Logging using SLF4J

Exercise 1: Logging Error Messages and Warning Levels

Task: Write a Java application that demonstrates logging error messages and warning levels

using SLF4J.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

2. Create a Java class that uses SLF4J for logging:

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class LoggingExample {

    // Get a logger instance

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

    public static void main(String[] args) {

        // Log an error message

        logger.error("This is an error message");

        // Log a warning message

        logger.warn("This is a warning message");

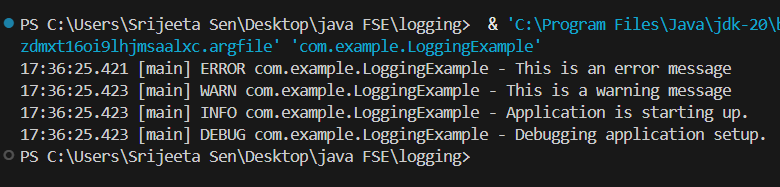
        // Optional: info and debug messages

        logger.info("Application is starting up.");

        logger.debug("Debugging application setup.");

    }

}



Exercise 2: Parameterized Logging

Task: Write a Java application that demonstrates parameterized logging using SLF4J.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

2. Create a Java class that uses SLF4J for parameterized logging:

Write code for this.

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class ParameterizedLoggingExample {

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

    public static void main(String[] args) {

        // Some example variables to log

        String username = "Srijeeta";

        int age = 21;

        double balance = 1050.75;

        // Parameterized logging using {}

        logger.info("User {} is {} years old and has a balance of ${}", username, age, balance);

        // Logging with a warning message

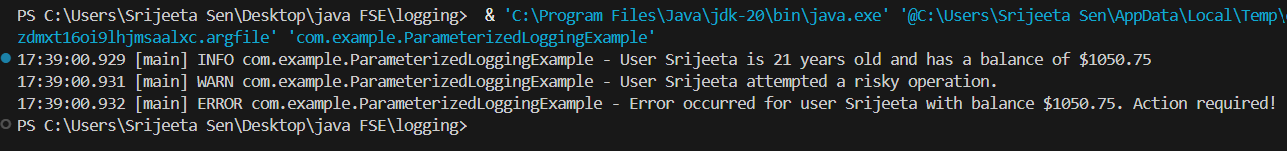
        logger.warn("User {} attempted a risky operation.", username);

        // Logging an error message with parameters

        logger.error("Error occurred for user {} with balance ${}. Action required!", username, balance);

    }

}



Exercise 3: Using Different Appenders

Task: Write a Java application that demonstrates using different appenders with SLF4J.

Step-by-Step Solution:

1. Add SLF4J and Logback dependencies to your `pom.xml` file:

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.30</version>

</dependency>

<dependency>

<groupId>ch.qos.logback</groupId>

<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

2. Create a `logback.xml` configuration file to define different appenders:

<configuration>

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

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<appender name="file" class="ch.qos.logback.core.FileAppender">

<file>app.log</file>

<encoder>

<pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</pattern>

</encoder>

</appender>

<root level="debug">

<appender-ref ref="console" />

<appender-ref ref="file" />

</root>

</configuration>

3. Create a Java class that uses SLF4J for logging:

Write code for this.

package com.example;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class MultiAppenderExample {

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

    public static void main(String[] args) {

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

        logger.info("This is an INFO message (normal program status).");

        logger.warn("This is a WARN message (something could be wrong).");

        logger.error("This is an ERROR message (something failed!).");

        // Test with variables

        String username = "Srijeeta";

        logger.info("User {} has logged in successfully.", username);

    }

}

