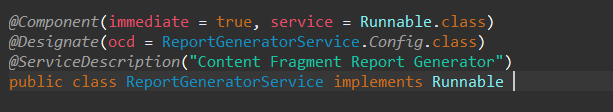
***Scheduler Task By Ayush***

***Requirement:***

***Imagine a scenario where an organization needs to generate daily reports on content fragments. Without this scheduled job, someone would have to manually run the report generation process every day, which is time-consuming and error-prone. By scheduling the job, the process becomes automated, ensuring reports are generated consistently and on time.***

***Case-1:***

******

**# In which scenario we need to implement runnable class .?**

The Runnable interface in Java is typically implemented in scenarios where you need to create a task that can be executed by a thread .

**# I did not understood the thread concept clearly can you exppalin me clearly .?**

**What is a Thread?**

A thread is a single sequence of execution within a program. A Java program can have many threads, which can run concurrently.

**Why Use Threads?**

Threads are useful for performing tasks in parallel, such as:

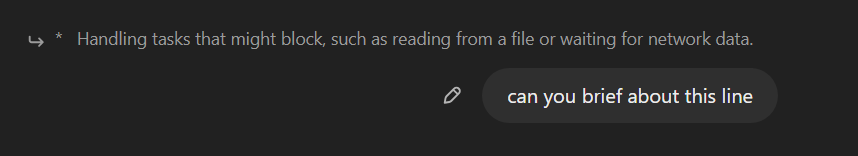
@Handling tasks that might block, such as reading from a file or waiting for network data.

 **Process vs. Thread**:

* **Process**: An independent program running in its own memory space.
* **Thread**: A lightweight sub-process, a smallest unit of processing. Threads share the same memory space within a process.

 **Concurrency**:

* The ability to run multiple threads simultaneously. This can improve performance, especially on multi-core processors.

****

**Blocking Tasks**

**Blocking tasks** are operations that can cause the program to pause its execution until a certain condition is met. Common examples include:

* **Reading from a file**: The program might have to wait until the file is read completely.
* **Waiting for network data**: The program might have to wait for data to be received from a network connection.

**Why Blocking is a Problem**

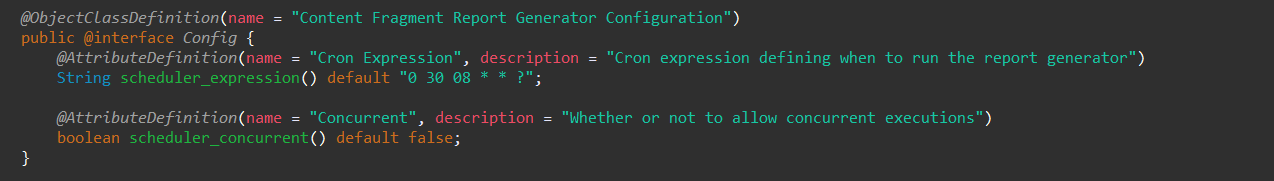
When a task blocks, it can prevent other tasks from executing if they are all running on the same thread. This is especially problematic in applications with a user interface (UI), where a blocking operation can make the UI unresponsive.

**Summary:**

Using threads for blocking tasks such as file I/O or network communication ensures that the main thread remains responsive. This is crucial for maintaining a good user experience, especially in applications with graphical user interfaces or those that require real-time responsiveness. By delegating blocking operations to separate threads, you can keep your application running smoothly while waiting for these potentially time-consuming operations to complete.

# Why we defined OSGI CONFIGURATION here?





*Scheduler task means we will set a particlur time in future using cron expressions, When that time occurs it executes the scheduler task .*

*Here we can set the cron expressions in the backend(in java) or in frontend(In system console in bundle) ,When we set the cron expressions in the backend we need to run the code it is time taking process otherwise if we set the data in the bundle it directly gives the output without need to run the backend code that is why we created the OSGI BUNDLE here* .

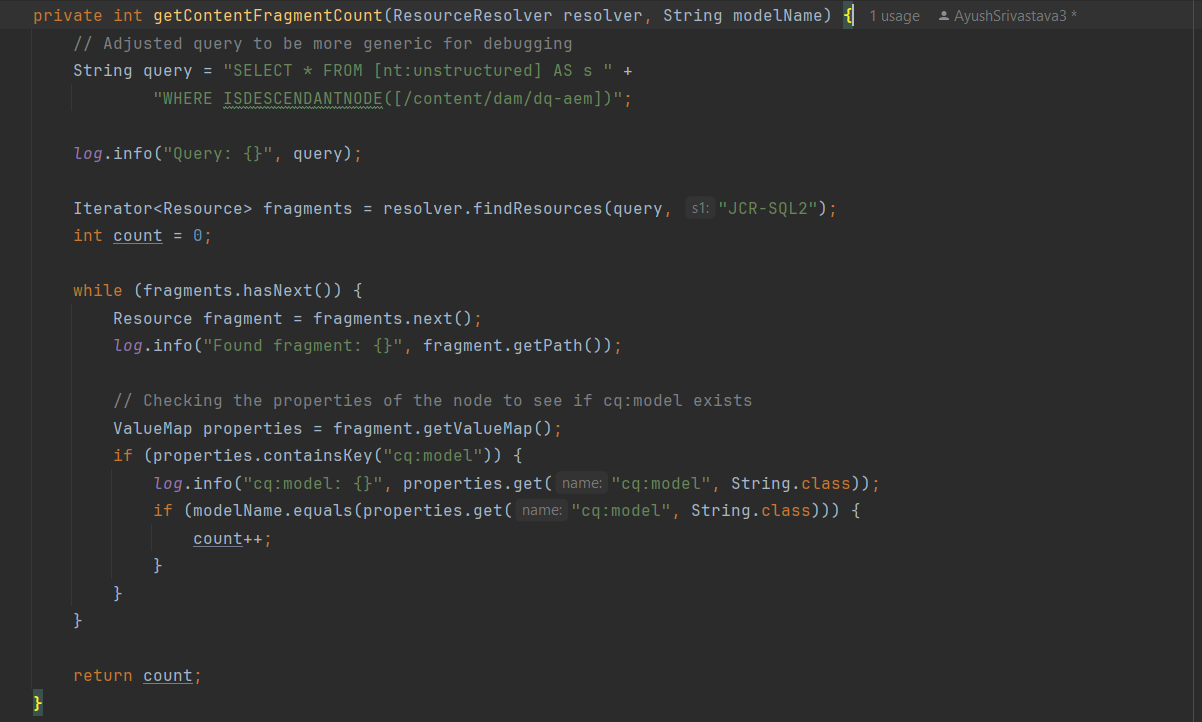
***Case-2:***

*Run() method is like main method in java ,For suppose if we created 5 methods in our class we need call all the classes in run() method.*

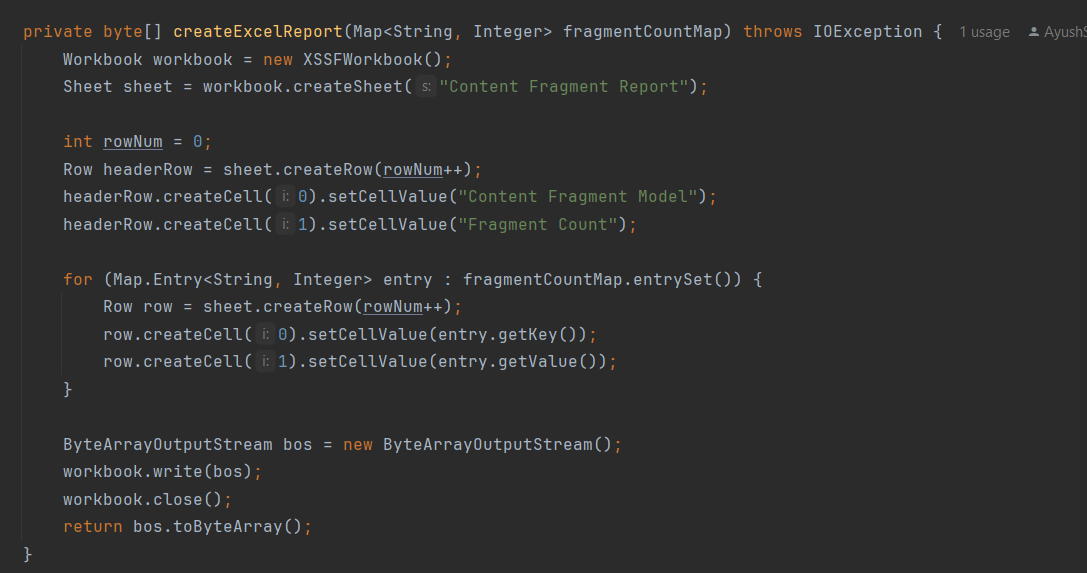
******

***Case-3:***

from this method we are getting all cf names and assests from this path ([/content/dam/dq-aem]) and assests differntiate by cq:model name ,from this method we are getting "cf count" .

******

***Case-4:***

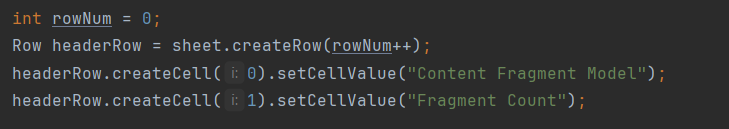
******

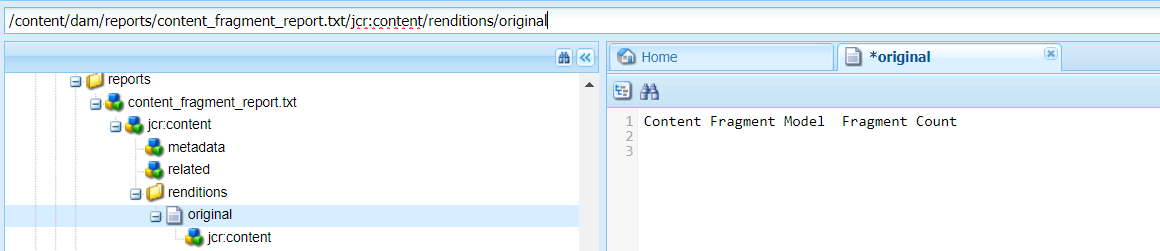
Workbook workbook = new XSSFWorkbook();

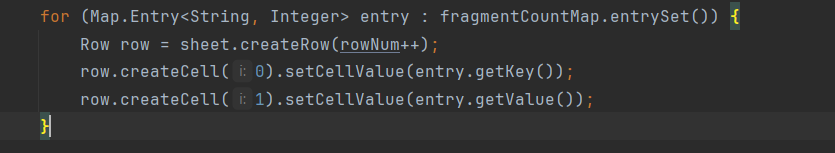
The line of code you've provided initializes a new instance of an Apache POI XSSFWorkbook, which represents an Excel workbook in the .xlsx format.

Sheet sheet = workbook.createSheet("Content Fragment Report");

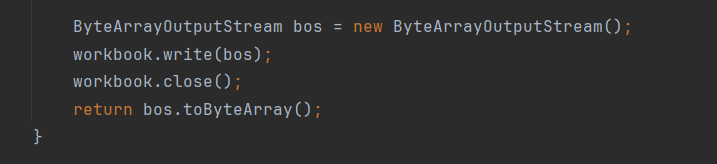
The line of code you've provided creates a new sheet in the workbook and names it "Content Fragment Report"

******

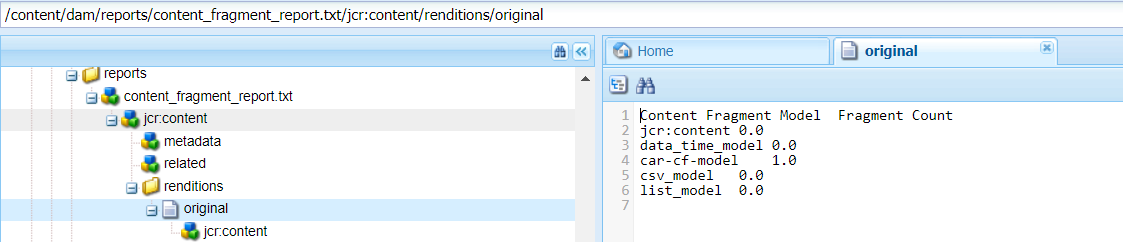
******

******

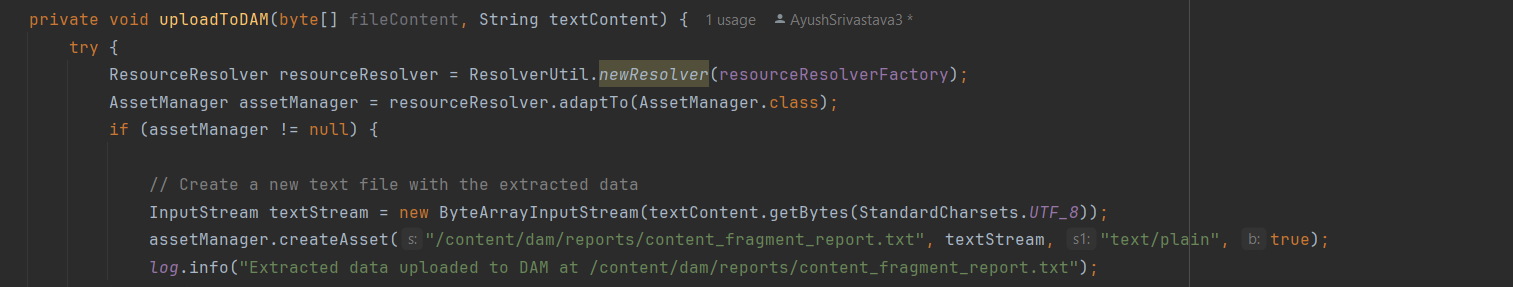
The snippet you provided iterates over a Map named fragmentCountMap and writes its entries to an Excel sheet. Each entry's key and value are written to consecutive cells in the same row. Let's integrate this snippet into a complete example to create an Excel workbook, add a sheet, and populate it with data from the fragmentCountMap.

******

Certainly! Let's break down the process of writing an Apache POI Workbook to a ByteArrayOutputStream and then converting it to a byte array step by step. This technique is useful when you want to generate an Excel file in memory rather than saving it directly to disk .

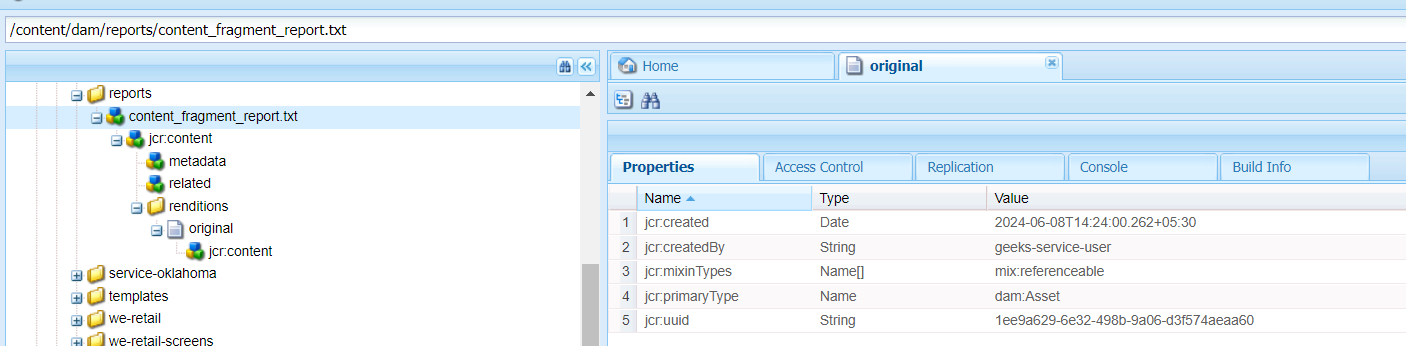
******

***Case-5:***

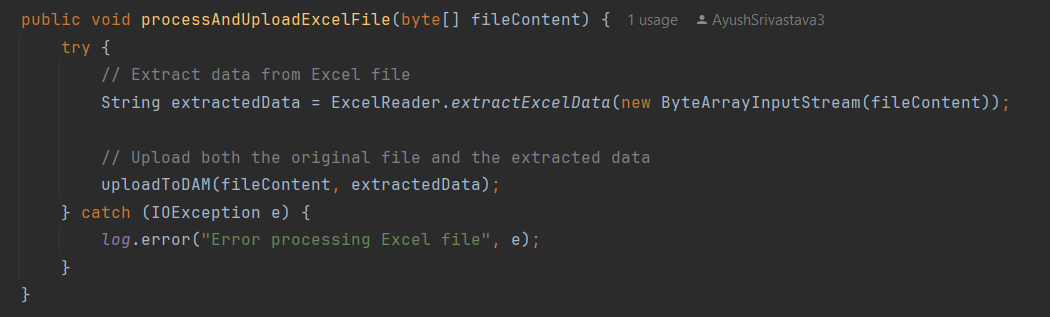
******

***assetManager.createAsset*: This method creates a new asset in the content repository.**

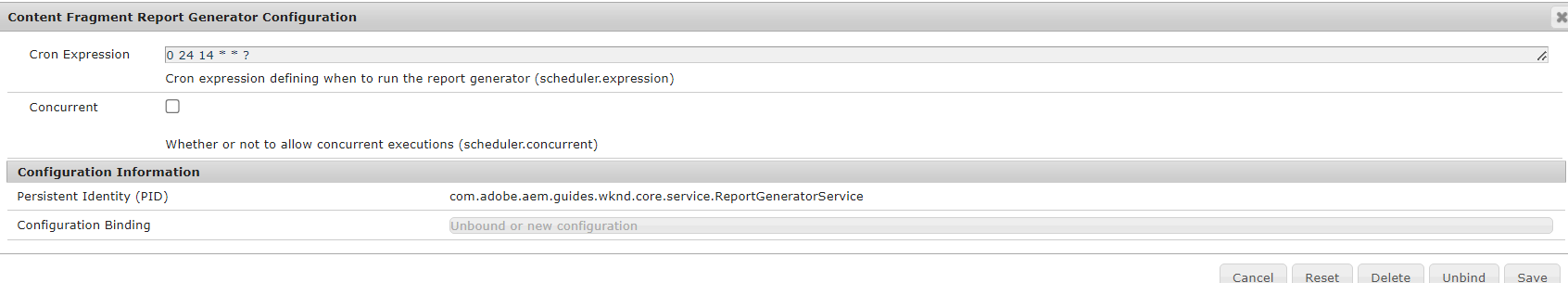
**>"/content/dam/reports/content\_fragment\_report.txt": The path where the new asset will be created.**

****

***Case-6:***

******

***Case-7:***

******

***0 24 14 \* \* ?:***

*0 24(minutes) 14(hours-24 hours time format) \* \* ?*