GraphQL API Error Handling Best Practices

# Overview

This document outlines best practices for handling exceptions in a Spring Boot GraphQL API using CQRS architecture.

# 1. Define Custom Exceptions

Create specific runtime exceptions to clearly indicate the problem domain.

public class TransferOrderNotFoundException extends RuntimeException {  
 public TransferOrderNotFoundException(String message) {  
 super(message);  
 }  
}

# 2. Use Exceptions in Services

Throw custom exceptions from your services or command/query handlers when appropriate:

public TransferOrder getTransferOrderById(String id) {  
 return repository.findById(id).orElseThrow(() ->  
 new TransferOrderNotFoundException("Transfer Order with ID " + id + " not found.")  
 );  
}

# 3. Implement Global GraphQL Error Handler

Customize error messages returned by GraphQL using GraphQLErrorHandler:

@Component  
public class CustomGraphQLErrorHandler implements GraphQLErrorHandler {  
  
 @Override  
 public List<GraphQLError> processErrors(List<GraphQLError> errors) {  
 return errors.stream()  
 .map(error -> {  
 if (error instanceof ThrowableGraphQLError) {  
 return error;  
 } else if (error.getMessage().contains("not found")) {  
 return new GenericGraphQLError(error.getMessage());  
 }  
 return error;  
 })  
 .collect(Collectors.toList());  
 }  
}

# 4. Use DataFetcherExceptionResolverAdapter (Modern Spring)

Leverage Spring GraphQL’s resolver to customize exception behavior:

@Component  
public class GraphQLExceptionResolver extends DataFetcherExceptionResolverAdapter {  
  
 @Override  
 protected DataFetcherExceptionHandlerResult resolveToSingleError(Throwable ex, DataFetcherExceptionHandlerParameters params) {  
 GraphQLError error = GraphQLError.newError()  
 .message(ex.getMessage())  
 .path(params.getPath())  
 .location(params.getSourceLocation())  
 .build();  
 return DataFetcherExceptionHandlerResult.newResult().error(error).build();  
 }  
}

# 5. Add Logging

Include structured logging to trace errors in logs:

private static final Logger logger = LoggerFactory.getLogger(TransferOrderService.class);  
  
public TransferOrder getTO(String id) {  
 try {  
 // business logic  
 } catch (TransferOrderNotFoundException ex) {  
 logger.warn("TO not found: {}", id);  
 throw ex;  
 }  
}

# Summary Table

|  |  |
| --- | --- |
| Layer | Best Practice |
| Domain/Service | Throw domain-specific exceptions |
| GraphQL Schema | Return structured error messages |
| Error Handling | Use GraphQLExceptionResolver or Kickstart’s handler |
| Logging | Use SLF4J/Logback |