Exception Handling Implementation

ADR Service Base Spring Boot Java Backend Project

# 1. Introduction

This document outlines the complete implementation and testing of a robust global exception handling mechanism in a ADR Service Base Spring Boot-based Java project. The approach leverages centralized exception handling using @ControllerAdvice, custom exception types, structured error response formatting, and test endpoints to verify behavior.

# 2. Objectives

* Centralize exception handling using `GlobalExceptionHandler`.
* Generate user-friendly, consistent error responses with correlation IDs.
* Support standard HTTP error codes (400, 404, 405, 412, 415, 500).
* Custom log formatting with trace IDs and error codes.
* Validation feedback using a structured `ValidationDetails` model.

# 3. Project Structure

* `GlobalExceptionHandler`: Central point for all exception handling.
* `ApiCustomException`: Custom runtime exception for business logic errors.
* `ErrorMessage` & `ValidationDetails`: Models to encapsulate structured error response.
* `ExceptionTestController`: Controller to simulate and test each type of error.

# 4. Exception Handling Scenarios

|  |  |  |
| --- | --- | --- |
| HTTP Status | Scenario | Description |
| 500 | Generic Exception | Unhandled exceptions return a 500 error. |
| 405 | Method Not Allowed | Triggered by invalid HTTP method. |
| 415 | Unsupported Media Type | Triggered when incorrect content type is used. |
| 400 | Invalid Input (Validation) | Triggers when request body fails validation. |
| 404 | Custom Not Found | Triggered when resource not found in DB. |
| 412 | Precondition Failed | Triggered by client version mismatch or duplicate. |

# 5. Error Response Format

Sample JSON error response structures:

{  
 "request\_id": "9d765ec5-bad3-4173-9c8d-570d80abd0e9",  
 "code": "EN006",  
 "message": "Idempotency identification value is a duplicate across your requests in the last 48 hours.",  
 "severity": "error",  
 "status": 412  
}

And

{  
 "request\_id": "9d765ec5-bad3-4173-9c8d-570d80abd0e9",  
 "code": "EN004",  
 "message": "Invalid request parameters provided",  
 "severity": "error",  
 "status": 400,  
 "reasons": [  
 {  
 "code": "R001",  
 "data": {  
 "field": "dateOfBirth",  
 "format": "yyyy-MM-dd"  
 },  
 "message": "Please provide valid value for 'dateOfBirth', in format 'yyyy-MM-dd'"  
 }  
 ]  
}

# 6. Testing Each Scenario

|  |  |  |  |
| --- | --- | --- | --- |
| URL | HTTP Method | Expected Status | Description |
| /api/test/generic | GET | 500 | Generic exception thrown. |
| /api/test/method-not-allowed | GET | 405 | Use GET instead of POST. |
| /api/test/media-type | POST | 415 | Send request with wrong media type. |
| /api/test/validation | POST | 400 | Send invalid input data. |
| /api/test/not-found/999 | GET | 404 | ID not found. |
| /api/test/precondition-failed | GET | 412 | Missing/invalid X-Client-Version. |
| /api/test/duplicate-check | POST | 412 | Send duplicate email. |
| /api/test/success | GET | 200 | Happy path. |

# 7. Benefits

* Single point of control for all exceptions.
* Improves maintainability and consistency.
* Easily extendable to add more business rules.
* Improves debuggability with detailed logs and request identifiers.