SLF4J logging

Exercise 2: Parameterized Logging

java

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 user = "admin";

int loginAttempts = 3;

String ipAddress = "192.168.1.100";

*// Basic parameterized logging*

logger.info("User {} logged in from {}", user, ipAddress);

*// With multiple parameters*

logger.warn("User {} has {} failed login attempts", user, loginAttempts);

*// With exception*

try {

*// Simulate an error*

throw new RuntimeException("Database connection failed");

} catch (Exception e) {

logger.error("Error occurred for user {}: {}", user, e.getMessage(), e);

}

*// Conditional logging*

if (logger.isDebugEnabled()) {

logger.debug("Detailed debug information for user {}", user);

}

}

}

Exercise 3: Using Different Appenders

java

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) {

*// These messages will go to both console and file as configured in logback.xml*

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

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

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

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

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

*// Example with variables and exception*

try {

int result = 10 / 0;

} catch (Exception e) {

logger.error("Division by zero error occurred", e);

}

*// Logging with markers (if needed)*

*// Marker importantMarker = MarkerFactory.getMarker("IMPORTANT");*

*// logger.info(importantMarker, "This is an important message");*

}

}