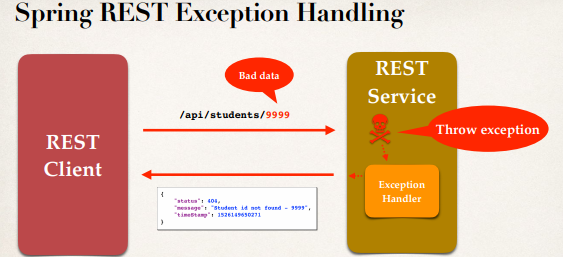
**61.4. Spring REST - Global Exception Handling**

**Spring REST - Global Exception Handling**:

In previous video we add code for exception handling. If bad data coming across the URL our REST service throw an exception and inside REST service, we had an exception handler that would actually pass back that exception as JSON data.



**Problem with this approach**:

* Exception handler code is only for the specific REST controller
* Can't be reused by other controllers (Large projects will have multiple controller)
* We need global ExceptionHandler
  + Promotes reuse
  + Centralizes exception handling

**Solve the problem**:

To solve the problem, we can use **@ControllerAdvice**.

**Spring @ControllerAdvice**:

* **@ControllerAdvice** is similar to an interceptor / filter
* Pre-process requests to controllers
* Post-process responses to handle exceptions
* Perfect for global exception handling

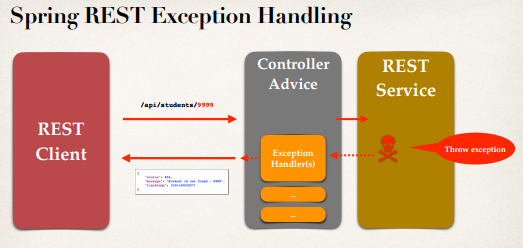
The **@ControllerAdvice** is Real-time use of AOP (Aspect Oriented Programming). This is AOP that is use to kind of pre-process, and post-process on controllers.

**Spring REST Exception Handling Global**:

Now with this Spring REST exception handling with **@ControllerAdvice** we have our REST client and REST server. We make a request across to the actual service. Then we will have the controller advice to kind of pre-process the request, and then the REST service will execute.

If there something wrong will throw an exception. Now the exception handlers going to be moved out

and placed in the controller advice. This will give us support for global exception handling.



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

1. Create new @ControllerAdvice
2. Refactor REST service … remove exception handling code from REST service
3. Add exception handling code to @ControllerAdvice

**Step-1: Create a new @ControllerAdvice**:

Create a new class "**StudentRestExceptionHandler.java**" and give the annotation "**@ControllerAdvice**"

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

@ControllerAdvice

**public** **class** StudentRestExceptionHandler {

// add exception handling code here

}

**Step-2: Refactor REST service … remove exception handling code from REST service**:

In our REST controller "**StudentRestController.java**" we don’t write any Exception handling code. All Exception Handling code we will write in the "**StudentRestExceptionHandler.class**" class. We remove the exception handling related code from this class.

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

**import** java.util.ArrayList;

**import** java.util.List;

**import** javax.annotation.PostConstruct;

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

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

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

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

**import** com.ruhul.springdemo.entity.Student;

@RestController

@RequestMapping("/api")

**public** **class** StudentRestController {

**private** List<Student> theStudents;

// define @PostConstruct to load the student data ... only once

@PostConstruct

**private** **void** loadData() {

theStudents = **new** ArrayList<>();

theStudents.add(**new** Student("Ruhul", "Amin"));

theStudents.add(**new** Student("Rezau", "Islam"));

theStudents.add(**new** Student("Osman", "Goni"));

theStudents.add(**new** Student("Kapil", "Dev"));

theStudents.add(**new** Student("Mezbaul", "Islam"));

}

// define endpoint for "/students" => return list of students

@GetMapping("/students")

**public** List<Student> getStudents() {

**return** theStudents;

}

// 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** StudentNotFoundException("Student ID not found: " +

studentId);

}

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

}

}

**Step-3: Add exception handling code to @ControllerAdvice**:

**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** StudentRestExceptionHandler {

// add exception handling code here

// 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***);

}

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

@ExceptionHandler

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

// create a StudentErrorResponse

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

error.setStatus(HttpStatus.***BAD\_REQUEST***.value());

error.setMessage(exc.getMessage());

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

// return ResponseEntity

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

}

}

61.4. Spring REST - Global Exception Handling