**Various Log Levels in SLF4J & Log4J**

**Various Log Levels in SLF4J & Log4J**

Log levels help control the verbosity of logs, allowing developers to filter logs based on their importance. The standard log levels in **SLF4J (used with Log4J, Logback, etc.)** are:

1. **TRACE** – Most detailed level, used for debugging very fine-grained details.

logger.trace("This is a TRACE message");

1. **DEBUG** – Less verbose than TRACE, useful for debugging during development.

logger.debug("This is a DEBUG message");

1. **INFO** – General informational messages about application execution.

logger.info("Application started successfully");

1. **WARN** – Indicates potential issues but not errors.

logger.warn("Memory usage is high, consider optimization");

1. **ERROR** – Indicates serious issues that need attention.

logger.error("Database connection failed", exception);

1. **FATAL (Only in Log4J 1.x)** – Used for critical failures that may cause application shutdown.

logger.fatal("System crash due to disk failure");

*(Note: SLF4J does not support FATAL; it's used in Log4J 1.x but replaced with ERROR in Log4J 2 and Logback.)*

**Configuring Log Levels in Log4J (log4j.properties)**

log4j.rootLogger=INFO, stdout, file

# Console output

log4j.appender.stdout=org.apache.log4j.ConsoleAppender

log4j.appender.stdout.layout=org.apache.log4j.PatternLayout

log4j.appender.stdout.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1} - %m%n

# File output

log4j.appender.file=org.apache.log4j.FileAppender

log4j.appender.file.File=application.log

log4j.appender.file.layout=org.apache.log4j.PatternLayout

log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd HH:mm:ss} %-5p %c{1} - %m%n

* To change the log level dynamically, update the log4j.properties file or use **Kafka dynamic log level settings**:

kafka-configs --bootstrap-server kafka:9092 --alter \

--add-config "kafka.server.ReplicaManager=DEBUG" \

--entity-type broker-logger --entity-name 101