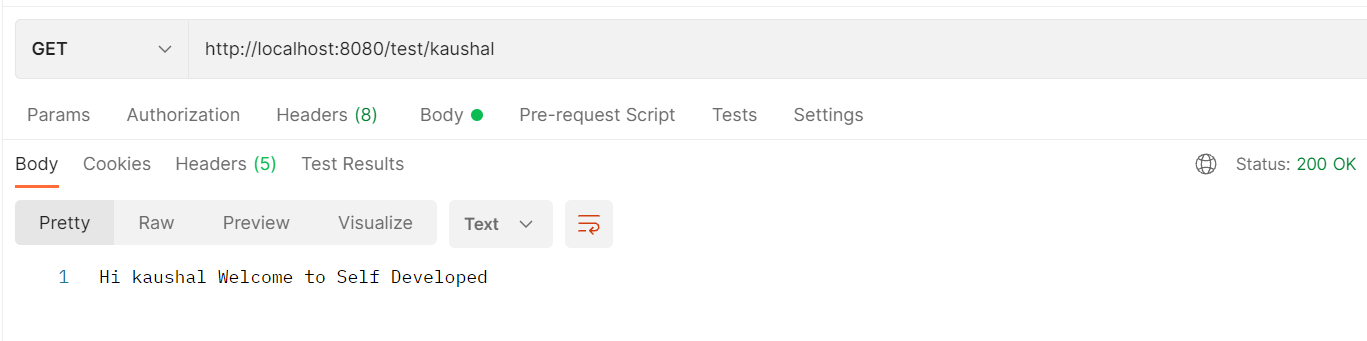
In this tutorial we are going to see how to implement logging mechanism in microservices using spring boot sl4j and how we can customize it using logback. So, let's do a small poc on that so that I will explain you what are the steps we need to follow. So, create a small spring boot application.

App Name- [**spring-boot-logging**](https://github.com/Java-Techie-jt/spring-boot-logging/tree/master)

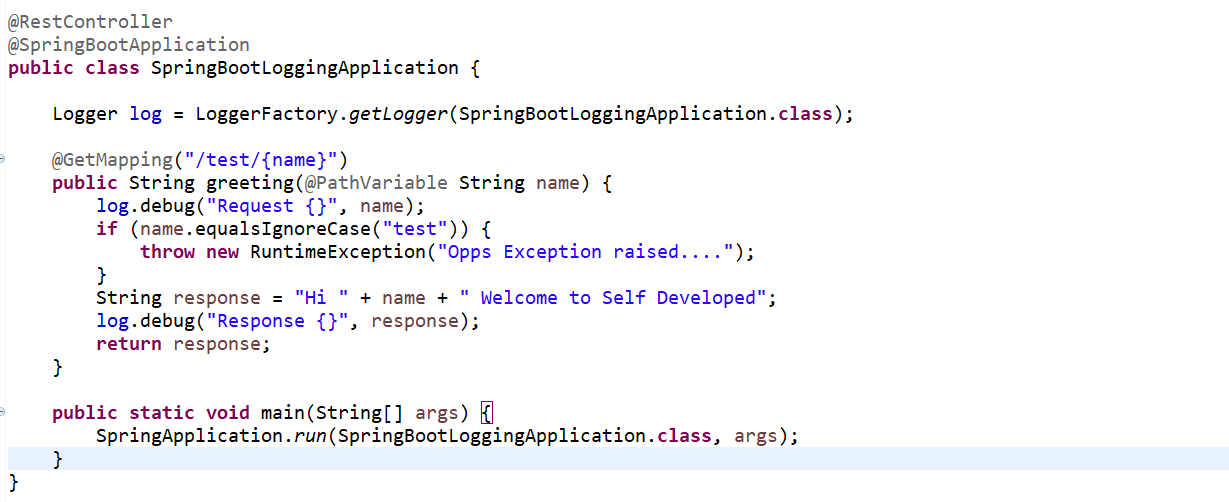
Dependencies – **Spring Boot Devtool, Spring Boot Web**

I want to like differentiate which between this sl4j and log4j. In sl4j you can mark the syntax like logger.debug I am giving the parameter syntax and I am passing the value but in case of log4j you need to write kind of string concatenation so in this sl4j there we no need to use the concatenation so we can save the memory here, so this is one of the difference between log4j and sl4j.

**GET** - <http://localhost:8080/test/kaushal>



------------------------------------------------------------------------------------------------------------------------------------------

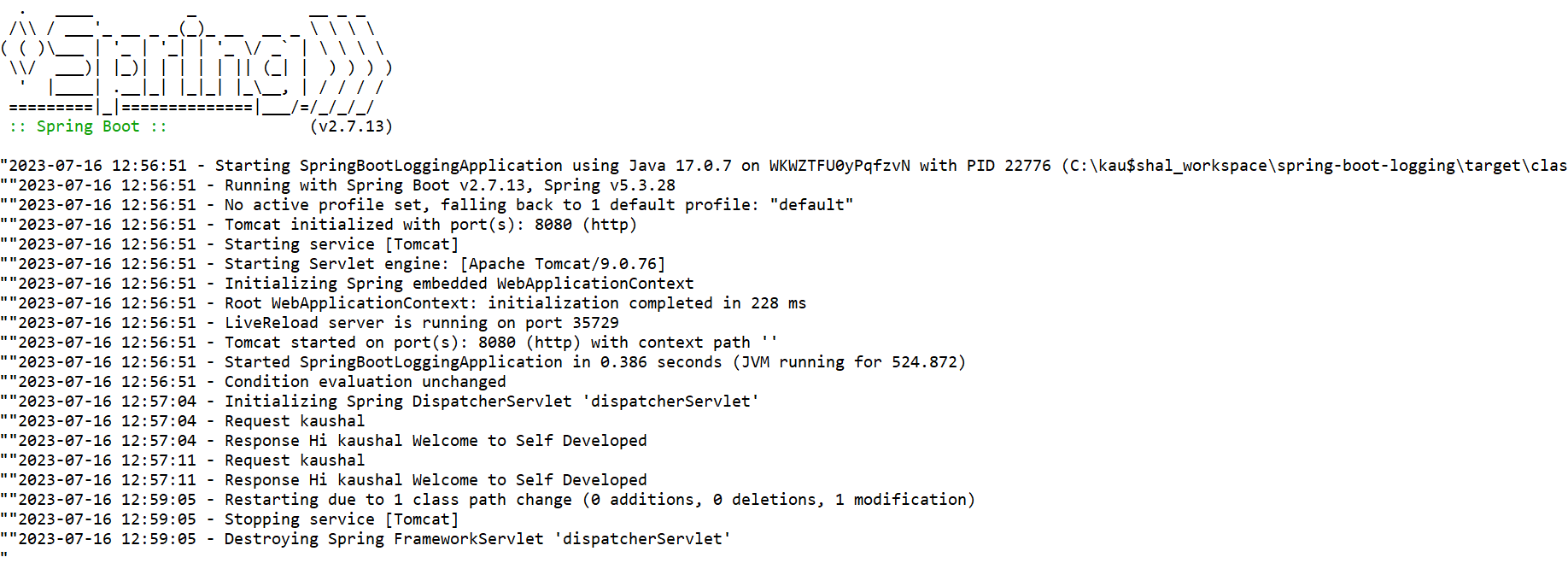


------------------------------------------------------------------------------------------------------------------------------------------

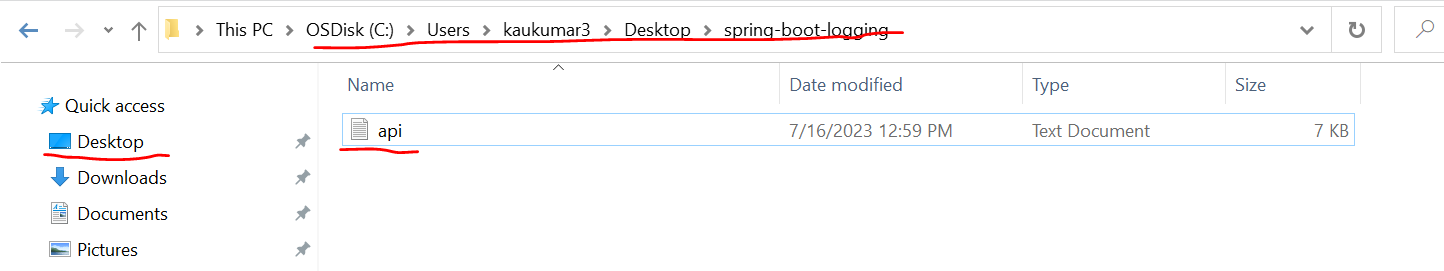
Now let's configure the logging related things so let's configure it in application.properties file. so here this is the predefined key provided by the spring boot logging level s7 application if you remove all the there is no issue and logging level like for spring framework web and here you need to specify your root package here we have the root package let's copy this one let's copy and paste here the root package ok and the logging pattern this is the console appender with this is the format the date and time format and this is the logging pattern for logging appender for file and I am giving the path to generate a file so let's change this spring boot logging so it will internally create a folder with name spring-boot-logging.log file also in console also it's generating the logging statement as we added two appender one is console another one is file.

**application.properties**

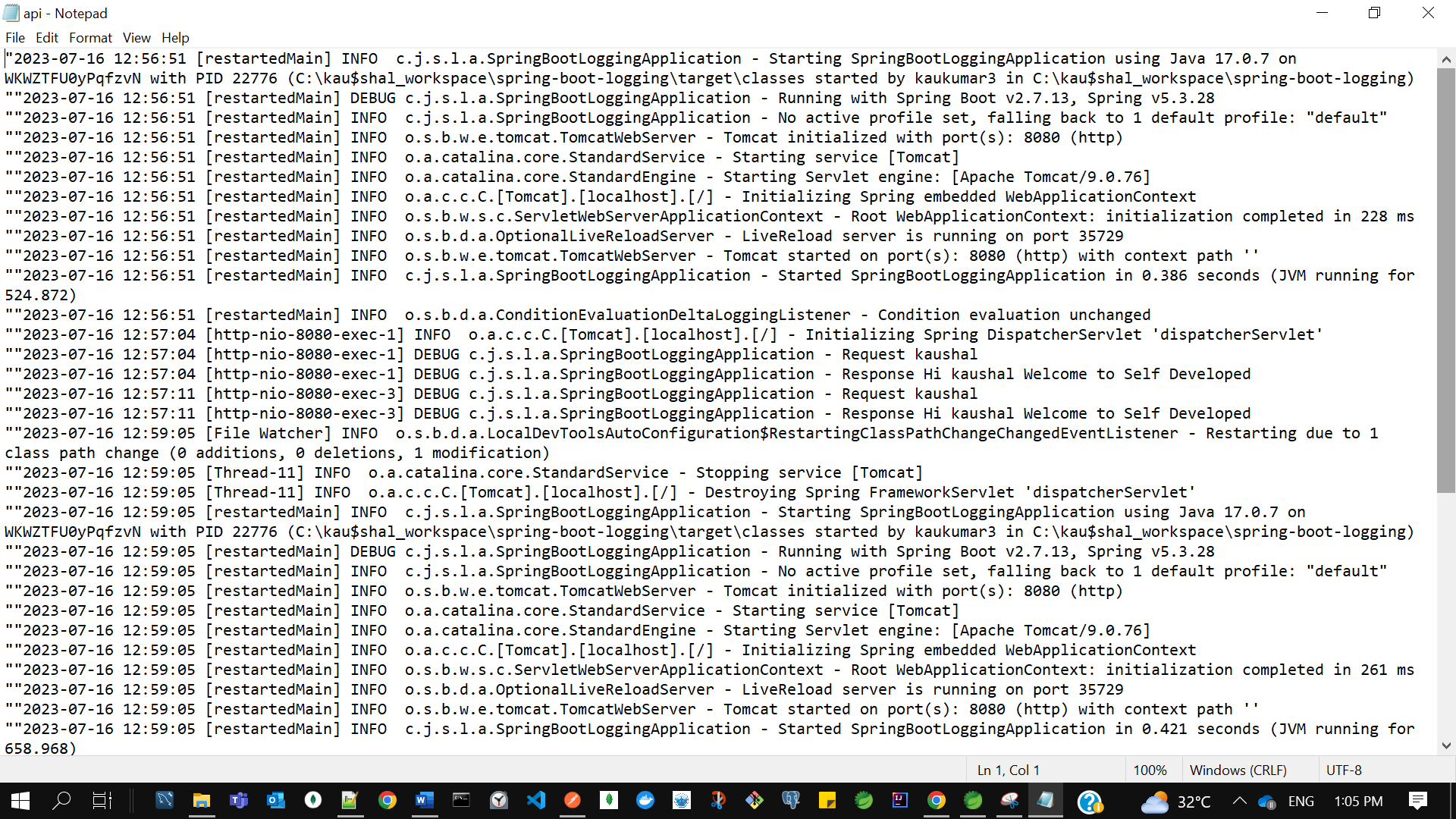




------------------------------------------------------------------------------------------------------------------------------------------



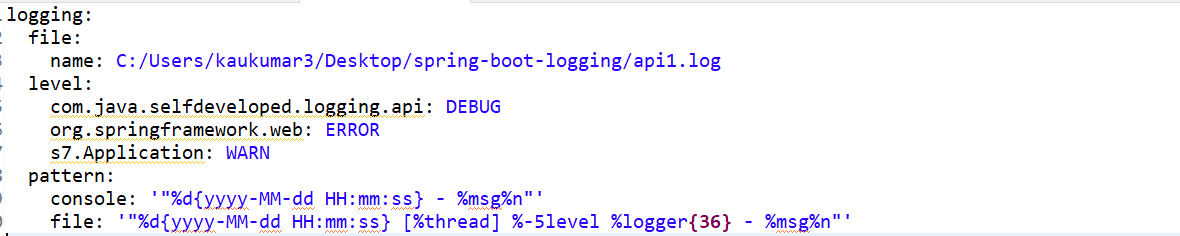
------------------------------------------------------------------------------------------------------------------------------------------



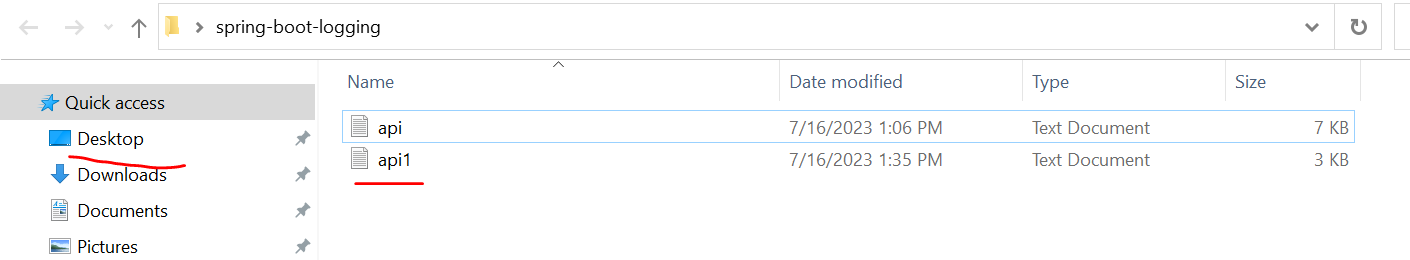
------------------------------------------------------------------------------------------------------------------------------------------

There is another way we can write this statement using the yml file this is using the application.properties file. now what we mention in this application.properties file same we will write in yml file in hierarchy manner so write logging.label.

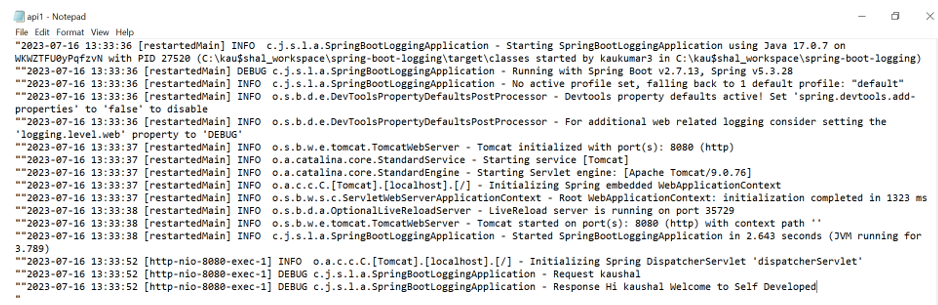
**application.yml**



------------------------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------------------------

so these are the features provided by spring boot default to implement the logging now we will have a customize our logging mechanism using the log back. so I will show you here for error there I want to generate the separate log file and for like if there is no error and we are getting the 200 status code like the response is perfect then I want to generate the one more log file ok so create a XML file here we need to mention the same name log back then only spring boot will pick this file mention .xml, logback.xml, also we need to add the console appender and file appender and one more appender will add for error. the property I mention with this folder this is the root folder with logs there we will get the our log file and this is the console appender and this layout is provided by log back you can check the input statement here and this is the pattern the way I want to print my log statement.

In second one more appender that is the file appender in this format it will print on console and in this pattern it will print on the file section and I am specifying the time base rule policy like the size of my log file and the location the name of file with the date format and this is for again one more file appender for error if there is error or exception in my application error.log one log file will be generate with this is the pattern and for that also I mention the policy like the size and like the folder name and finally in my application I refer both the appender so change this package name to our root package "**com.java.selfdeveloped.logging.api**".

**logback.xml**

**<?xml version="1.0" encoding="UTF-8"?>**

**<configuration>**

**<property name="DEV\_HOME" value="logs" />**

**<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">**

**<layout class="ch.qos.logback.classic.PatternLayout">**

**<Pattern>**

**%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} -**

**%msg%n**

**</Pattern>**

**</layout>**

**</appender>**

**<appender name="FILE-AUDIT"**

**class="ch.qos.logback.core.rolling.RollingFileAppender">**

**<file>${DEV\_HOME}/debug.log</file>**

**<encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">**

**<Pattern>**

**%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} -**

**%msg%n**

**</Pattern>**

**</encoder>**

**<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">**

**<!-- rollover daily -->**

**<fileNamePattern>${DEV\_HOME}/archived/debug.%d{yyyy-MM-dd}.%i.log**

**</fileNamePattern>**

**<timeBasedFileNamingAndTriggeringPolicy**

**class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">**

**<maxFileSize>10MB</maxFileSize>**

**</timeBasedFileNamingAndTriggeringPolicy>**

**</rollingPolicy>**

**</appender>**

**<appender name="FILE-ERROR"**

**class="ch.qos.logback.core.rolling.RollingFileAppender">**

**<file>${DEV\_HOME}/error.log</file>**

**<encoder class="ch.qos.logback.classic.encoder.PatternLayoutEncoder">**

**<Pattern>**

**%d{yyyy-MM-dd HH:mm:ss} [%thread] %-5level %logger{36} -**

**%msg%n**

**</Pattern>**

**</encoder>**

**<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">**

**<!-- rollover daily -->**

**<fileNamePattern>${DEV\_HOME}/archived/error.%d{yyyy-MM-dd}.%i.log**

**</fileNamePattern>**

**<timeBasedFileNamingAndTriggeringPolicy**

**class="ch.qos.logback.core.rolling.SizeAndTimeBasedFNATP">**

**<maxFileSize>10MB</maxFileSize>**

**</timeBasedFileNamingAndTriggeringPolicy>**

**</rollingPolicy>**

**</appender>**

**<!-- Send logs to both console and file audit -->**

**<logger name="com.java.selfdeveloped.logging.api" level="debug"**

**additivity="false">**

**<appender-ref ref="FILE-AUDIT" />**

**<appender-ref ref="STDOUT" />**

**</logger>**

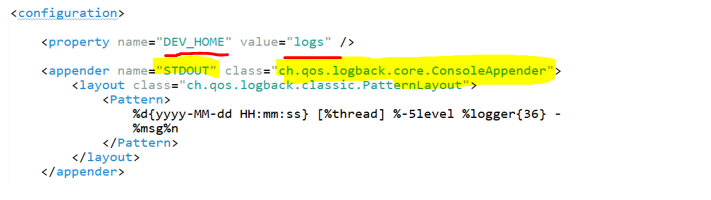
**<root level="error">**

**<appender-ref ref="FILE-ERROR" />**

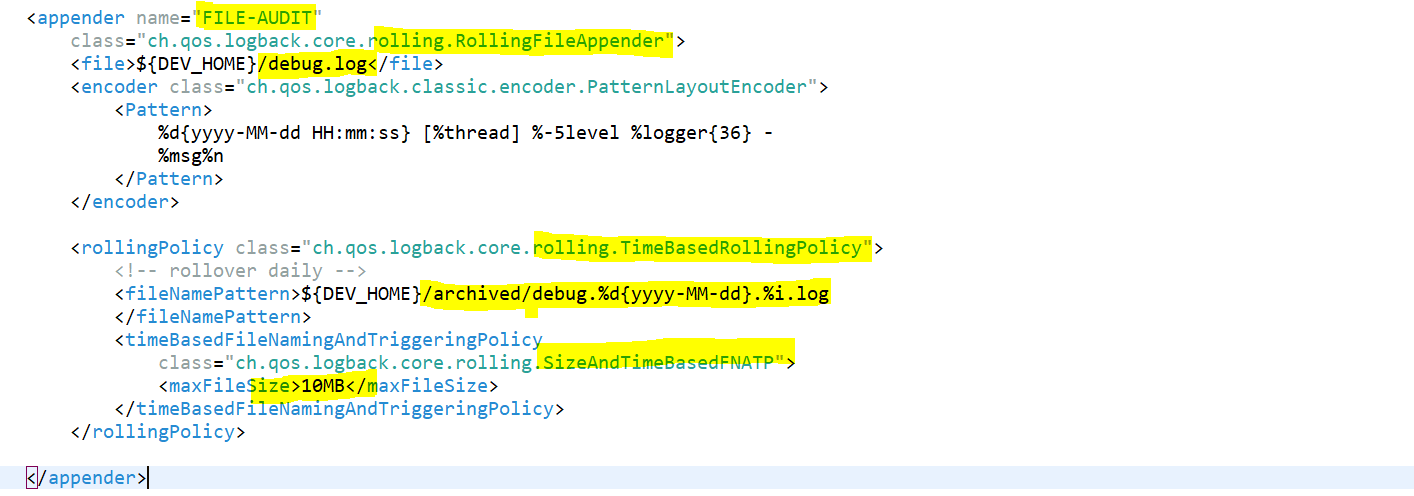
**</root>**

**</configuration>**

------------------------------------------------------------------------------------------------------------------------------------------



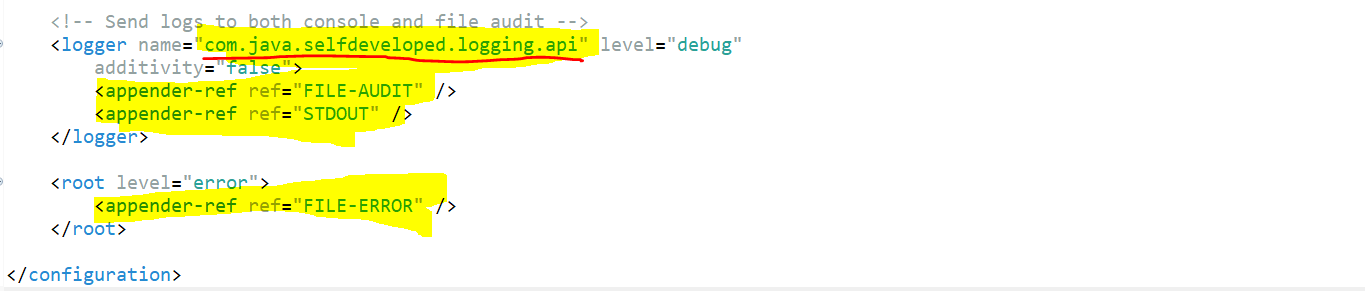
------------------------------------------------------------------------------------------------------------------------------------------



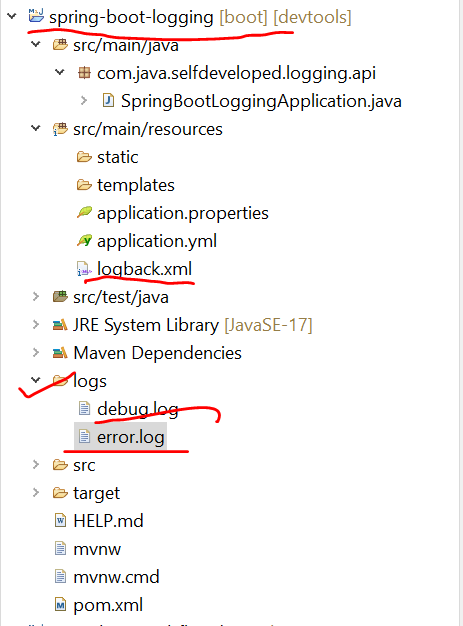
------------------------------------------------------------------------------------------------------------------------------------------



------------------------------------------------------------------------------------------------------------------------------------------

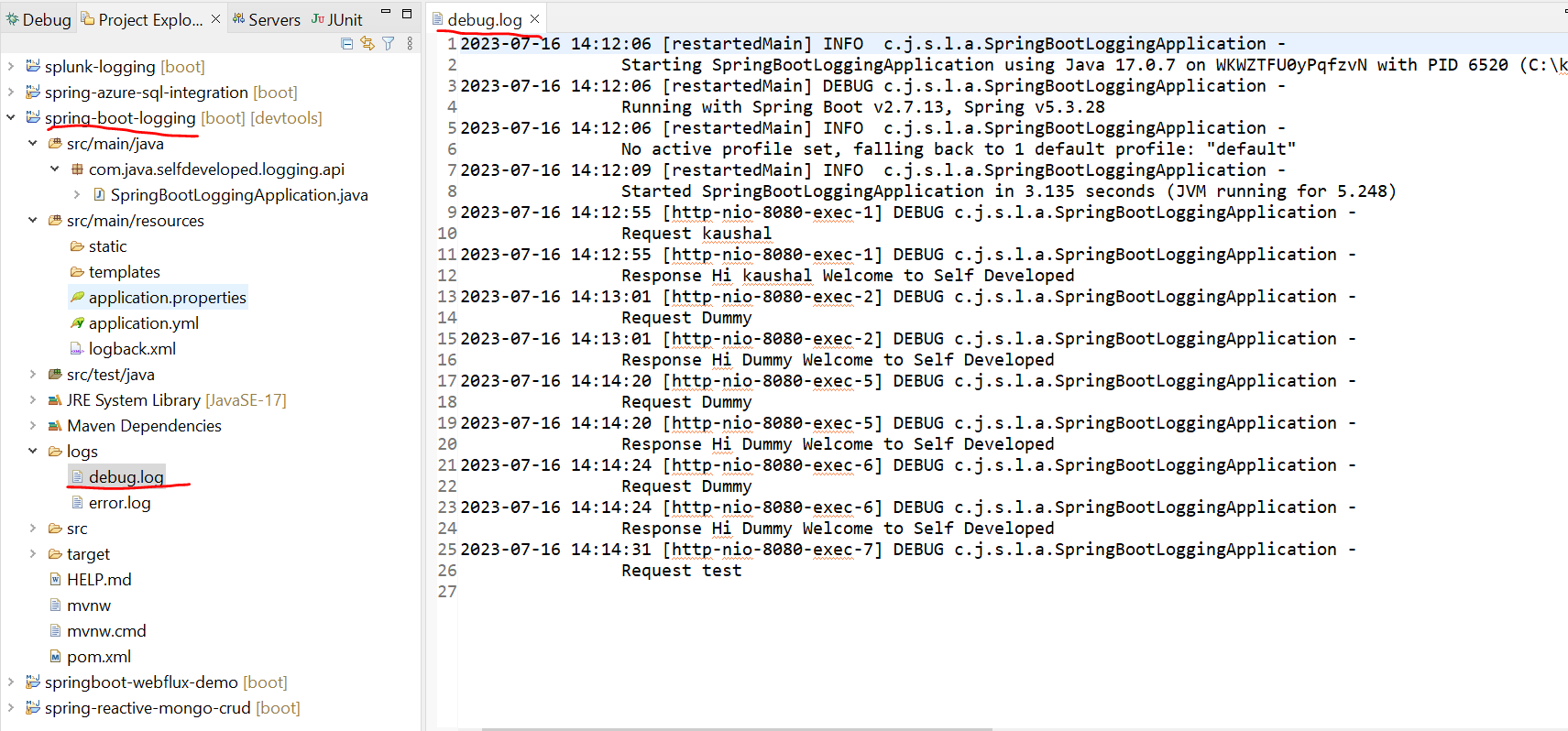


------------------------------------------------------------------------------------------------------------------------------------------



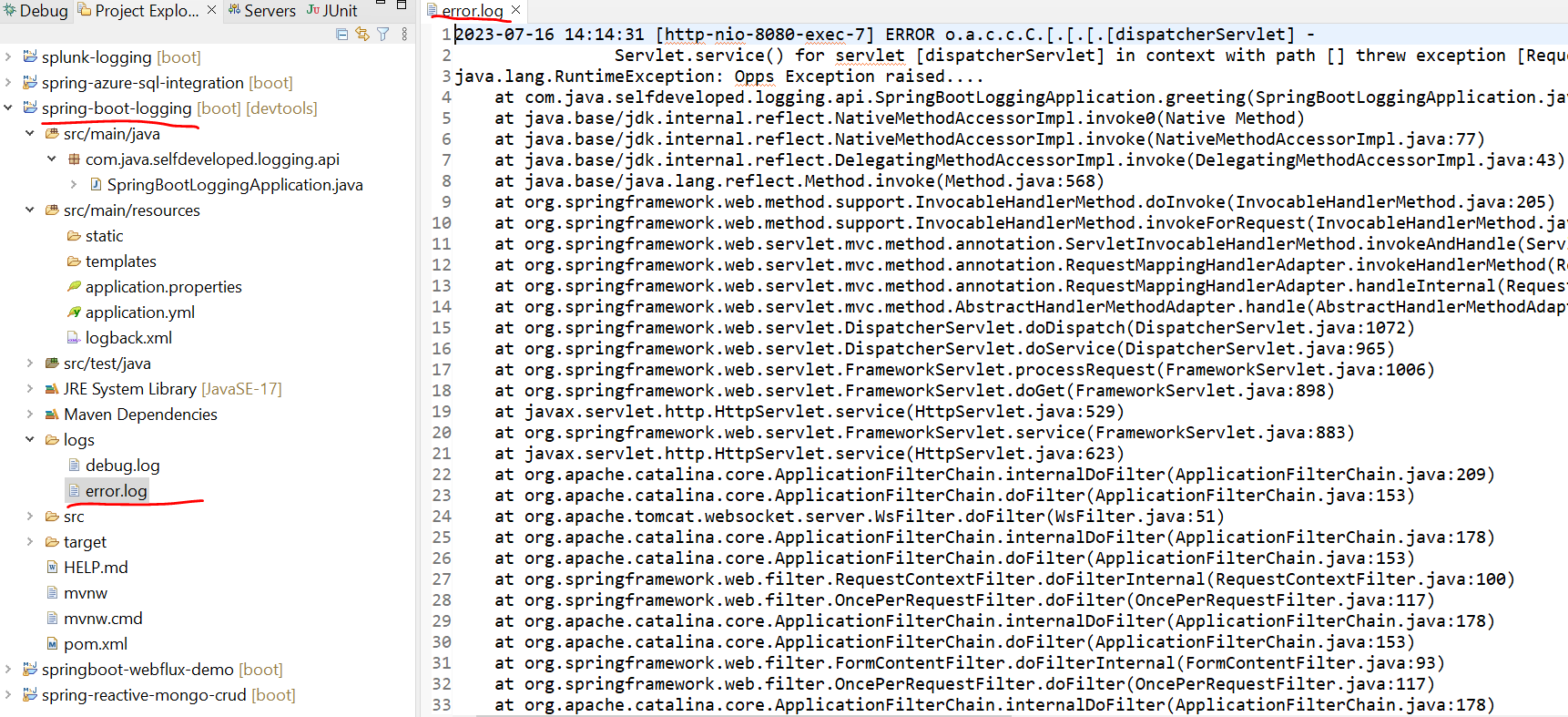
------------------------------------------------------------------------------------------------------------------------------------------

**debug.log**



------------------------------------------------------------------------------------------------------------------------------------------

**error.log**



there is the debug.log and error.log so this we customized using the log back so for that we didn't add any additional dependency as we used sl4j this mechanism is provided by sl4j itself so this is the way we can implement the logging mechanism and we can customize the logging in our microservice