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: org.slf4j slf4j-api 1.7.30 ch.qos.logback logback-classic 1.2.3 2. 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"); } }

// pom.xml - Maven Dependencies

/\*

<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

http://maven.apache.org/xsd/maven-4.0.0.xsd">

<modelVersion>4.0.0</modelVersion>

<groupId>com.example</groupId>

<artifactId>slf4j-logging-demo</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<maven.compiler.source>11</maven.compiler.source>

<maven.compiler.target>11</maven.compiler.target>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<!-- SLF4J API -->

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>1.7.36</version>

</dependency>

<!-- Logback Classic (SLF4J Implementation) -->

<dependency>

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

<artifactId>logback-classic</artifactId>

<version>1.2.12</version>

</dependency>

<!-- Logback Core -->

<dependency>

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

<artifactId>logback-core</artifactId>

<version>1.2.12</version>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.8.1</version>

<configuration>

<source>11</source>

<target>11</target>

</configuration>

</plugin>

</plugins>

</build>

</project>

\*/

// logback.xml - Logging Configuration (place in src/main/resources/)

/\*

<?xml version="1.0" encoding="UTF-8"?>

<configuration>

<!-- Console Appender -->

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

<encoder>

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

</encoder>

</appender>

<!-- File Appender -->

<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>logs/application.log</file>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>logs/application.%d{yyyy-MM-dd}.%i.log</fileNamePattern>

<maxFileSize>10MB</maxFileSize>

<maxHistory>30</maxHistory>

<totalSizeCap>3GB</totalSizeCap>

</rollingPolicy>

<encoder>

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

</encoder>

</appender>

<!-- Error File Appender -->

<appender name="ERROR\_FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">

<file>logs/error.log</file>

<filter class="ch.qos.logback.classic.filter.LevelFilter">

<level>ERROR</level>

<onMatch>ACCEPT</onMatch>

<onMismatch>DENY</onMismatch>

</filter>

<rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">

<fileNamePattern>logs/error.%d{yyyy-MM-dd}.log</fileNamePattern>

<maxHistory>30</maxHistory>

</rollingPolicy>

<encoder>

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

</encoder>

</appender>

<!-- Root Logger -->

<root level="INFO">

<appender-ref ref="CONSOLE" />

<appender-ref ref="FILE" />

<appender-ref ref="ERROR\_FILE" />

</root>

<!-- Package-specific Loggers -->

<logger name="com.example.service" level="DEBUG" additivity="false">

<appender-ref ref="CONSOLE" />

<appender-ref ref="FILE" />

</logger>

<logger name="com.example.database" level="WARN" additivity="false">

<appender-ref ref="CONSOLE" />

<appender-ref ref="FILE" />

</logger>

</configuration>

\*/

// 1. LoggingExample.java - Basic Logging Demonstration

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.slf4j.MDC;

public class LoggingExample {

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

public static void main(String[] args) {

logger.info("🚀 Starting SLF4J Logging Demonstration");

// Basic logging levels

demonstrateBasicLogging();

// Parameterized logging

demonstrateParameterizedLogging();

// Exception logging

demonstrateExceptionLogging();

// Conditional logging

demonstrateConditionalLogging();

// MDC (Mapped Diagnostic Context)

demonstrateMDC();

logger.info("✅ SLF4J Logging Demonstration completed");

}

public static void demonstrateBasicLogging() {

logger.info("=== Basic Logging Levels ===");

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

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

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

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

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

System.out.println(); // Add spacing for readability

}

public static void demonstrateParameterizedLogging() {

logger.info("=== Parameterized Logging ===");

String userName = "John Doe";

int userId = 12345;

double balance = 1250.75;

// Efficient parameterized logging (no string concatenation)

logger.info("User {} with ID {} has balance ${}", userName, userId, balance);

logger.warn("User {} attempted to withdraw ${} but only has ${}",

userName, 2000.0, balance);

logger.error("Failed login attempt for user {} from IP {}",

userName, "192.168.1.100");

System.out.println();

}

public static void demonstrateExceptionLogging() {

logger.info("=== Exception Logging ===");

try {

// Simulate an exception

int result = 10 / 0;

} catch (ArithmeticException e) {

logger.error("Arithmetic error occurred: {}", e.getMessage(), e);

}

try {

// Simulate another exception

String text = null;

int length = text.length();

} catch (NullPointerException e) {

logger.error("Null pointer exception in text processing", e);

}

// Logging with additional context

try {

processUser("invalid-user-id");

} catch (Exception e) {

logger.error("Failed to process user with context: operation=processUser, " +

"timestamp={}, error={}",

java.time.Instant.now(), e.getMessage(), e);

}

System.out.println();

}

public static void demonstrateConditionalLogging() {

logger.info("=== Conditional Logging ===");

// Check if debug is enabled before expensive operations

if (logger.isDebugEnabled()) {

String expensiveDebugInfo = generateExpensiveDebugInfo();

logger.debug("Debug info: {}", expensiveDebugInfo);

}

// Check other levels

if (logger.isTraceEnabled()) {

logger.trace("Trace level is enabled");

}

if (logger.isWarnEnabled()) {

logger.warn("Warning level is enabled");

}

System.out.println();

}

public static void demonstrateMDC() {

logger.info("=== MDC (Mapped Diagnostic Context) ===");

// Set context information

MDC.put("userId", "12345");

MDC.put("sessionId", "abc-def-123");

MDC.put("operation", "payment");

logger.info("Processing payment request");

logger.warn("Payment amount exceeds daily limit");

logger.error("Payment processing failed");

// Clear context

MDC.clear();

logger.info("Context cleared - no MDC info");

System.out.println();

}

private static void processUser(String userId) {

throw new IllegalArgumentException("Invalid user ID: " + userId);

}

private static String generateExpensiveDebugInfo() {

// Simulate expensive operation

StringBuilder sb = new StringBuilder();

for (int i = 0; i < 100; i++) {

sb.append("Debug data ").append(i).append(" ");

}

return sb.toString();

}

}

// 2. UserService.java - Service Class with Logging

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class UserService {

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

public boolean createUser(String username, String email) {

logger.info("Creating user: username={}, email={}", username, email);

try {

// Validate input

if (username == null || username.trim().isEmpty()) {

logger.warn("User creation failed: username is null or empty");

return false;

}

if (email == null || !email.contains("@")) {

logger.warn("User creation failed: invalid email format: {}", email);

return false;

}

// Simulate user creation

if (username.equals("admin")) {

logger.error("Cannot create user with reserved username: {}", username);

throw new IllegalArgumentException("Reserved username");

}

// Success case

logger.info("User created successfully: username={}", username);

return true;

} catch (Exception e) {

logger.error("Unexpected error during user creation: username={}, email={}",

username, email, e);

return false;

}

}

public String getUserInfo(int userId) {

logger.debug("Fetching user info for userId: {}", userId);

if (userId <= 0) {

logger.warn("Invalid userId provided: {}", userId);

return null;

}

try {

// Simulate database access

if (userId == 999) {

logger.error("Database connection failed for userId: {}", userId);

throw new RuntimeException("Database error");

}

String userInfo = "User-" + userId;

logger.debug("Successfully retrieved user info: {}", userInfo);

return userInfo;

} catch (Exception e) {

logger.error("Failed to retrieve user info for userId: {}", userId, e);

return null;

}

}

}

// 3. DatabaseService.java - Database Operations with Logging

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

public class DatabaseService {

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

public void connectToDatabase() {

logger.info("Attempting to connect to database");

try {

// Simulate connection

Thread.sleep(100); // Simulate connection time

logger.info("Database connection established successfully");

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

logger.error("Database connection interrupted", e);

} catch (Exception e) {

logger.error("Failed to connect to database", e);

}

}

public boolean executeQuery(String query) {

logger.debug("Executing query: {}", query);

if (query == null || query.trim().isEmpty()) {

logger.warn("Cannot execute empty query");

return false;

}

try {

// Simulate query execution

if (query.toLowerCase().contains("drop")) {

logger.error("Dangerous query detected and blocked: {}", query);

return false;

}

Thread.sleep(50); // Simulate query time

logger.debug("Query executed successfully");

return true;

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

logger.error("Query execution interrupted: {}", query, e);

return false;

} catch (Exception e) {

logger.error("Query execution failed: {}", query, e);

return false;

}

}

}

// 4. ApplicationDemo.java - Complete Application Demo

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.slf4j.MDC;

public class ApplicationDemo {

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

public static void main(String[] args) {

logger.info("🌟 Starting Complete SLF4J Application Demo");

// Set application context

MDC.put("application", "SLF4J-Demo");

MDC.put("version", "1.0.0");

// Initialize services

UserService userService = new UserService();

DatabaseService databaseService = new DatabaseService();

// Demo 1: Database operations

demonstrateDatabaseOperations(databaseService);

// Demo 2: User operations

demonstrateUserOperations(userService);

// Demo 3: Error scenarios

demonstrateErrorScenarios(userService, databaseService);

// Demo 4: Performance logging

demonstratePerformanceLogging();

MDC.clear();

logger.info("✅ Application demo completed successfully");

}

private static void demonstrateDatabaseOperations(DatabaseService dbService) {

logger.info("=== Database Operations Demo ===");

dbService.connectToDatabase();

dbService.executeQuery("SELECT \* FROM users");

dbService.executeQuery("UPDATE users SET status = 'active'");

dbService.executeQuery(""); // Empty query

dbService.executeQuery("DROP TABLE users"); // Dangerous query

System.out.println();

}

private static void demonstrateUserOperations(UserService userService) {

logger.info("=== User Operations Demo ===");

userService.createUser("john\_doe", "john@example.com");

userService.createUser("", "invalid@example.com");

userService.createUser("jane\_doe", "invalid-email");

userService.createUser("admin", "admin@example.com");

userService.getUserInfo(123);

userService.getUserInfo(-1);

userService.getUserInfo(999);

System.out.println();

}

private static void demonstrateErrorScenarios(UserService userService,

DatabaseService dbService) {

logger.info("=== Error Scenarios Demo ===");

try {

// Simulate various error conditions

throw new RuntimeException("Simulated application error");

} catch (Exception e) {

logger.error("Application error occurred", e);

}

try {

// Another error scenario

processPayment(null, -100.0);

} catch (Exception e) {

logger.error("Payment processing error: amount={}, error={}",

-100.0, e.getMessage(), e);

}

System.out.println();

}

private static void demonstratePerformanceLogging() {

logger.info("=== Performance Logging Demo ===");

long startTime = System.currentTimeMillis();

try {

// Simulate time-consuming operation

Thread.sleep(200);

long endTime = System.currentTimeMillis();

long duration = endTime - startTime;

if (duration > 100) {

logger.warn("Slow operation detected: duration={}ms", duration);

} else {

logger.info("Operation completed: duration={}ms", duration);

}

} catch (InterruptedException e) {

Thread.currentThread().interrupt();

logger.error("Operation interrupted", e);

}

System.out.println();

}

private static void processPayment(String paymentId, double amount) {

if (paymentId == null) {

throw new IllegalArgumentException("Payment ID cannot be null");

}

if (amount <= 0) {

throw new IllegalArgumentException("Payment amount must be positive");

}

}

}

🚀 Running SLF4J Logging Examples...

=== Console Output (LoggingExample.main()) ===

2024-01-15 14:30:15.123 [main] INFO com.example.LoggingExample - 🚀 Starting SLF4J Logging Demonstration

2024-01-15 14:30:15.125 [main] INFO com.example.LoggingExample - === Basic Logging Levels ===

2024-01-15 14:30:15.126 [main] DEBUG com.example.LoggingExample - This is a DEBUG message - development info

2024-01-15 14:30:15.127 [main] INFO com.example.LoggingExample - This is an INFO message - general information

2024-01-15 14:30:15.128 [main] WARN com.example.LoggingExample - This is a WARN message - potential problems

2024-01-15 14:30:15.129 [main] ERROR com.example.LoggingExample - This is an ERROR message - serious issues

2024-01-15 14:30:15.130 [main] INFO com.example.LoggingExample - === Parameterized Logging ===

2024-01-15 14:30:15.131 [main] INFO com.example.LoggingExample - User John Doe with ID 12345 has balance $1250.75

2024-01-15 14:30:15.132 [main] WARN com.example.LoggingExample - User John Doe attempted to withdraw $2000.0 but only has $1250.75

2024-01-15 14:30:15.133 [main] ERROR com.example.LoggingExample - Failed login attempt for user John Doe from IP 192.168.1.100

2024-01-15 14:30:15.134 [main] INFO com.example.LoggingExample - === Exception Logging ===

2024-01-15 14:30:15.135 [main] ERROR com.example.LoggingExample - Arithmetic error occurred: / by zero

java.lang.ArithmeticException: / by zero

at com.example.LoggingExample.demonstrateExceptionLogging(LoggingExample.java:67)

at com.example.LoggingExample.main(LoggingExample.java:22)

2024-01-15 14:30:15.140 [main] ERROR com.example.LoggingExample - Null pointer exception in text processing

java.lang.NullPointerException: Cannot invoke "String.length()" because "text" is null

at com.example.LoggingExample.demonstrateExceptionLogging(LoggingExample.java:73)

at com.example.LoggingExample.main(LoggingExample.java:22)

2024-01-15 14:30:15.145 [main] ERROR com.example.LoggingExample - Failed to process user with context: operation=processUser, timestamp=2024-01-15T19:30:15.145Z, error=Invalid user ID: invalid-user-id

java.lang.IllegalArgumentException: Invalid user ID: invalid-user-id

at com.example.LoggingExample.processUser(LoggingExample.java:125)

at com.example.LoggingExample.demonstrateExceptionLogging(LoggingExample.java:78)

2024-01-15 14:30:15.150 [main] INFO com.example.LoggingExample - === Conditional Logging ===

2024-01-15 14:30:15.151 [main] DEBUG com.example.LoggingExample - Debug info: Debug data 0 Debug data 1 Debug data 2 ...

2024-01-15 14:30:15.152 [main] WARN com.example.LoggingExample - Warning level is enabled

2024-01-15 14:30:15.153 [main] INFO com.example.LoggingExample - === MDC (Mapped Diagnostic Context) ===

2024-01-15 14:30:15.154 [main] INFO com.example.LoggingExample - Processing payment request

2024-01-15 14:30:15.155 [main] WARN com.example.LoggingExample - Payment amount exceeds daily limit

2024-01-15 14:30:15.156 [main] ERROR com.example.LoggingExample - Payment processing failed

2024-01-15 14:30:15.157 [main] INFO com.example.LoggingExample - Context cleared - no MDC info

2024-01-15 14:30:15.158 [main] INFO com.example.LoggingExample - ✅ SLF4J Logging Demonstration completed

=== Console Output (ApplicationDemo.main()) ===

2024-01-15 14:30:16.200 [main] INFO com.example.ApplicationDemo - 🌟 Starting Complete SLF4J Application Demo

2024-01-15 14:30:16.201 [main] INFO com.example.ApplicationDemo - === Database Operations Demo ===

2024-01-15 14:30:16.202 [main] INFO com.example.DatabaseService - Attempting to connect to database

2024-01-15 14:30:16.305 [main] INFO com.example.DatabaseService - Database connection established successfully

2024-01-15 14:30:16.306 [main] DEBUG com.example.DatabaseService - Executing query: SELECT \* FROM users

2024-01-15 14:30:16.358 [main] DEBUG com.example.DatabaseService - Query executed successfully

2024-01-15 14:30:16.359 [main] DEBUG com.example.DatabaseService - Executing query: UPDATE users SET status = 'active'

2024-01-15 14:30:16.411 [main] DEBUG com.example.DatabaseService - Query executed successfully

2024-01-15 14:30:16.412 [main] WARN com.example.DatabaseService - Cannot execute empty query

2024-01-15 14:30:16.413 [main] ERROR com.example.DatabaseService - Dangerous query detected and blocked: DROP TABLE users

2024-01-15 14:30:16.415 [main] INFO com.example.ApplicationDemo - === User Operations Demo ===

2024-01-15 14:30:16.416 [main] INFO com.example.UserService - Creating user: username=john\_doe, email=john@example.com

2024-01-15 14:30:16.417 [main] INFO com.example.UserService - User created successfully: username=john\_doe

2024-01-15 14:30:16.418 [main] INFO com.example.UserService - Creating user: username=, email=invalid@example.com

2024-01-15 14:30:16.419 [main] WARN com.example.UserService - User creation failed: username is null or empty

2024-01-15 14:30:16.420 [main] INFO com.example.UserService - Creating user: username=jane\_doe, email=invalid-email

2024-01-15 14:30:16.421 [main] WARN com.example.UserService - User creation failed: invalid email format: invalid-email

2024-01-15 14:30:16.422 [main] INFO com.example.UserService - Creating user: username=admin, email=admin@example.com

2024-01-15 14:30:16.423 [main] ERROR com.example.UserService - Cannot create user with reserved username: admin

2024-01-15 14:30:16.424 [main] ERROR com.example.UserService - Unexpected error during user creation: username=admin, email=admin@example.com

java.lang.IllegalArgumentException: Reserved username

at com.example.UserService.createUser(UserService.java:28)

2024-01-15 14:30:16.430 [main] DEBUG com.example.UserService - Fetching user info for userId: 123

2024-01-15 14:30:16.431 [main] DEBUG com.example.UserService - Successfully retrieved user info: User-123

2024-01-15 14:30:16.432 [main] WARN com.example.UserService - Invalid userId provided: -1

2024-01-15 14:30:16.433 [main] DEBUG com.example.UserService - Fetching user info for userId: 999

2024-01-15 14:30:16.434 [main] ERROR com.example.UserService - Database connection failed for userId: 999

2024-01-15 14:30:16.435 [main] ERROR com.example.UserService - Failed to retrieve user info for userId: 999

java.lang.RuntimeException: Database error

at com.example.UserService.getUserInfo(UserService.java:49)

2024-01-15 14:30:16.440 [main] INFO com.example.ApplicationDemo - === Error Scenarios Demo ===

2024-01-15 14:30:16.441 [main] ERROR com.example.ApplicationDemo - Application error occurred

java.lang.RuntimeException: Simulated application error

at com.example.ApplicationDemo.demonstrateErrorScenarios(ApplicationDemo.java:76)

2024-01-15 14:30:16.445 [main] ERROR com.example.ApplicationDemo - Payment processing error: amount=-100.0, error=Payment ID cannot be null

java.lang.IllegalArgumentException: Payment ID cannot be null

at com.example.ApplicationDemo.processPayment(ApplicationDemo.java:115)

2024-01-15 14:30:16.450 [main] INFO com.example.ApplicationDemo - === Performance Logging Demo ===

2024-01-15 14:30:16.655 [main] WARN com.example.ApplicationDemo - Slow operation detected: duration=205ms

2024-01-15 14:30:16.660 [main] INFO com.example.ApplicationDemo - ✅ Application demo completed successfully

=== File Output (logs/application.log) ===

2024-01-15 14:30:15.123 [main] INFO com.example.LoggingExample - 🚀 Starting SLF4J Logging Demonstration

2024-01-15 14:30:15.125 [main] INFO com.example.LoggingExample - === Basic Logging Levels ===

2024-01-15 14:30:15.126 [main] DEBUG com.example.LoggingExample - This is a DEBUG message - development info

[... same content as console output ...]

=== File Output (logs/error.log) ===

2024-01-15 14:30:15.129 [main] ERROR com.example.LoggingExample - This is an ERROR message - serious issues

2024-01-15 14:30:15.133 [main] ERROR com.example.LoggingExample - Failed login attempt for user John Doe from IP 192.168.1.100

2024-01-15 14:30:15.135 [main] ERROR com.example.LoggingExample - Arithmetic error occurred: / by zero

java.lang.ArithmeticException: / by zero

at com.example.LoggingExample.demonstrateExceptionLogging(LoggingExample.java:67)

2024-01-15 14:30:15.140 [main] ERROR com.example.LoggingExample - Null pointer exception in text processing

java.lang.NullPointerException: Cannot invoke "String.length()" because "text" is null

at com.example.LoggingExample.demonstrateExceptionLogging(LoggingExample.java:73)

2024-01-15 14:30:15.156 [main] ERROR com.example.LoggingExample - Payment processing failed

2024-01-15 14:30:16.413 [main] ERROR com.example.DatabaseService - Dangerous query detected and blocked: DROP TABLE users

2024-01-15 14:30:16.423 [main] ERROR com.example.UserService - Cannot create user with reserved username: admin

2024-01-15 14:30:16.424 [main] ERROR com.example.UserService - Unexpected error during user creation: username=admin, email=admin@example.com

2024-01-15 14:30:16.435 [main] ERROR com.example.UserService - Failed to retrieve user info for userId: 999

2024-01-15 14:30:16.441 [main] ERROR com.example.ApplicationDemo - Application error occurred

2024-01-15 14:30:16.445 [main] ERROR com.example.ApplicationDemo - Payment processing error: amount=-100.0, error=Payment ID cannot be null

=== Maven Build Output ===

[INFO] Scanning for projects...

[INFO]

[INFO] --------------------< com.example:slf4j-logging-demo >---------------------

[INFO] Building slf4j-logging-demo 1.0.0

[INFO] --------------------------------[ jar ]---------------------------------

[INFO]

[INFO] --- maven-compiler-plugin:3.8.1:compile (default-compile) @ slf4j-logging-demo ---

[INFO] Changes detected - recompiling the module!

[INFO] Compiling 4 source files to /target/classes

[INFO]

[INFO] --- exec:java (default-cli) @ slf4j-logging-demo ---

[Logging output appears here...]

[INFO] ------------------------------------------------------------------------

[INFO] BUILD SUCCESS

[INFO] ------------------------------------------------------------------------

=== Key Learning Points from Output ===

✅ LOGGING LEVELS HIERARCHY:

TRACE < DEBUG < INFO < WARN < ERROR

(Only levels >= configured level are shown)

✅ PARAMETERIZED LOGGING:

logger.info("User {} has balance ${}", name, balance);

→ Efficient string interpolation without concatenation

✅ EXCEPTION LOGGING:

logger.error("Error message", exception);

→ Automatically includes full stack trace

✅ CONDITIONAL LOGGING:

if (logger.isDebugEnabled()) { ... }

→ Avoids expensive operations when level disabled

✅ MDC (Mapped Diagnostic Context):

MDC.put("userId", "123");

→ Adds context to all subsequent log messages

✅ FILE SEPARATION:

- application.log: All messages

- error.log: Only ERROR level messages

✅ CONFIGURATION BENEFITS:

- Rolling file appenders prevent disk space issues

- Different log levels per package

- Structured log format with timestamps and thread info

=== Execution Commands ===

# Compile and run with Maven:

mvn clean compile

mvn exec:java -Dexec.mainClass="com.example.LoggingExample"

mvn exec:java -Dexec.mainClass="com.example.ApplicationDemo"

# Or run directly with Java:

java -cp target/classes com.example.LoggingExample

java -cp target/classes com.example.ApplicationDemo

# Check log files:

cat logs/application.log

cat logs/error.log