**LOGGING:**

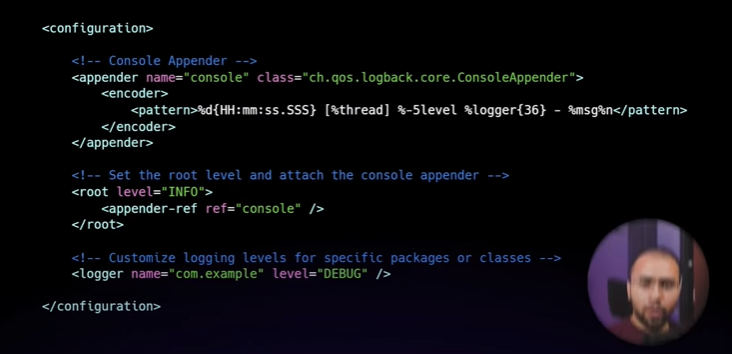
1. Logback: A popular logging framework that serves as the default in many Spring Boot applications. It offers a flexible configuration and good performance.
2. Log4j2: Another widely used logging framework with features such as asynchronous logging and support for various output formats.
3. Java Util Logging (JUL): The default logging framework include in the Java Standard Edition. While it’s less feature-rich than some third-party frameworks, it is straightforward and is part of the Java platform.

Spring Boot comes with a default logging configuration that uses ***Logback as the default logging implementation***. It provides a good balance between simplicity and flexibility.

The default configuration is embedded within the Spring Boot libraries, and it may not be visible in your project’s source code.

If you want to customize the logging configuration, you can create your own logback-spring.xml or ***logback.xml*** file in the ***src/main/resources*** directory. When Spring Boot detects this file in your project, it will use it instead of the default configuration.

**MAJORLY USED = Logback**



**Logging Levels:**

Logging levels help in categorize log statements based on their severity. The common logging levels are :

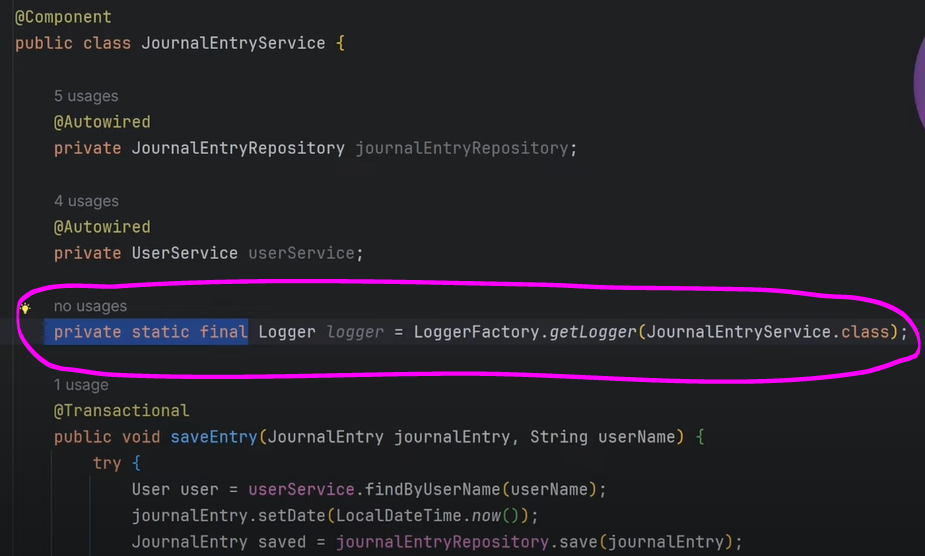
* **TRACE**
* **DEBUG**
* **INFO**
* **WARN**
* **ERROR**

We can set the desired logging level for specific packages or classes, allowing them to control the amount of information logged at runtime.

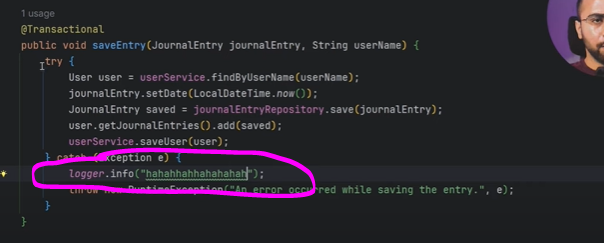
By default logging is enabled for **INFO**, **WARN** and **ERROR**.

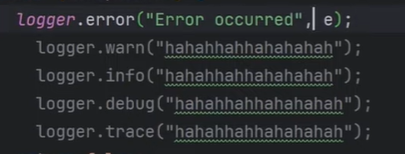
Spring Boot provides annotations like ***@Slf4j & @Log4j2*** that you can use to automatically inject logger instance into your classes.

Example : (use org.slf4j.Logger)



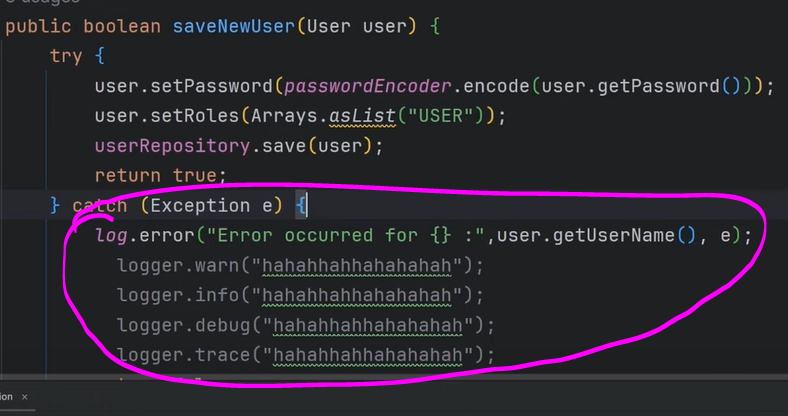
Now instead of wrinting print conditions in between function/methods we cab just write logger.info()/.warn()/.error() etc etc ….

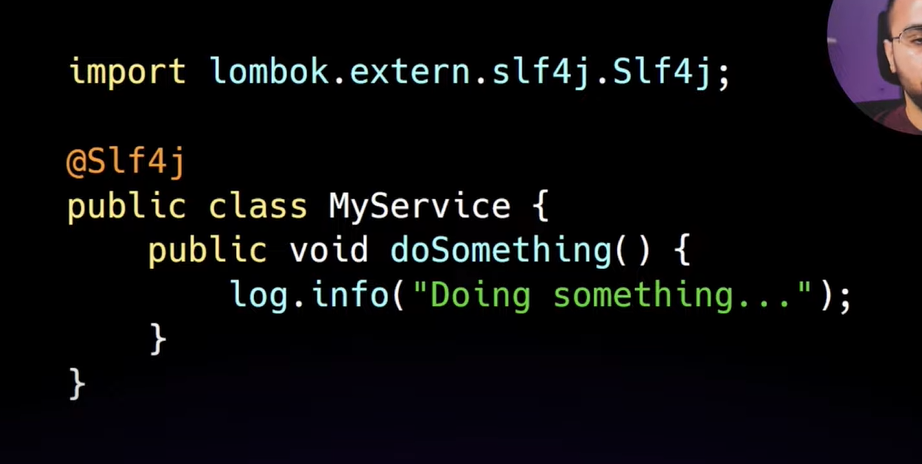




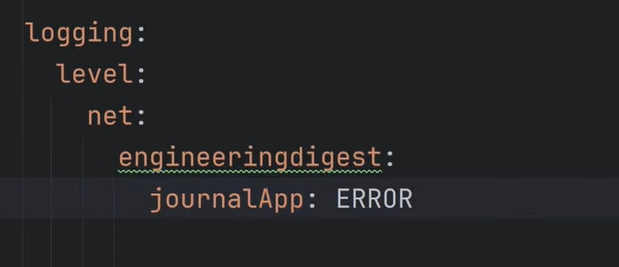
THERES ANOTHER WAY OF WRITING THIS ABOVE WITH THE HELP OF LOMBOF

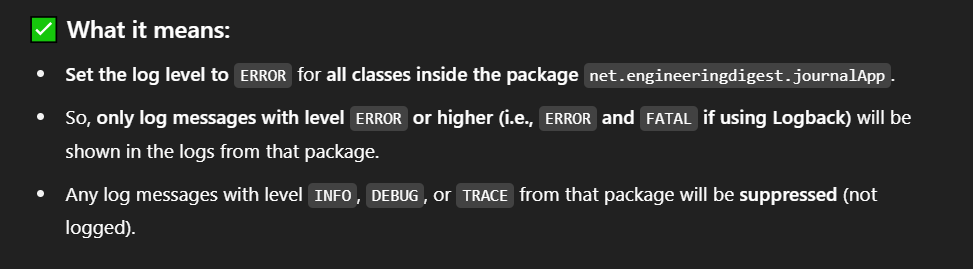
For this we can use @Slf4j annotation and instead of creating a “private final static” for Logger we can directly use log variable !   
For example:



**LOGGING IN RESOURCES APPLICATION.PROPERTIES**







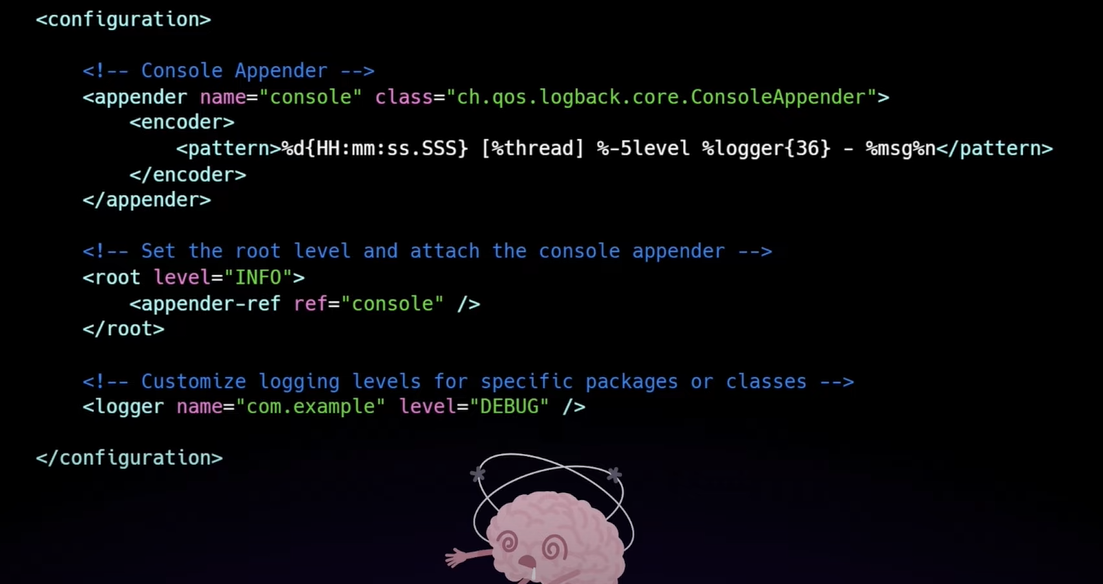
If I only want to include INFO, WARN and ERROR logs of a package.  
  
 

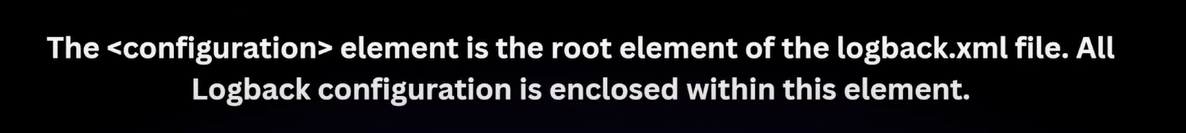
This will stop all the logs of that package.



This will stop all the logs of that particular class.

**FILE WHICH WILL CONTAIN ALL THE LOGS : XML APPROACH !**





APPENDER : Where do we want to print our logs

TYPES : Many types but the basic are 2. That are ***Console Appender*** and ***File Appender***.(File appender will print logs in a file)