**61.2. Spring REST - Exception Handling (code)**:

**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 student not found
4. Add an exception handler method using **@ExceptionHandler**

**Step-1: Create a custom error response class (StudentErrorResponse)**:

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

**public** **class** StudentErrorResponse {

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

**private** String message;

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

// create constructor

// getter / setter method

}

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

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

**public** **class** StusentNotFoundException **extends** RuntimeException {

// generate constructor from superclass

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

**super**(message, cause);

}

**public** StusentNotFoundException(String message) {

**super**(message);

}

**public** StusentNotFoundException(Throwable cause) {

**super**(cause);

}

}

That’s (**StudentNotFoundException.java**) our custom Student Exception class. We will use this in our REST service, when it's actually time to throw an exception, if the student\_id is a bad student\_id or we cannot find the student\_id.

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

**File: StudentRestController.java**:

// define endpoint for "/students/{studentId}" => return student at index

@GetMapping("/students/{studentId}")

**public** Student getStudent(@PathVariable **int** studentId) {

// check the studentId against list size

**if** (studentId >= theStudents.size() || studentId < 0) {

**throw** **new** StusentNotFoundException("Student ID not found: " +

studentId);

}

**return** theStudents.get(studentId);

}

**Step-4: Add an exception handler method using @ExceptionHandler (StudentRestController.java)**:

**File: StudentRestController.java**:

// Add an Exception Handler using @ExceptionHandler

@ExceptionHandler

**public** ResponseEntity<StudentErrorResponse> handleException(

StudentNotFoundException exc) {

// create a StudentErrorResponse

StudentErrorResponse error = **new** StudentErrorResponse();

error.setStatus(HttpStatus.***NOT\_FOUND***.value());

error.setMessage(exc.getMessage());

error.setTimeStamp(System.*currentTimeMillis*());

// return ResponseEntity

// error => Body

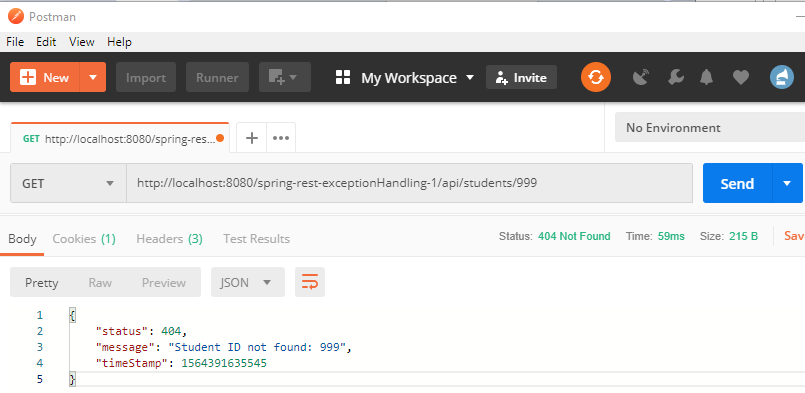
// HttpStatus.NOT\_FOUND => Status code

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

}

**Test the app**:

Now after run the application if we search an invalid id then we will see the error as JSON format.



61.2. Spring REST - Exception Handling (code)