injects properties with @Inject, and provides getter methods so the data can be displayed in the HTL template.

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

The HTL script in ui.apps fetches data from the HelloWorldModel and displays it on the page using ${model.message}. It ensures content is dynamically rendered based on the model’s data.

* How are properties and dialogs defined for this component?

The Hello World component's properties and dialogs are defined in the cq:dialog file within the component folder in ui.apps. This dialog enables authors to input values in AEM’s Touch UI, which are saved as properties in the JCR. These properties are then retrieved by the Sling Model and displayed in the HTL script.

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

Ans: Different types of modules:

* core → Contains backend logic like Sling Models, OSGi services, and Servlets.
* ui.apps → Stores frontend components, HTL scripts, dialogs, and client libraries.
* ui.content → Manages content like templates, policies, and initial data.
* ui.config → Holds OSGi configurations for different environments.
* all → Packages and deploys all modules together.

7. How does Maven build these modules?

Ans: Maven builds AEM modules using a parent POM that manages all submodules (core, ui.apps, etc.). Each module has its own POM file with dependencies and plugins. The core module is built as an OSGi bundle, while ui.apps and ui.content are packaged as AEM ZIP files. Running mvn clean install compiles, packages, and prepares the project for deployment.

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

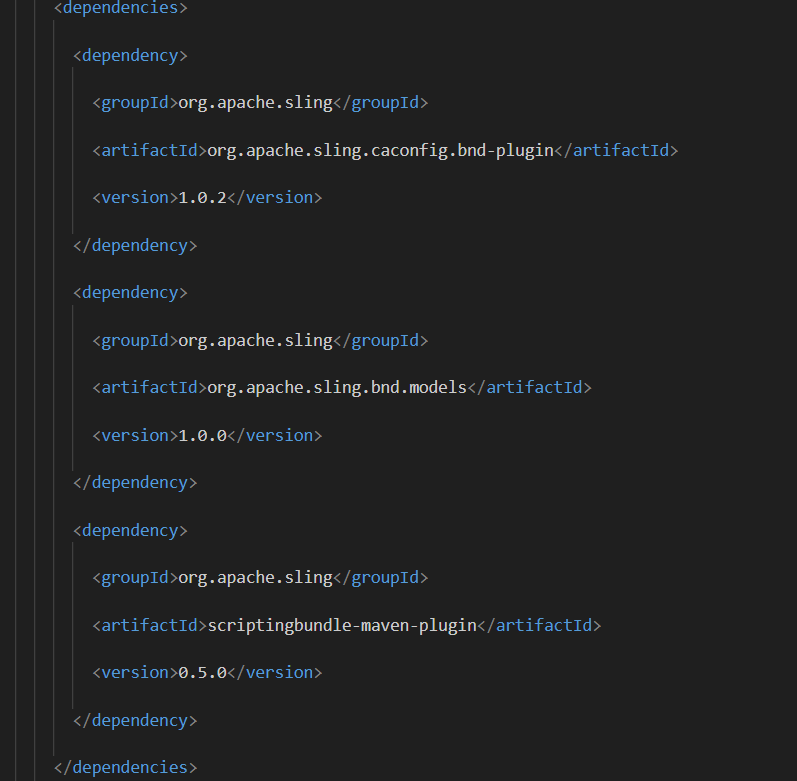
Ans: Maven's build lifecycle in AEM follows these key phases:

1. clean : Removes old build files.
2. compile : Compiles Java code (core module).
3. package : Bundles core as an OSGi JAR and ui.apps/ui.content as ZIPs.
4. install : Saves built packages locally.
5. deploy : Deploys to AEM.

AEM uses content-package-maven-plugin and maven-bundle-plugin for packaging and deployment. Running mvn clean install -PautoInstallPackage automates the process.

9. How are dependencies managed in pom.xml?

Ans: In pom.xml, we list the required libraries (like AEM APIs and OSGi) in the dependencies section. The parent pom.xml makes sure all modules use the same versions. Maven plugins like content-package-maven-plugin (for packaging) and maven-bundle-plugin (for OSGi bundles) help automate the build. Maven then downloads everything from online repositories, making sure the project compiles and runs smoothly.



10. Why is Maven used instead of other build tools?

Ans: Maven is used because it automates dependency management, builds, and deployments. It follows a standard project structure, downloads required libraries automatically, and has powerful plugins for tasks like packaging and deployment. It also supports multi-module projects, making it ideal for AEM development.

11. What advantages does Maven offer for AEM development?

Ans: Maven makes AEM development easier with automated dependency management, a standard project structure, and powerful plugins for packaging and OSGi bundling. It supports multi-module projects and simplifies deployment with commands like mvn clean install -PautoInstallPackage.

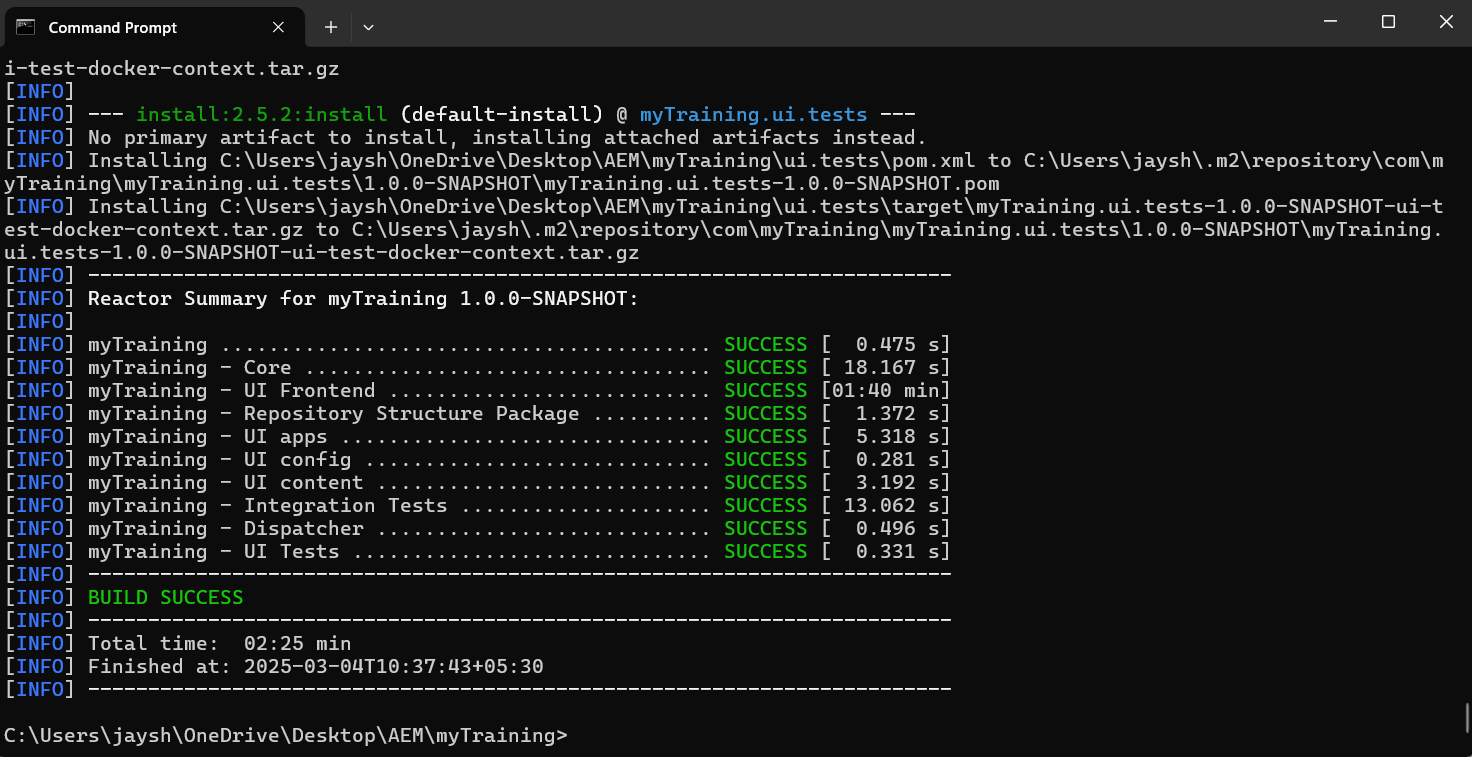
12. How does Maven help in managing dependencies and plugins in AEM projects?

Ans: Maven manages dependencies by automatically downloading required libraries (like AEM APIs, Sling, and OSGi) from repositories. The parent pom.xml ensures all modules use the same versions.

For plugins, Maven uses tools like content-package-maven-plugin (for packaging AEM content) and maven-bundle-plugin (for OSGi bundles), automating tasks like building, packaging, and deployment.

13. What does mvn clean install do in an AEM project?

Ans: mvn clean install cleans up old files, compiles the Java code, and packages everything needed for the AEM project. It then saves the build locally so it can be used later. However, this doesn’t deploy it to AEM. To install it on AEM, you need to run mvn clean install -PautoInstallPackage.



14. How to deploy packages directly to AEM using Maven commands?

Ans: To deploy packages to AEM using Maven, the following commands are used:

1. Full Project Deployment: mvn clean install -PautoInstallPackage → Builds and installs all modules (core, ui.apps, ui.content).
2. Deploy Only ui.apps and ui.content: mvn clean install -PautoInstallSinglePackage → Installs only the main package without rebuilding everything.
3. Deploy a Specific Module (e.g., core): cd core && mvn clean install -PautoInstallBundle → Installs only the backend (core) module.

These commands deploy the package to http://localhost:4502, provided AEM is running.

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

Ans: Maven profiles in AEM help in managing different build and deployment tasks efficiently. The most commonly used profiles are autoInstallPackage and autoInstallBundle.

The autoInstallPackage profile is used to build and deploy the entire project, including core, ui.apps, and ui.content, to AEM. It ensures that all modules are installed properly. The command to use this is: mvn clean install -PautoInstallPackage

The autoInstallBundle profile is used when only the backend (core module) needs to be deployed. This is helpful when making changes to Java code without affecting the front-end. The command is: cd core && mvn clean install -PautoInstallBundle

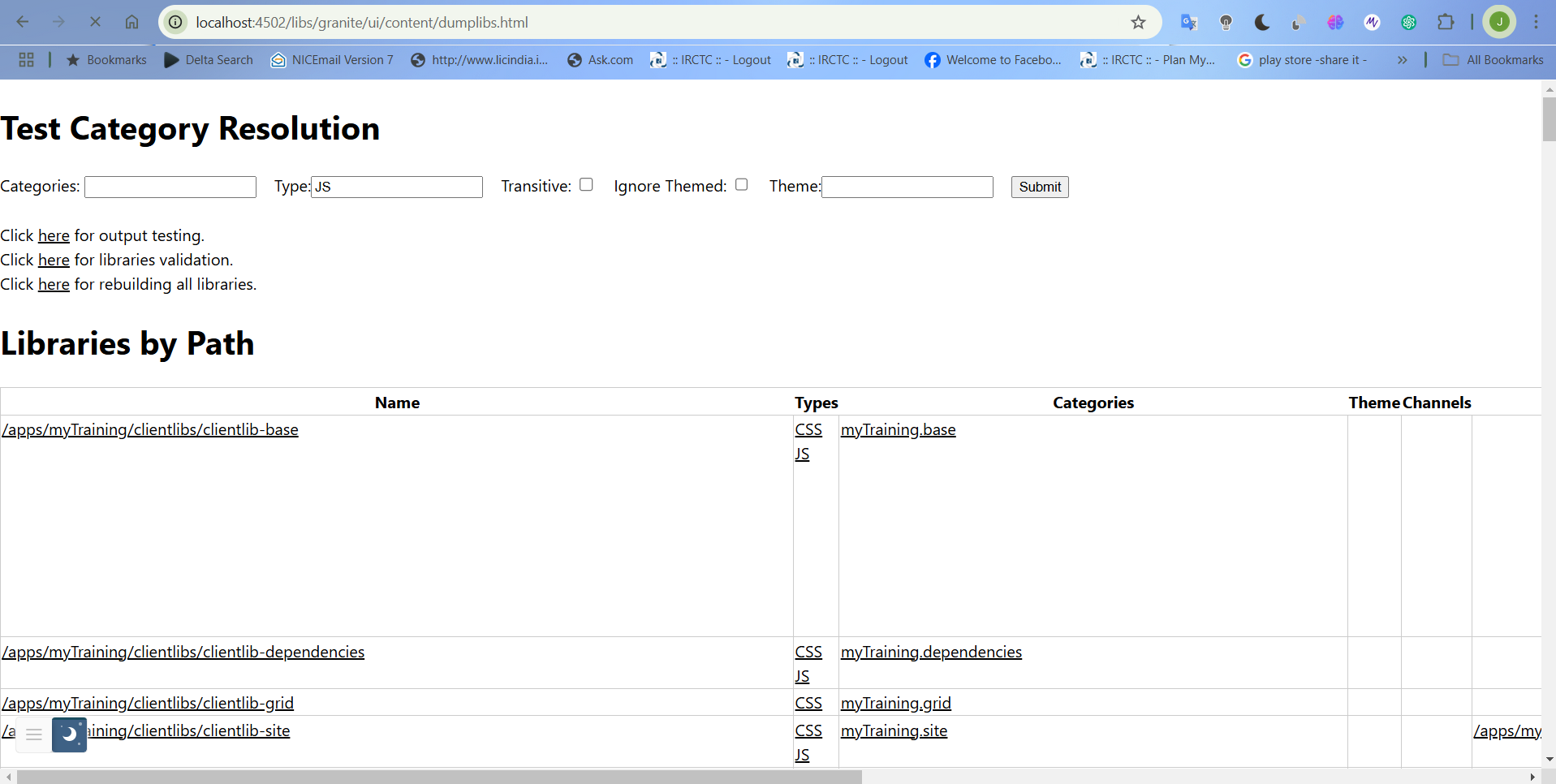
These profiles help speed up the development process by allowing targeted deployments instead of rebuilding the entire project every time.

16. What is the purpose of dumplibs in AEM?

Ans: The dumplibs tool in AEM is used to debug and analyze client libraries (clientlibs). It helps developers check which CSS and JavaScript files are loaded, their dependencies, and how they are included in a page. By accessing /etc.clientlibs, developers can view all available client libraries and troubleshoot missing or incorrectly loaded assets. This is useful for performance optimization and ensuring the correct files are delivered to the browser.

17. How can you view client libraries using dumplibs?

Through this url: http://localhost:4502/libs/granite/ui/content/dumplibs.html



18. Explain how client libraries are structured in AEM.

Ans: Client Libraries in AEM manage CSS, JavaScript, and other front-end assets. They are stored under:/apps/project-name/clientlibs

Each clientlib folder contains css files, js files and resources.

Key files are :

* q:ClientLibraryFolder : Marks it as a client library.
* js.txt & css.txt : Defines load order.
* categories : Assigns a unique category for usage.

