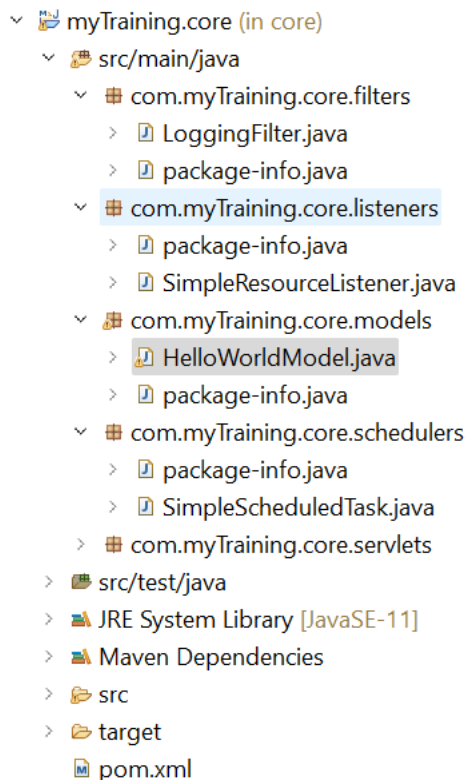


1.What is the purpose of the core module in AEM?

It is responsible to store backend logic of an application. It contains classes, OSGi services and sling models that provide business logic. Process data and interact with AEM's Repository.

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

In core folder it contains a backend logic written in java. It is structured as a Maven module and builds an OSGi bundles that runs inside AEM's container.




















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

It is responsible for storing and deploying front end codes like components, templates, client libraries into AEM's repository. It also ensures that all content, styles and scripts are properly structured and available for rendering AEM pages.

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















AEM components in the ui.apps folder follow a structured hierarchy under /apps/<project-name>/components/, where each folder has its own folder.

- >  apps.myTraining.components.accordion
- >  apps.myTraining.components.accordion._cq_template
- >  apps.myTraining.components.breadcrumb
- >  apps.myTraining.components.button
- >  apps.myTraining.components.carousel
- >  apps.myTraining.components.container
- >  apps.myTraining.components.contentfragment
- >  apps.myTraining.components.contentfragmentlist
- >  apps.myTraining.components.download
- >  apps.myTraining.components.embed
- >  apps.myTraining.components.experiencefragment
- >  apps.myTraining.components.form.button
- >  apps.myTraining.components.form.container
- >  apps.myTraining.components.form.hidden
- >  apps.myTraining.components.form.options
- >  apps.myTraining.components.form.text
- >  apps.myTraining.components.helloworld

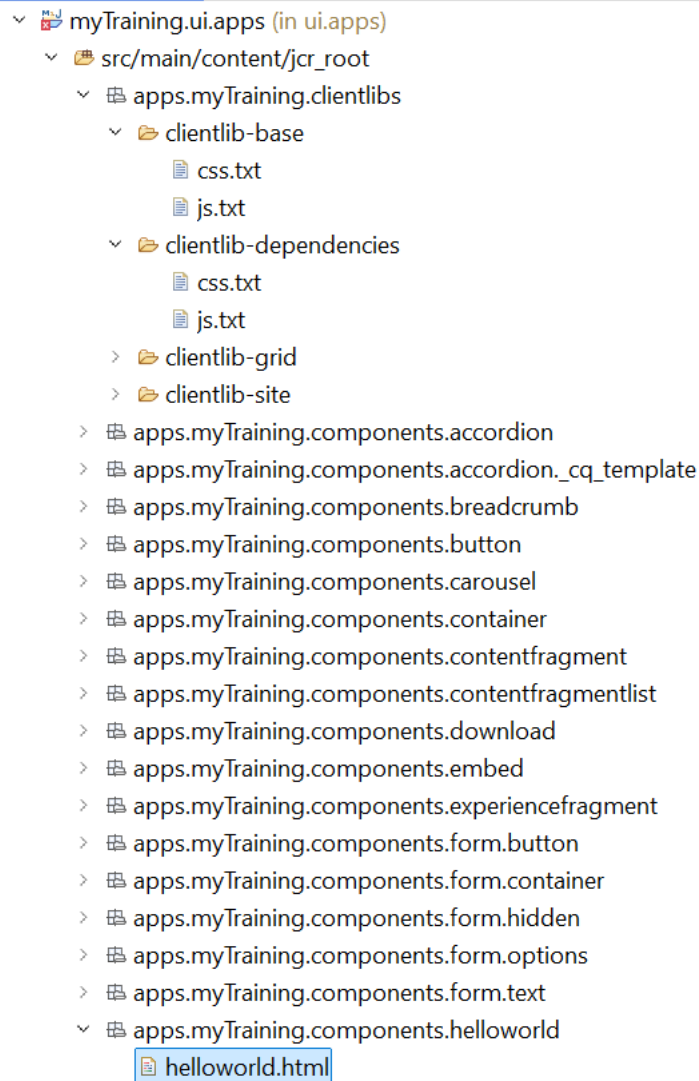
5. Hello World Component:

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

In core:

- >  Maven Dependencies
- >  src
 - >  main
 - >  java
 - >  com
 - >  myTraining
 - >  core
 - >  filters
 - >  listeners
 - >  models
 -  HelloWorldModel.java
 -  package-info.java
 - >  schedulers
 - >  servlets

In apps:



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

This Java class defines a Sling Model for the "Hello World" component in AEM. It retrieves the resource type, current page path, and resource properties and generates a message.











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

In AEM, the HTL (Sightly) script is responsible for rendering the frontend of the Hello World component by fetching data from the Sling Model (HelloWorld.java) in the core module.

- How are properties and dialogs defined for this component?

The dialog (`_cq_dialog.xml`) allows authors to input a message, which is stored in JCR as `./message`. The Sling Model (`HelloWorldModel.java`) retrieves this property using `@ValueMapValue`, defaulting to "Hello, World!" if empty. The HTL (`helloworld.html`) then displays the message dynamically on the page.

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

- >  myTraining.all (in all)
- >  myTraining.core (in core)
- >  myTraining.dispatcher.ams (in dispatcher)
- >  myTraining.it.tests (in it.tests)
- >  myTraining.ui.apps (in ui.apps)
- >  myTraining.ui.apps.structure (in ui.apps.structure)
- >  myTraining.ui.config (in ui.config)
- >  myTraining.ui.content (in ui.content)
- >  myTraining.ui.frontend (in ui.frontend)
- >  myTraining.ui.tests (in ui.tests)

7.How does Maven build these modules?

Maven compiles, packages, and deploys the AEM project using the AEM Archetype structure, where each module (core, ui.apps, etc.) is built separately and then combined into an installable package.

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

Maven builds AEM projects through sequential **phases**, ensuring proper compilation, packaging, and deployment.

- **compile** → Converts Java code in core into an **OSGi bundle (core.jar)**.
- **Package** → Creates **ui.apps.zip** with components, dialogs, and clientlibs.
- **install** → Combines core.jar + ui.apps.zip into a deployable **AEM package**.
- **deploy** → Installs the package into AEM using **CRX Package Manager**.

Command to build and deploy:

mvn clean install -PautoInstallPackage

9.How are dependencies managed in pom.xml?

Maven manages dependencies using the <dependencies> section in pom.xml, pulling required libraries from repositories.

- 1)**AEM APIs (aem-sdk-api)** → Provides AEM services for the core module.
- 2) **OSGi Bundles (org.apache.sling.models.api)** → Supports Sling Models and OSGi services.
- 3) **Content Packages (content-package-maven-plugin)** → Bundles and deploys ui.apps.
- 4) **Dependency Management (dependencyManagement)** → Centralizes shared dependencies in the parent pom.xml.

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

Maven is preferred for AEM projects due to its structured dependency management, modular build lifecycle, and seamless integration with AEM.

11.What advantages does Maven offer for AEM development?

Maven simplifies building, packaging, and deploying AEM projects by providing a modular, automated, and dependency-managed workflow.

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

Maven simplifies dependency and plugin management in AEM projects by automatically resolving dependencies and integrating essential plugins for building, packaging, and deploying AEM applications.

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

used to build and package an AEM project by executing Maven's build lifecycle.

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

`mvn clean install -PautoInstallPackage`

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

autoInstallPackage->Deploy the full AEM Package that is all zip ,used when you need to install both frontend and backend changes together.

autoInstallBundle->it only install core.jar , used for only backend needs.

16.What is the purpose of dumplib in AEM?

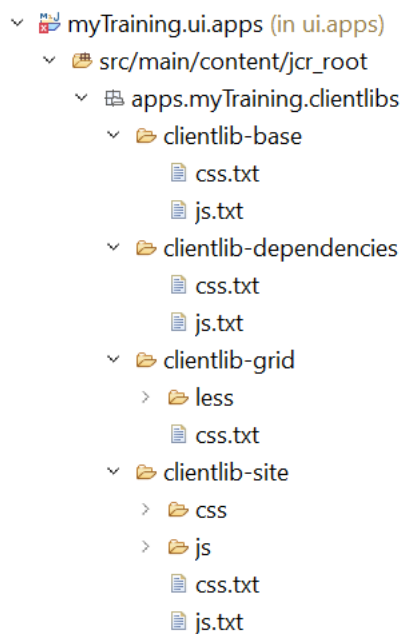
It is a debugging tool in AEM used to analyze and troubleshoot Client Libraries.

17.How can you view client libraries using dumplib?

<http://localhost:4502/libs/granite/ui/content/dumplib.html>

18.Explain how client libraries are structured in AEM.

Client Libraries (clientlibs) in AEM are used to **manage and serve CSS, JavaScript, and other frontend assets efficiently**. They are stored under /apps or /etc and follow a structured format.



The screenshot shows a file explorer view of an AEM project structure. The root is 'myTraining.ui.apps (in ui.apps)'. Under it is 'src/main/content/jcr_root'. Inside that is 'apps.myTraining.clientlibs'. This folder contains several sub-folders: 'clientlib-base', 'clientlib-dependencies', 'clientlib-grid', and 'clientlib-site'. Each of these sub-folders contains 'css.txt' and 'js.txt' files. The 'clientlib-grid' folder also contains a 'less' sub-folder with a 'css.txt' file. The 'clientlib-site' folder contains 'css' and 'js' sub-folders, each with a 'css.txt' and 'js.txt' file.