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>

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

import org.slf4j.Logger;

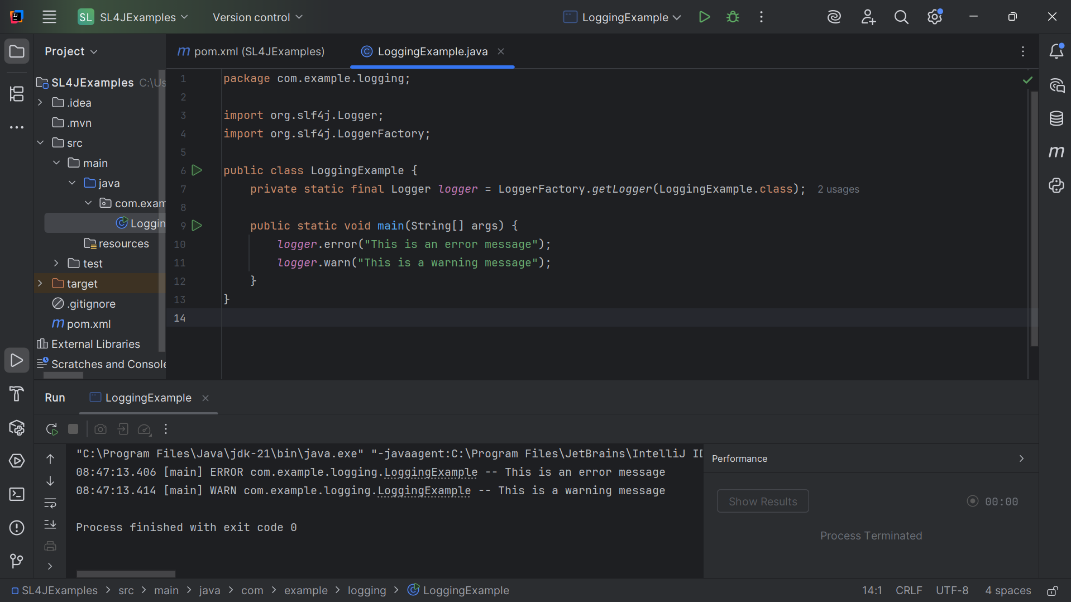
import org.slf4j.LoggerFactory;

public class LoggingExample {

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

public static void main(String[] args) { logger.error("This is an error message"); logger.warn("This is a warning message");

}

}

# 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>

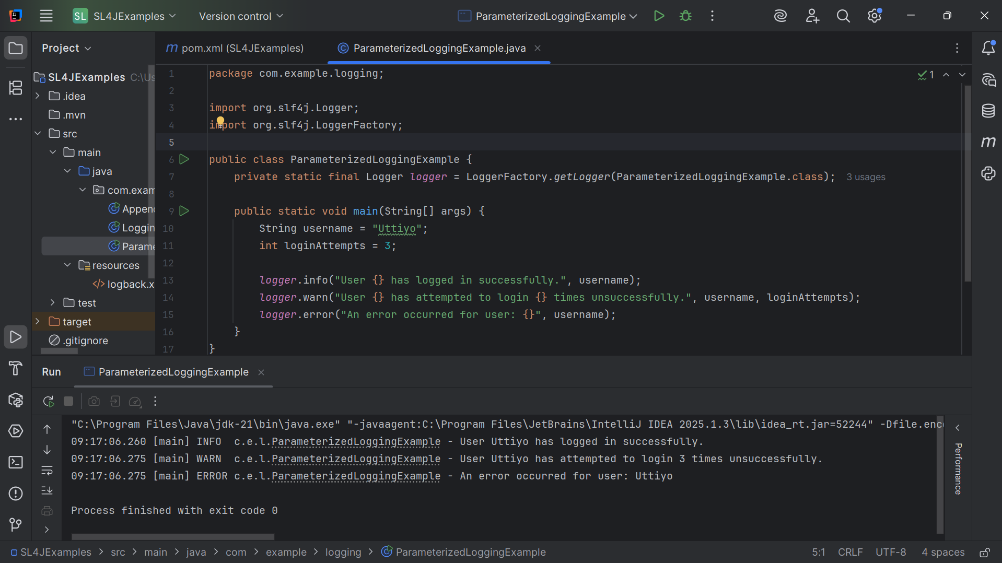
<artifactId>logback-classic</artifactId>

<version>1.2.3</version>

</dependency>

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

Write code for this.

**package com.example.logging;  
  
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) {  
 String username = "Uttiyo";  
 int loginAttempts = 3;  
  
 *logger*.info("User {} has logged in successfully.", username);  
 *logger*.warn("User {} has attempted to login {} times unsuccessfully.", username, loginAttempts);  
 *logger*.error("An error occurred for user: {}", username);  
 }  
}**

# 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>

1. 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>

1. Create a Java class that uses SLF4J for logging: Write code for this.

**package com.example.logging;  
  
import org.slf4j.Logger;  
import org.slf4j.LoggerFactory;  
  
public class AppenderExample {  
 private static final Logger *logger* = LoggerFactory.*getLogger*(AppenderExample.class);  
  
 public static void main(String[] args) {  
 *logger*.info("This message should go to both console and file.");  
 *logger*.debug("Debug information for troubleshooting.");  
 *logger*.error("Error encountered during process.");  
 }  
}**

