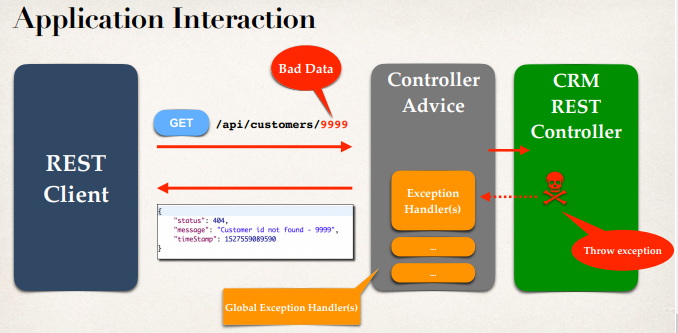
**65. Spring REST - CRUD Database Project - Exception Handling**

**Exception Handling**:

Here we will add Exception Handling code for our "spring-crm-rest" application. In our API, we add Exception Handling code for give the appropriate response for bad data.



**Development Process (Step-by-Step)**:

1. Create a custom error response class
2. Create a custom exception class
3. Update REST service to throw exception if customer not found
4. Add an exception handler method using **@ExceptionHandler**

**Step-1: Create a custom error response class**:

**package** com.ruhul.springdemo.rest;

**public** **class** CustomerErrorResponce {

**private** **int** status;

**private** String message;

**private** **long** timeStamp;

**public** CustomerErrorResponce() {

}

**public** CustomerErrorResponce(**int** status, String message, **long** timeStamp) {

**super**();

**this**.status = status;

**this**.message = message;

**this**.timeStamp = timeStamp;

}

**public** **int** getStatus() {

**return** status;

}

**public** **void** setStatus(**int** status) {

**this**.status = status;

}

**public** String getMessage() {

**return** message;

}

**public** **void** setMessage(String message) {

**this**.message = message;

}

**public** **long** getTimeStamp() {

**return** timeStamp;

}

**public** **void** setTimeStamp(**long** timeStamp) {

**this**.timeStamp = timeStamp;

}

}

**Step-2: Create a custom exception class**:

**package** com.ruhul.springdemo.rest;

**public** **class** CustomerNotFoundException **extends** RuntimeException {

**public** CustomerNotFoundException() {

}

**public** CustomerNotFoundException(String message) {

**super**(message);

}

**public** CustomerNotFoundException(Throwable cause) {

**super**(cause);

}

**public** CustomerNotFoundException(String message, Throwable cause) {

**super**(message, cause);

}

**public** CustomerNotFoundException(String message, Throwable cause,

**boolean** enableSuppression,

**boolean** writableStackTrace) {

**super**(message, cause, enableSuppression, writableStackTrace);

}

}

**Step-3: Update REST service to throw exception if customer not found**:

@RestController

@RequestMapping("/api")

**public** **class** CustomerRestController {

// autowire the CustomerService => inject the dependency

@Autowired

**private** CustomerService customerService;

// add mapping for GET /customers

@GetMapping("/customers")

**public** List<Customer> getCustomers() {

**return** customerService.getCustomers();

}

// add mapping for GET /customers/{customerId}

@GetMapping("/customers/{customerId}")

**public** Customer getCustomer(@PathVariable **int** customerId) {

Customer theCustomer = customerService.getCustomer(customerId);

**if** (theCustomer == **null**) {

**throw** **new** CustomerNotFoundException("Customer id not found - "

+ customerId);

}

**return** theCustomer;

}

}

**Step-4: Add an exception handler method using @ExceptionHandler:**

Create "**CustomerRestExceptionHandler.class**". Here we will add global Exception Handling code.

**package** com.ruhul.springdemo.rest;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.ControllerAdvice;

**import** org.springframework.web.bind.annotation.ExceptionHandler;

@ControllerAdvice

**public** **class** CustomerRestExceptionHandler {

// Add an exception handler for CustomerNotFoundException

@ExceptionHandler

**public** ResponseEntity<CustomerErrorResponce> handleException(

CustomerNotFoundException exc) {

// create CustomerErrorResponce

CustomerErrorResponce error = **new** CustomerErrorResponce(

HttpStatus.***NOT\_FOUND***.value(),

exc.getMessage(),

System.*currentTimeMillis*());

// return ResponseEntity

**return** **new** ResponseEntity<>(error, HttpStatus.***NOT\_FOUND***);

}

// Add another exception handler .. to catch any exception (catch all)

@ExceptionHandler

**public** ResponseEntity<CustomerErrorResponce> handleException(Exception exc) {

// create CustomerErrorResponce

CustomerErrorResponce error = **new** CustomerErrorResponce(

HttpStatus.***BAD\_REQUEST***.value(),

exc.getMessage(),

System.*currentTimeMillis*());

// return ResponseEntity

**return** **new** ResponseEntity<>(error, HttpStatus.***BAD\_REQUEST***);

}

}

Exception Handling