Exception and Logging Handling in Spring Boot

# 1. Overview

This document describes the global exception handling and centralized logging setup implemented in a Spring Boot application. It includes usage of a global exception handler, MDC-based traceId filter, and a custom log format using Logback Encoder.

# 2. Global Exception Handling

The global exception handler class (`GlobalExceptionHandler`) handles all known and unknown exceptions in the application. It returns structured responses with traceable reference IDs.

## 2.1 Class: GlobalExceptionHandler

@ControllerAdvice  
public class GlobalExceptionHandler {  
 ...  
}

Key Features:

- Handles MethodArgumentNotValidException, HttpMediaTypeNotSupportedException, HttpRequestMethodNotSupportedException, custom exceptions, and generic exceptions.

- Returns a standardized `ErrorMessage` response.

- Extracts or generates a `correlationId` from HTTP headers.

# 3. Trace ID Filter

The `TraceIdFilter` is a servlet filter that generates and attaches a unique trace ID for each HTTP request using MDC (Mapped Diagnostic Context). This helps trace the log entries across layers.

@Component  
public class TraceIdFilter implements Filter {  
 public void doFilter(...) {  
 MDC.put("traceId", UUID.randomUUID().toString());  
 chain.doFilter(request, response);  
 MDC.clear();  
 }  
}

# 4. Log Format Configuration

A custom log format is used with Logback to improve readability and consistency. Colors and structure are added for clarity.

## 4.1 Pattern Used in Logback Encoder

encoder.setPattern(  
 "%cyan([%d{yyyy-MM-dd HH:mm:ss.SSS}]) " +  
 "%highlight(%-5level) " +  
 "[traceId=%X{traceId}] " +  
 "%blue([MyAwesomeApp]) " +  
 "%cyan(%file -> %class.%M) " +  
 ":%line " +  
 "%yellow(errorCode=%X{errorCode}) " +  
 "%msg%n"  
);

Highlights:

- Timestamp in cyan (sky blue)

- Log level with highlighting (INFO → green, ERROR → red, etc.)

- Trace ID from MDC

- Project name dynamically shown

- Class, file, method, and line number trace

- Error code in yellow

# 5. Benefits of This Approach

- Centralized, consistent error handling

- Clean and structured log output

- Easy traceability with traceId and refId

- Better developer productivity during debugging

- Production-ready format with proper severity levels