Logging

1. Advantages of logging over s.o.f
   1. Pattern
   2. Levels problem
   3. Console vs multiple files

**Comparison**

* **SLF4J**: A logging abstraction. Used to write code independently of any specific logging implementation.
* **Log4j**: A legacy logging framework that is still widely used. It has been replaced by Log4j 2, which addresses performance and security concerns.
* **Logback**: A modern, faster, and more flexible logging framework, often preferred over Log4j, especially when used with SLF4J.

**When to use what:**

* **SLF4J + Logback** is a common and powerful combination for logging in modern Java applications.
* **Log4j 2** is still a valid alternative if you prefer or have legacy systems using Log4j.

1. Log frameworks / libraries & Slf4j - Diagram
   1. <https://www.slf4j.org/manual.html>

1. How to create Logger class.
   1. Object & lombok @Slf4j annotation
   2. Show how Sl4fJ is wrapping the internal libraries. Both Logger & LogFactory

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

**private static final** Logger ***logger*** = LoggerFactory.*getLogger*(PaymentController.**class**);

1. Log levels using logger objects
   1. Greater than or equal
   2. Write all logger levels & show that only INFO - global setting
   3. IN property file we can change that|
      1. logging.level.root=DEBUG
      3. logging.level.com.hulkhiretech=TRACE
   4. But flexible for xml

1. Add logback-spring.xml in resources, show that pattern of what was earlier & now changed.. + more control.
   1. <https://drive.google.com/file/d/1hRlzobSyDH6c7V3Q5hE4adKwSY5YF6hx/view?usp=drive_link>

1. ROOT Logger level,
   1. Default for the application
   2. Change & show how logs are changing.

1. Package specific logging
   1. Configure logger: com.hulkhiretech
   2. ROOT as INFO, and application logger as Whatever level, TRACE

1. Explaining log pattern

1. How to set env specific application logger level
   1. In xml, keep it as INFO, and whichever env file you want to change directly configure that
      1. logging.level.com.hulkhiretech=TRACE
   2. Experiment with xml, application.property, local property

1. File & Mobaxterm
   1. File
      1. File path from pom.xml & "@"
      2. Tail command, & show logs coming.
      3. Tail & grep for specific
   2. VI basics
   3. Show rolling behaviour & version.
   4. In project setup:
      1. Daily log enabled
      2. And 7 days keep in machine.
      3. Daily log also write to Kibana.
      4. Maintain in Kibana for 3months.
      5. Put in AWS S3 bucket.

**Micrometer setup for distributed tracking - TraceId & SpanId**

Add XML

[%X{traceId:-}] [%X{spanId:-}]

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-actuator</artifactId>

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing</artifactId>

</dependency>

<dependency>

<groupId>io.micrometer</groupId>

<artifactId>micrometer-tracing-bridge-brave</artifactId>

</dependency>

management.tracing.enabled=true

management.tracing.sampling.probability=1.0