Exception Module - Detailed Sequence Diagrams

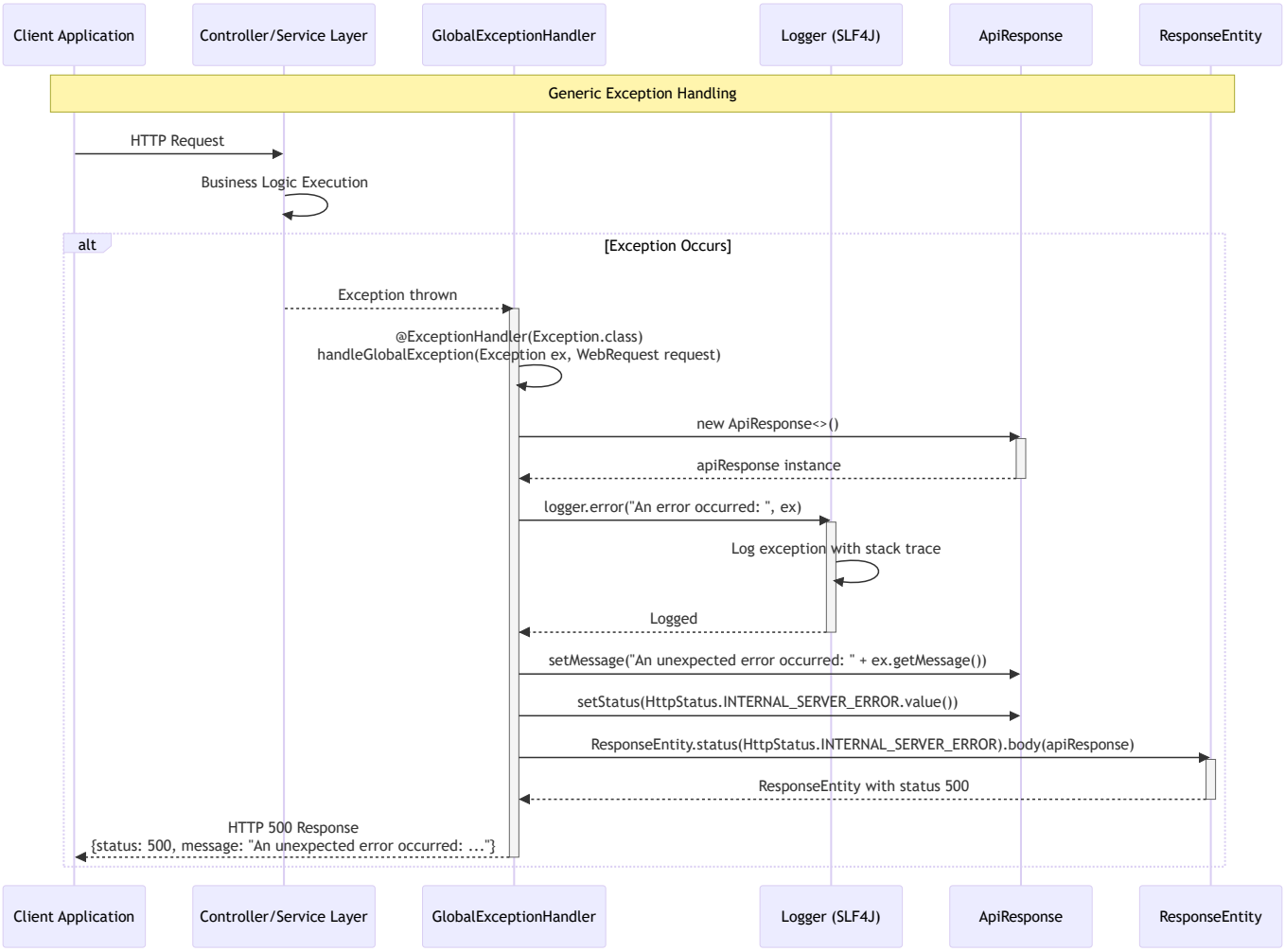
This document describes the **exception handling flows** implemented in the `com.i4o.dms.kubota.exception` module:

- **Global Exception Handling** (Generic exceptions, HTTP method errors, validation errors, and custom API exceptions).
- **Error Response Construction** (ApiErrorResponse and ApiResponse structures).
- **Validation Error Processing** (Field errors, object errors, and constraint violations).

All diagrams use Mermaid sequence diagrams and reflect the current implementation of the Exception module.

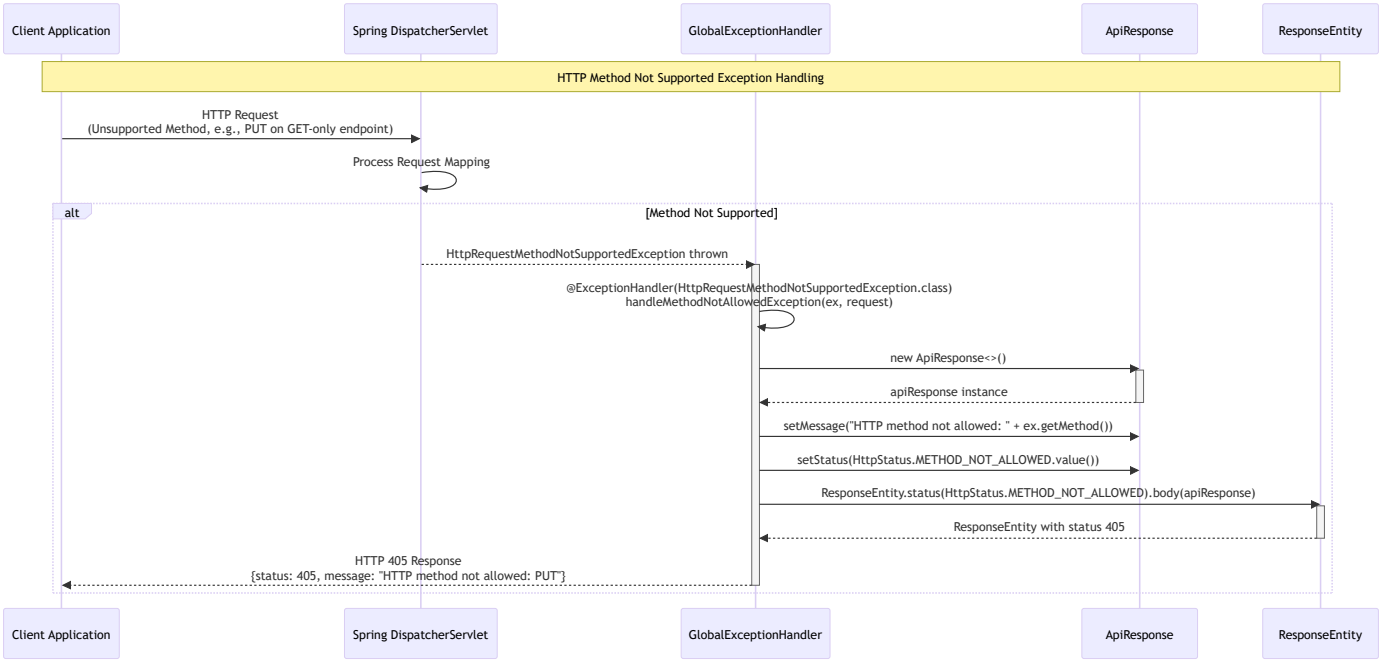
1. Generic Exception Handling Flow

This flow shows how **unexpected exceptions** are caught and handled by the GlobalExceptionHandler.



2. HTTP Method Not Supported Exception Flow

This flow shows how **HTTP method not allowed exceptions** are handled when an unsupported HTTP method is used.

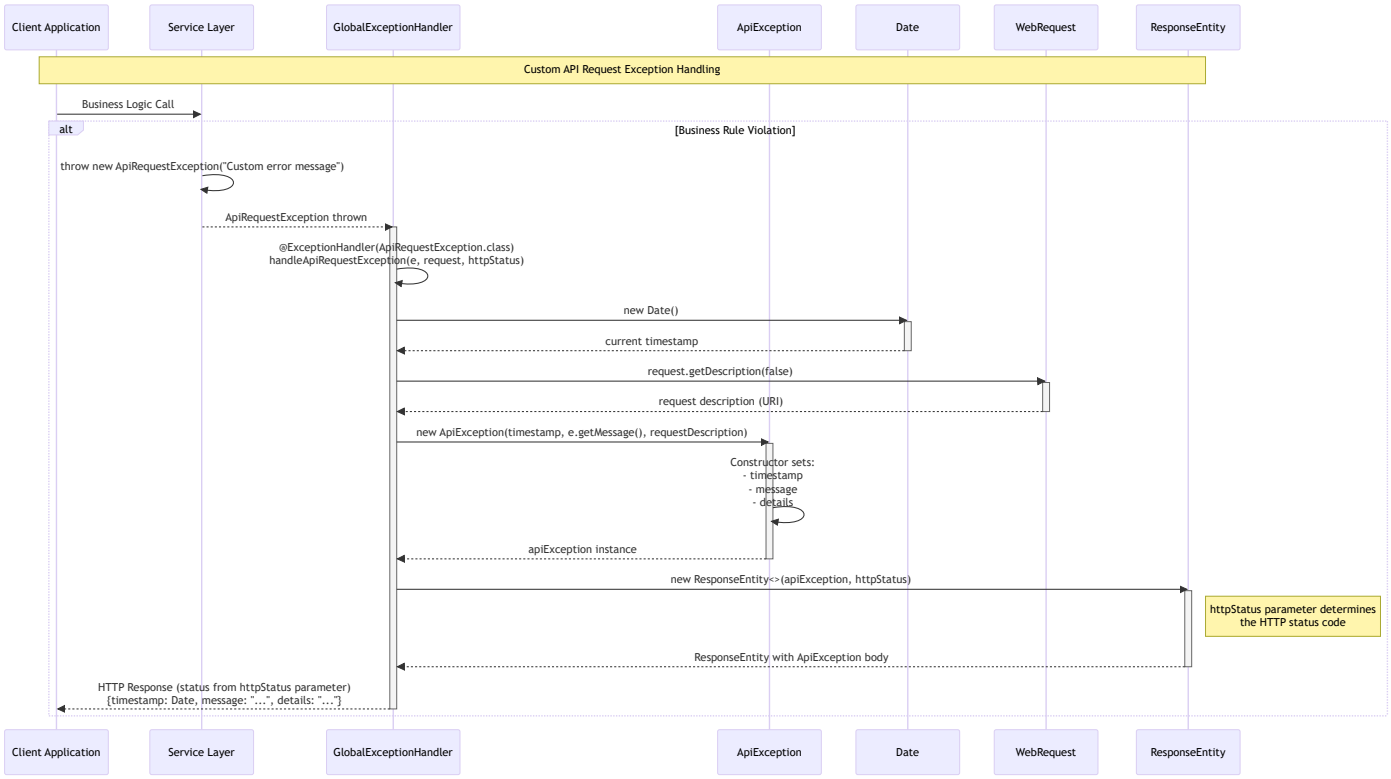


3. Validation Exception Handling Flow

This flow shows how **method argument validation errors** are processed and returned to the client.

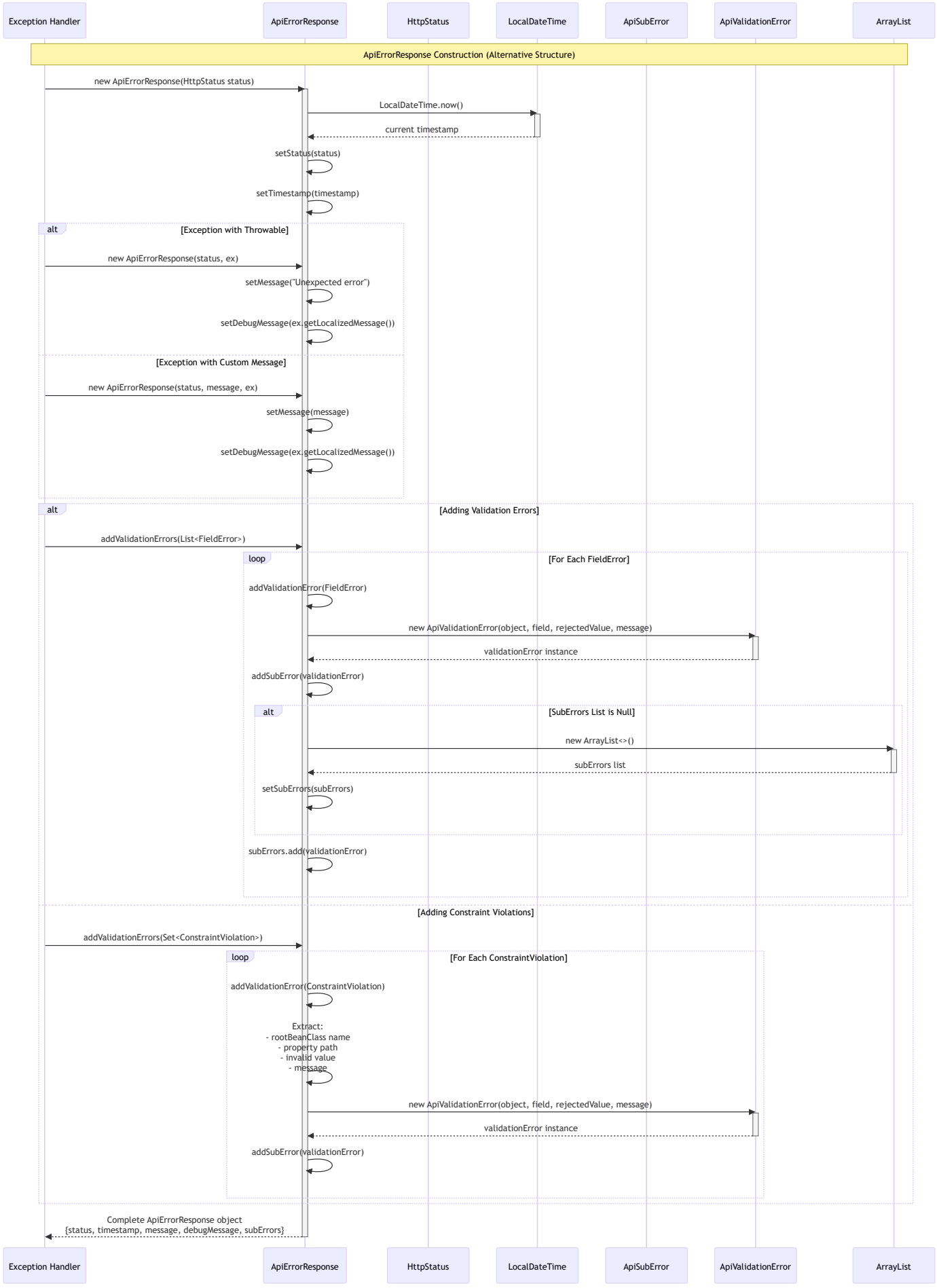
4. Custom API Request Exception Handling Flow

This flow shows how **custom ApiRequestException** instances are handled and converted to ApiException responses.



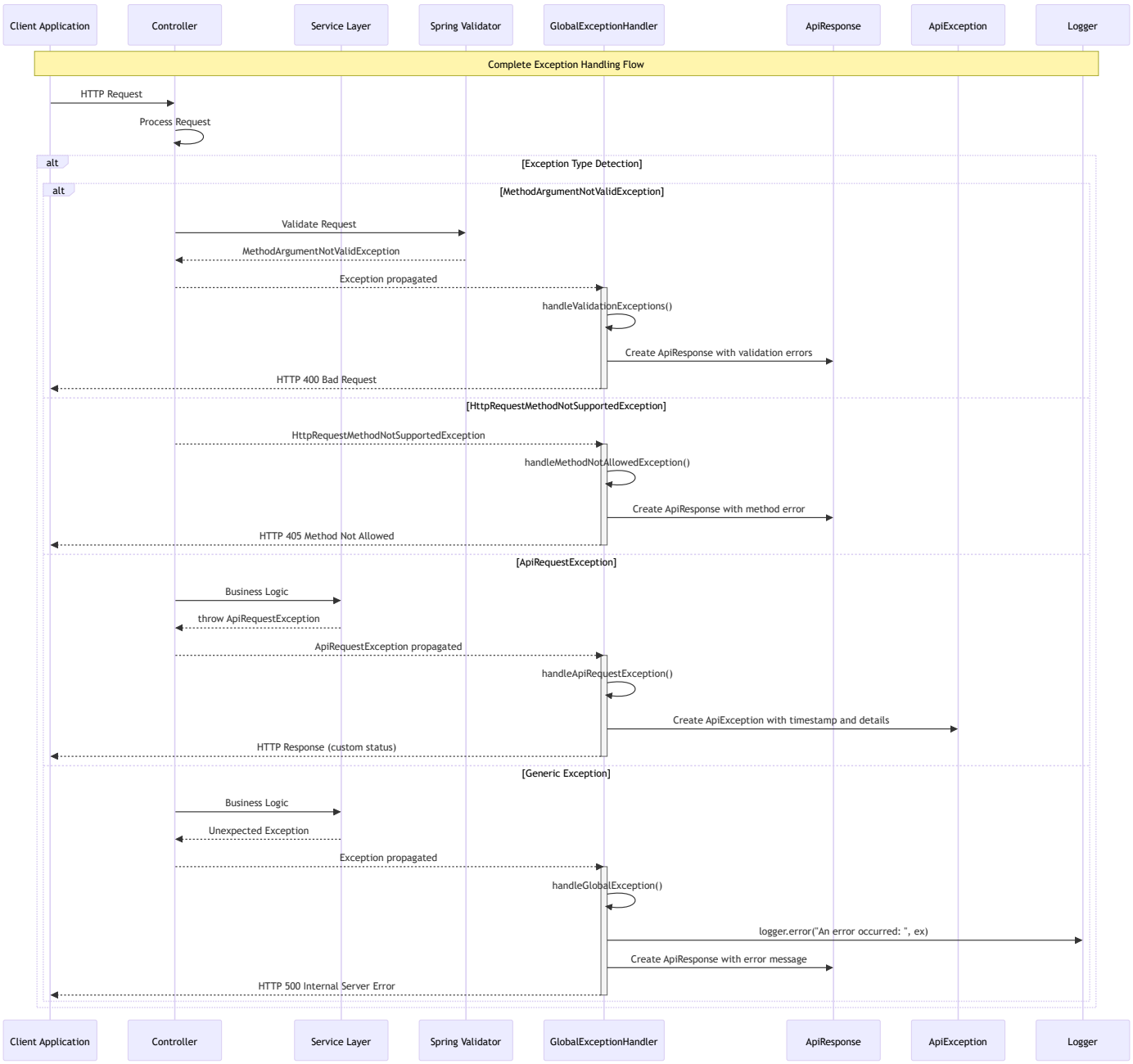
5. ApiErrorResponse Construction Flow (Alternative Error Response Structure)

This flow shows how **ApiErrorResponse** objects are constructed with detailed error information, including sub-errors for validation failures. Note: This class exists but is not currently used in GlobalExceptionHandler (which uses ApiResponse instead).


Exception Handler
ApiErrorResponse
HttpStatus
LocalDateTime
ApiSubError
ApiValidationError
ArrayList
ApiErrorResponse Construction (Alternative Structure)
new ApiErrorResponse(HttpStatus status)
LocalDateTime.now()
current timestamp
setStatus(status)
setTimestamp(timestamp)
alt
[Exception with Throwable]
new ApiErrorResponse(status, ex)
setMessage("Unexpected error")
setDebugMessage(ex.getLocalizedMessage())
[Exception with Custom Message]
new ApiErrorResponse(status, message, ex)
setMessage(message)
setDebugMessage(ex.getLocalizedMessage())
alt
[Adding Validation Errors]
addValidationErrors(List<FieldError>)
loop
[For Each FieldError]
addValidationError(FieldError)
new ApiValidationError(object, field, rejectedValue, message)
validationError instance
addSubError(validationError)
alt
[SubErrors List is Null]
new ArrayList<>()
subErrors list
setSubErrors(subErrors)
subErrors.add(validationError)
[Adding Constraint Violations]
addValidationErrors(Set<ConstraintViolation>)
loop
[For Each ConstraintViolation]
addValidationError(ConstraintViolation)
Extract:
- rootBeanClass name
- property path
- invalid value
- message
new ApiValidationError(object, field, rejectedValue, message)
validationError instance
addSubError(validationError)
Complete ApiErrorResponse object
{status, timestamp, message, debugMessage, subErrors}
Exception Handler
ApiErrorResponse
HttpStatus
LocalDateTime
ApiSubError
ApiValidationError
ArrayList

6. Complete Exception Handling Flow with Multiple Exception Types

This comprehensive flow shows how different exception types are routed to appropriate handlers in the GlobalExceptionHandler.



Class Structure Overview

GlobalExceptionHandler

- **Purpose**: Central exception handler using Spring's `@ControllerAdvice`
- **Exception Handlers**:
  i. `handleGlobalException()` - Catches all unhandled exceptions
  ii. `handleMethodNotAllowedException()` - Handles HTTP method errors
  iii. `handleValidationExceptions()` - Handles validation errors
  iv. `handleApiRequestException()` - Handles custom API exceptions

ApiErrorResponse

- **Purpose**: Structured error response with detailed error information
- **Features**:
  - HTTP status code
  - Timestamp (formatted as "dd-MM-yyyy hh:mm:ss")
  - Error message and debug message
  - Sub-errors for validation failures
  - Support for FieldError, ObjectError, and ConstraintViolation

ApiResponse

- **Purpose**: Standard API response wrapper used by GlobalExceptionHandler
- **Fields**: status, message, result, count, id, token

ApiException

- **Purpose**: Exception response structure for ApiRequestException
- **Fields**: timestamp (Date), message, details

ApiRequestException

- **Purpose**: Custom runtime exception for API-specific errors
- **Extends**: RuntimeException

Notes

1. **Current Implementation**: The `GlobalExceptionHandler` uses `ApiResponse` for most exceptions, while `ApiErrorResponse` exists but is not currently integrated into the handler methods.
2. **Exception Priority**: Spring's exception handling follows a priority order - more specific exception handlers are matched before generic ones.
3. **Logging**: Generic exceptions are logged with full stack traces using SLF4J Logger.
4. **Validation Errors**: The validation exception handler extracts field-level errors from Spring's BindingResult and maps them to a HashMap structure.
5. **Custom Exceptions**: `ApiRequestException` allows custom error messages and can be thrown with different HTTP status codes via the handler method parameter.