Spring REST - Exception Handling

Remember our problem?

Remember our problem?

- Bad student id of 9999 …

Remember our problem?

- Bad student id of 9999 …

Remember our problem?

- Bad student id of 9999 …

Scary HTML error page

We really want this …

We really want this …

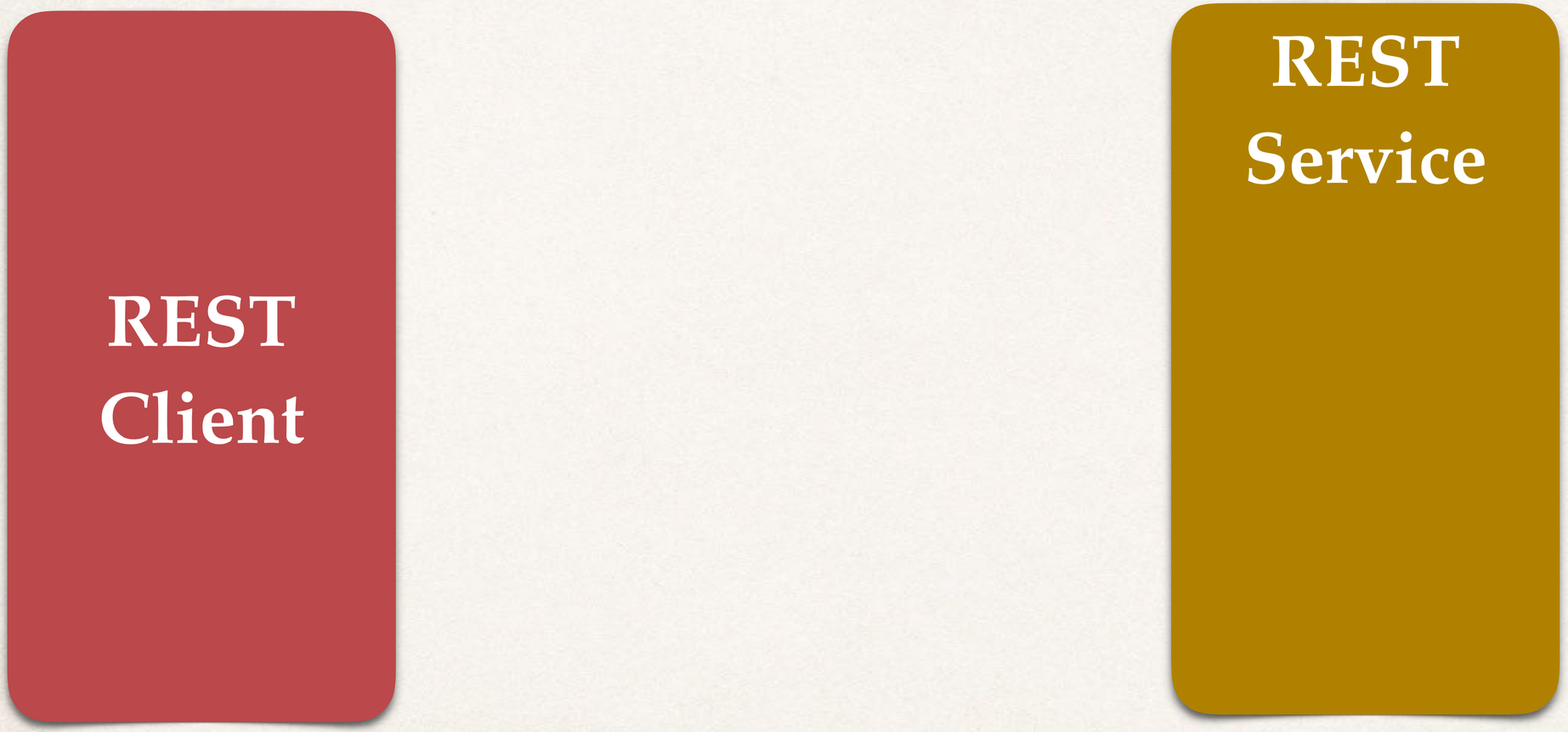
- Handle the exception and return error as JSON

We really want this …

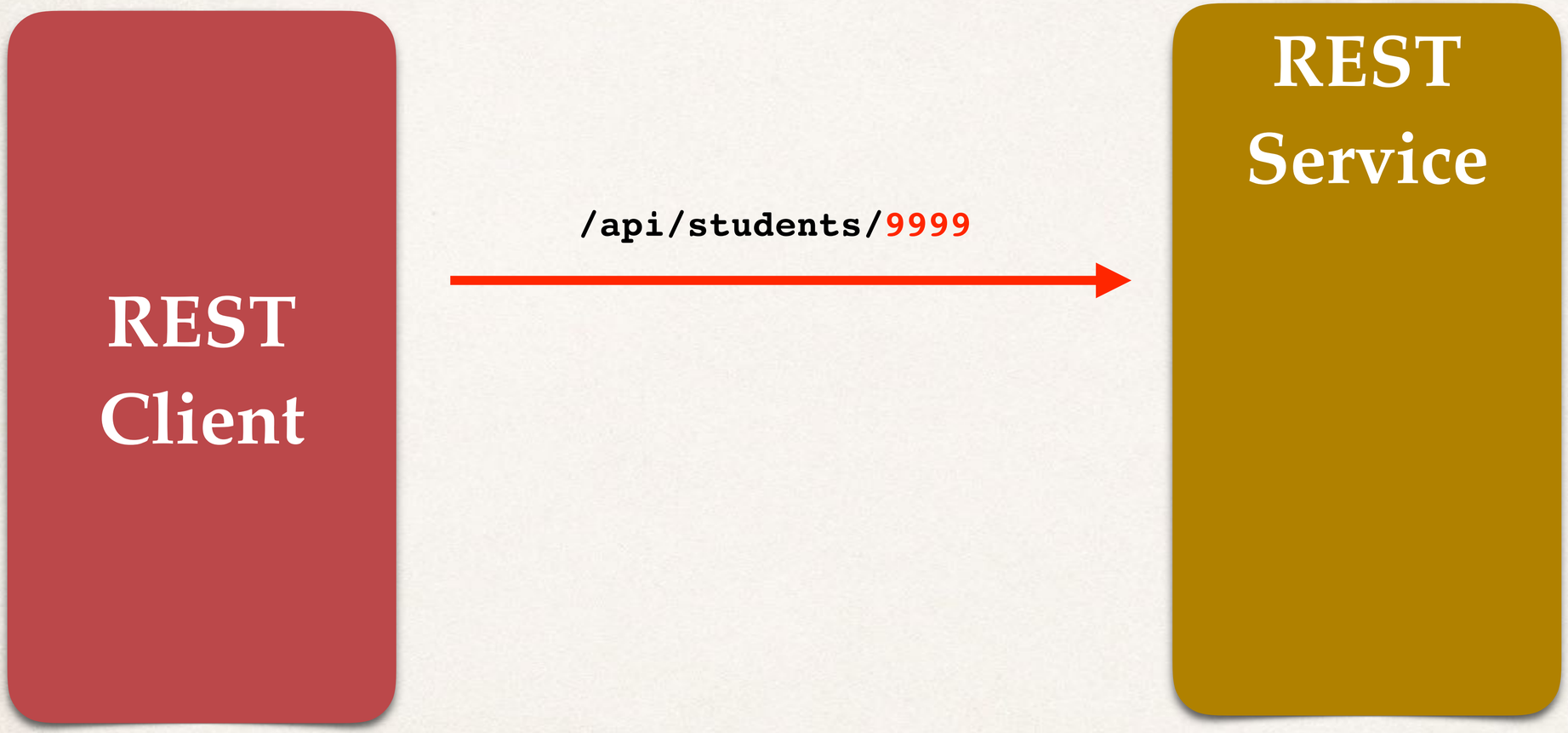
- Handle the exception and return error as JSON

```
{
    "status": 404,
    "message": "Student id not found - 9999",
    "timeStamp": 1526149650271
}
```

Spring REST Exception Handling



Spring REST Exception Handling

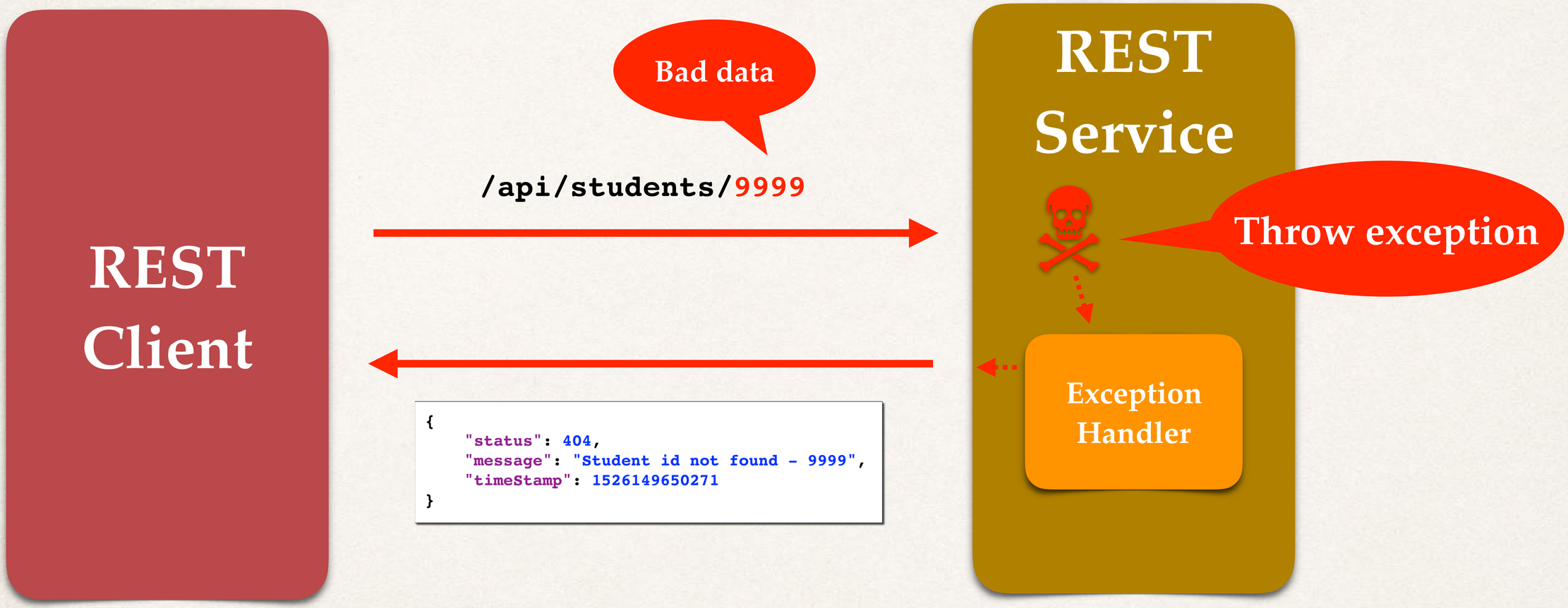


Spring REST Exception Handling

Spring REST Exception Handling

Spring REST Exception Handling

Spring REST Exception Handling



Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create a custom error response class

Development Process

Step-By-Step

1. Create a custom error response class

2. Create a custom exception class

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

Development Process

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 custom error response class

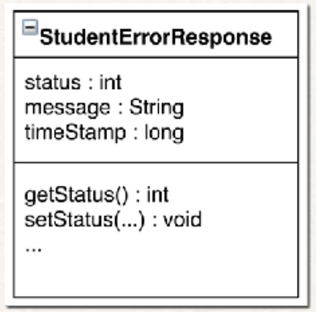
Step 1: Create custom error response class

- The custom error response class will be sent back to client as JSON

Step 1: Create custom error response class

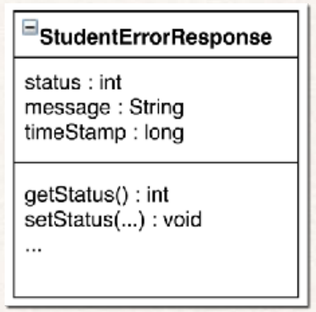
- The custom error response class will be sent back to client as JSON

- We will define as Java class (POJO)



Step 1: Create custom error response class

- The custom error response class will be sent back to client as JSON
- We will define as Java class (POJO)

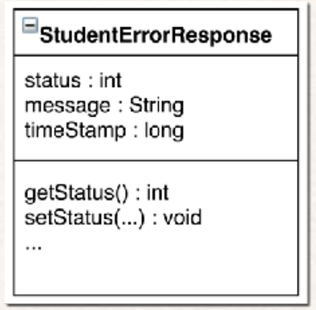


- Jackson will handle converting it to JSON

```
{
    "status": 404,
    "message": "Student id not found - 9999",
    "timeStamp": 1526149650271
}
```

Step 1: Create custom error response class

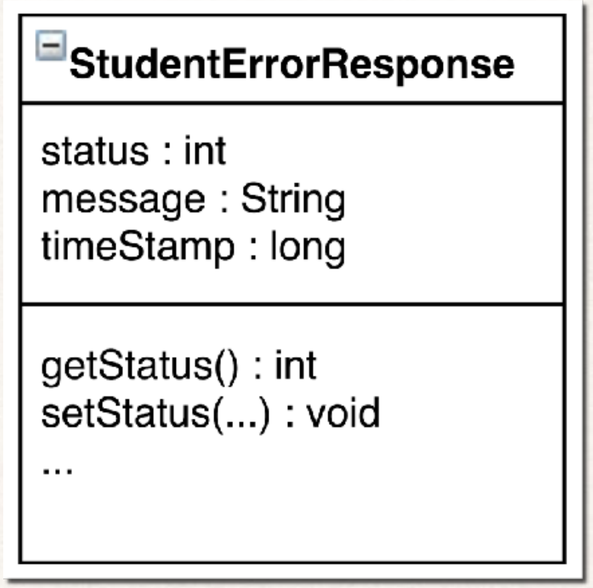
- The custom error response class will be sent back to client as JSON
- We will define as Java class (POJO)
- Jackson will handle converting it to JSON



You can define any custom fields that you want to track

```
{
    "status": 404,
    "message": "Student id not found - 9999",
    "timeStamp": 1526149650271
}
```

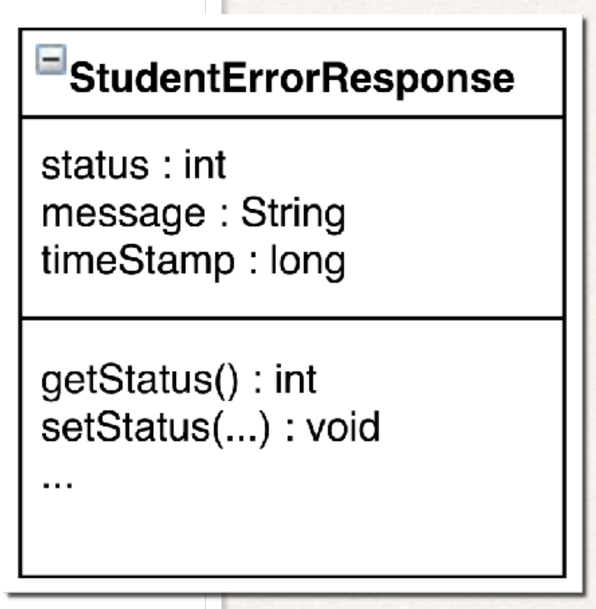
Step 1: Create custom error response class



Step 1: Create custom error response class

File: StudentErrorResponse.java

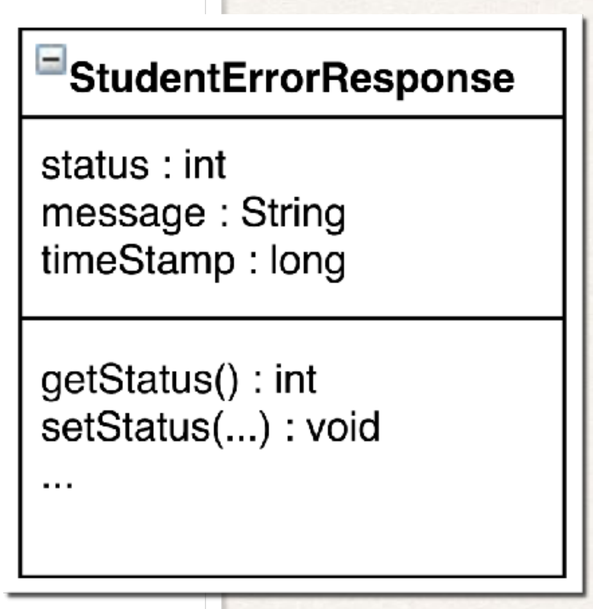
```
public class StudentErrorResponse {

  private int status;
  private String message;
  private long timeStamp;
```



Step 1: Create custom error response class

File: StudentErrorResponse.java

```
public class StudentErrorResponse {

  private int status;
  private String message;
  private long timeStamp;

  // constructors

  // getters / setters

}
```




```
{
    "status": 404,
    "message": "Student id not found - 9999",
    "timeStamp": 1526149650271
}
```

Step 2: Create custom student exception

Step 2: Create custom student exception

- The custom student exception will used by our REST service

Step 2: Create custom student exception

- The custom student exception will used by our REST service

- In our code, if we can't find student, then we'll throw an exception

Step 2: Create custom student exception

- The custom student exception will used by our REST service
- In our code, if we can't find student, then we'll throw an exception
- Need to define a custom student exception class

Step 2: Create custom student exception

- The custom student exception will used by our REST service
- In our code, if we can't find student, then we'll throw an exception
- Need to define a custom student exception class
  - `StudentNotFoundException`

Step 2: Create custom student exception

Step 2: Create custom student exception

File: StudentNotFoundException.java

```
public class StudentNotFoundException extends RuntimeException {

}
```

Step 2: Create custom student exception

File: StudentNotFoundException.java

```
public class StudentNotFoundException extends RuntimeException {

  public StudentNotFoundException(String message) {
    super(message);
  }

}
```

Call super class constructor

Step 3: Update REST service to throw exception

Step 3: Update REST service to throw exception

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
```

Step 3: Update REST service to throw exception

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {

  @GetMapping("/students/{studentId}")
  public Student getStudent(@PathVariable int studentId) {
```

Step 3: Update REST service to throw exception

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {

  @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);
    }
```

Throw exception

Step 3: Update REST service to throw exception

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {

  @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);
    }
```

Could also check results from DB

Throw exception

Step 3: Update REST service to throw exception

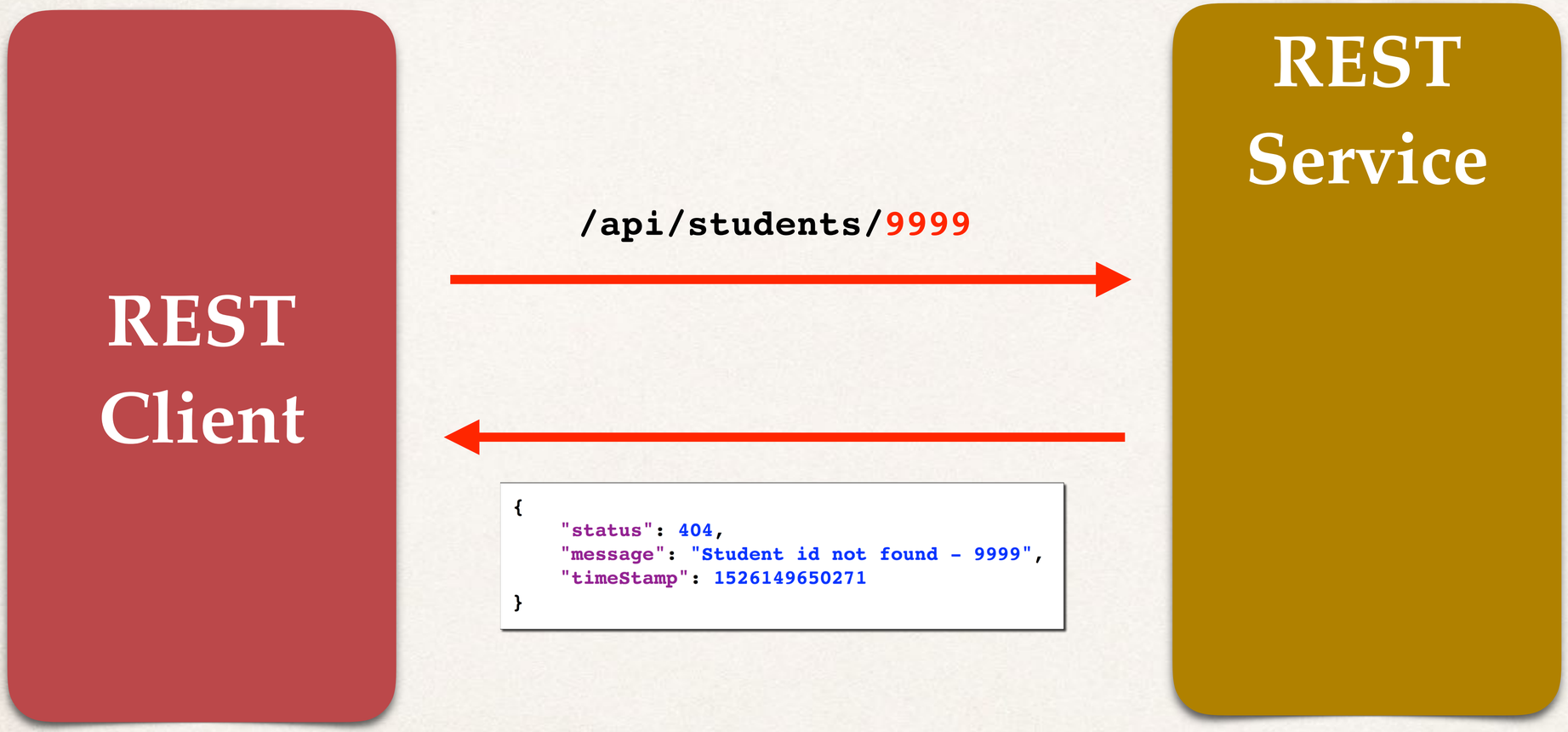
File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {

  @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);

  }
  ...
}
```

Throw exception

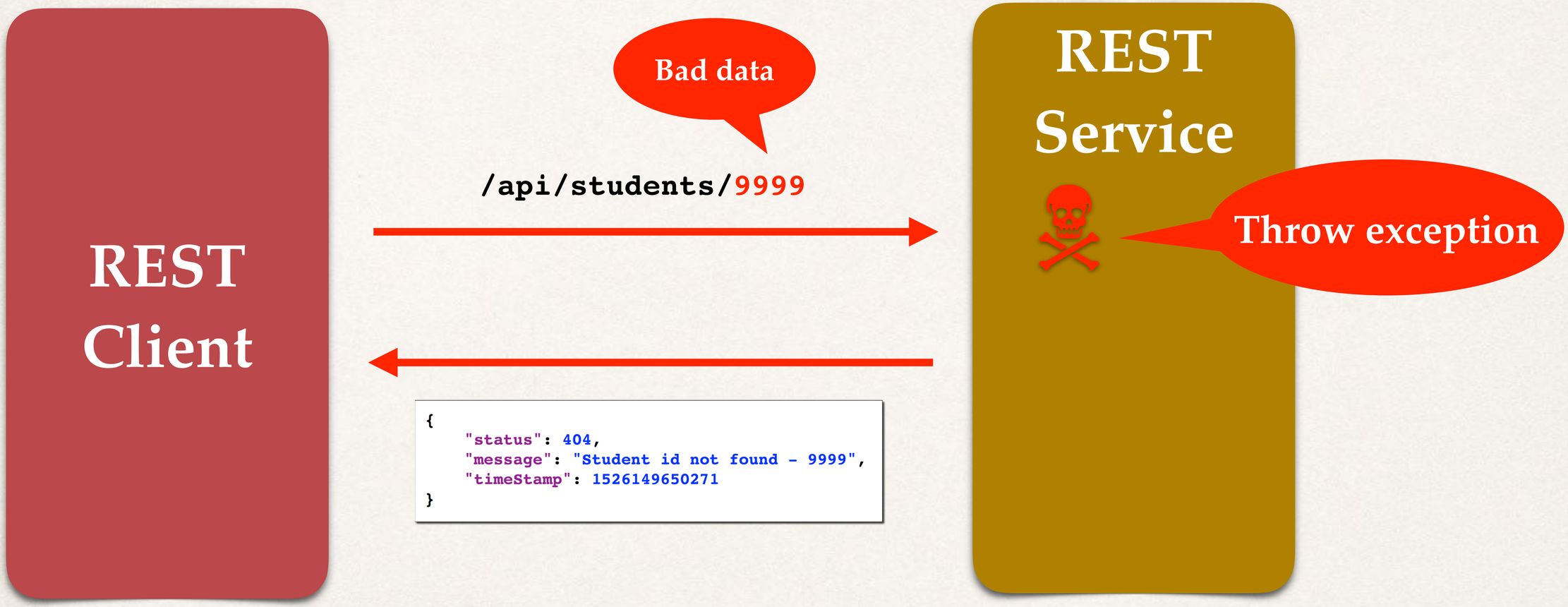
Happy path

Spring REST Exception Handling

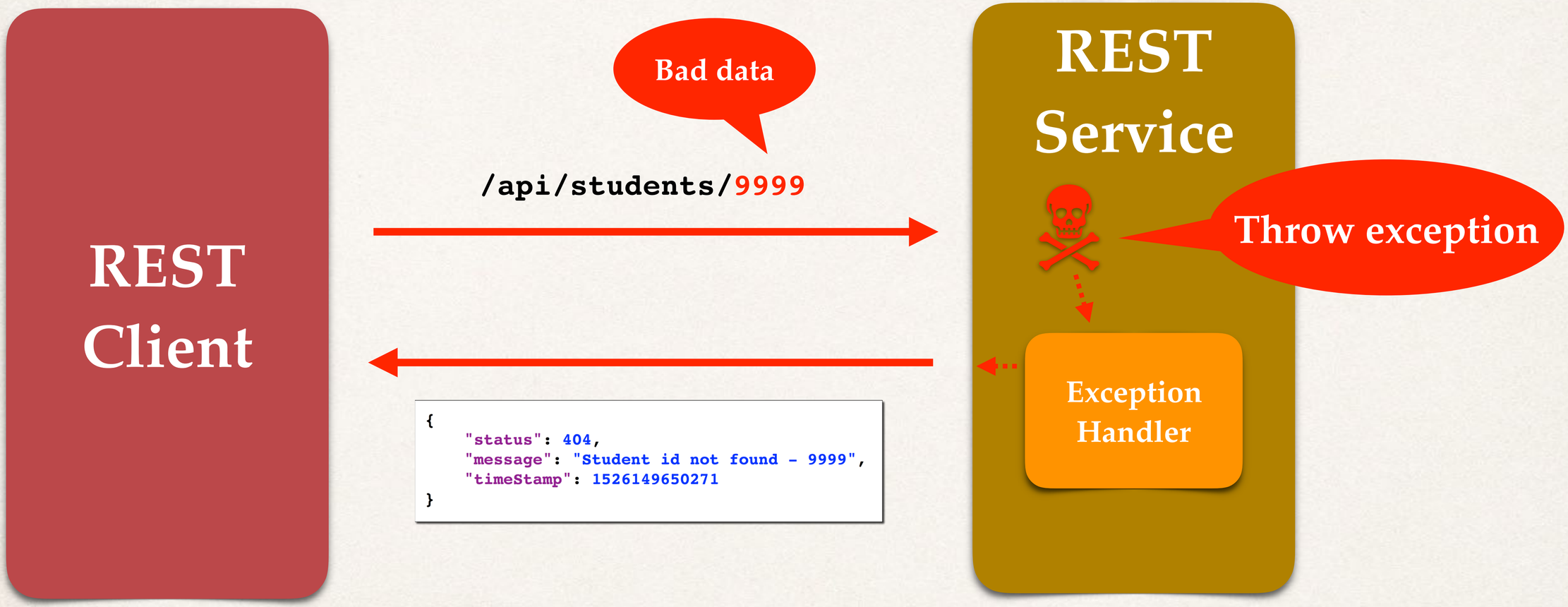


Spring REST Exception Handling

Spring REST Exception Handling



Spring REST Exception Handling



Step 4: Add exception handler method

Step 4: Add exception handler method

- Define exception handler method(s) with `@ExceptionHandler` annotation

Step 4: Add exception handler method

- Define exception handler method(s) with `@ExceptionHandler` annotation

- Exception handler will return a `ResponseEntity`

Step 4: Add exception handler method

- Define exception handler method(s) with `@ExceptionHandler` annotation
- Exception handler will return a `ResponseEntity`
- `ResponseEntity` is a wrapper for the HTTP response object

Step 4: Add exception handler method

- Define exception handler method(s) with `@ExceptionHandler` annotation
- Exception handler will return a `ResponseEntity`
- `ResponseEntity` is a wrapper for the HTTP response object
- `ResponseEntity` provides fine-grained control to specify:

Step 4: Add exception handler method

- Define exception handler method(s) with `@ExceptionHandler` annotation
- Exception handler will return a `ResponseEntity`
- `ResponseEntity` is a wrapper for the HTTP response object
- `ResponseEntity` provides fine-grained control to specify:
  - HTTP status code, HTTP headers and Response body

Step 4: Add exception handler method

Step 4: Add exception handler method

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …
```

Step 4: Add exception handler method

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {
```

Step 4: Add exception handler method

Exception handler method

java

```
pi")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {
```

Step 4: Add exception handler method

Exception handler method

Type of the response body

java

```
pi")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {
```

Step 4: Add exception handler method

Exception handler method

Type of the response body

Exception type to handle / catch

java

```
pi")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {
```

Step 4: Add exception handler method

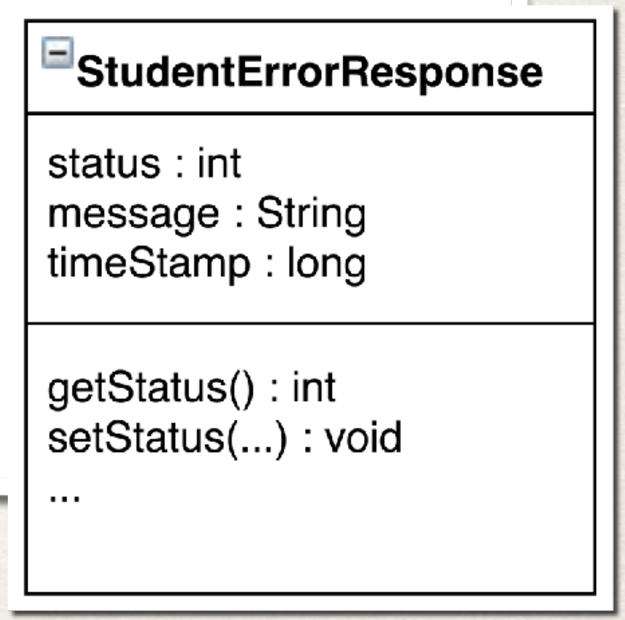
File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();
```

Step 4: Add exception handler method

File: StudentRestController.java

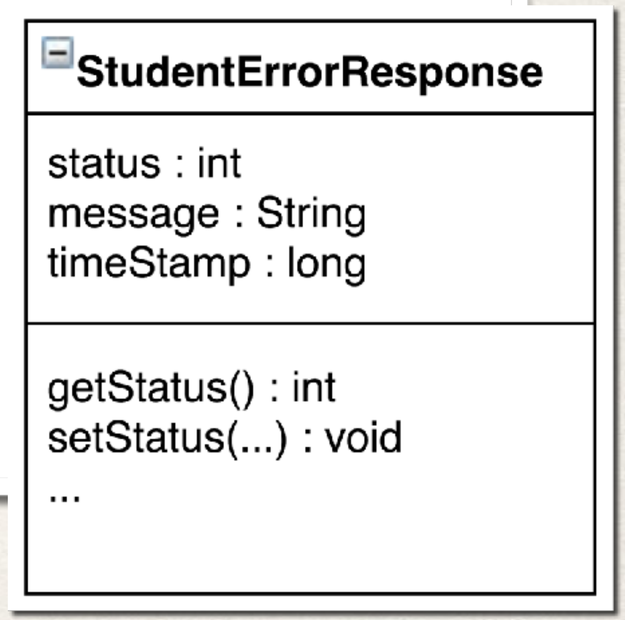
```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();
```



Step 4: Add exception handler method

File: StudentRestController.java

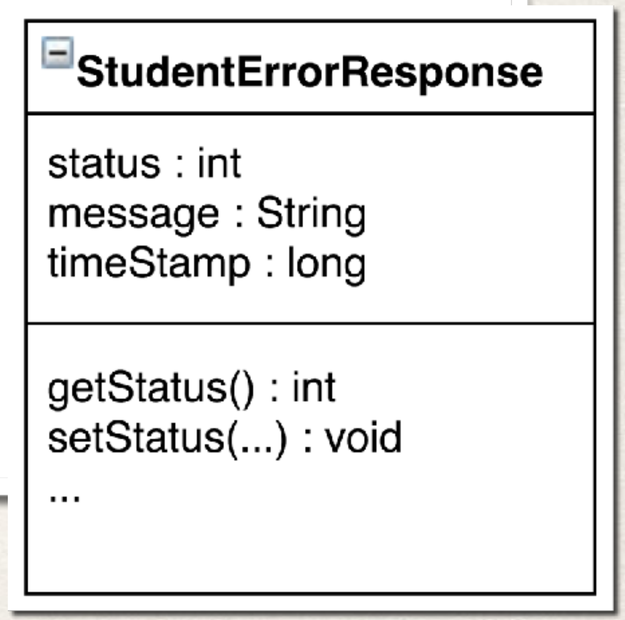
```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
```



Step 4: Add exception handler method

File: StudentRestController.java

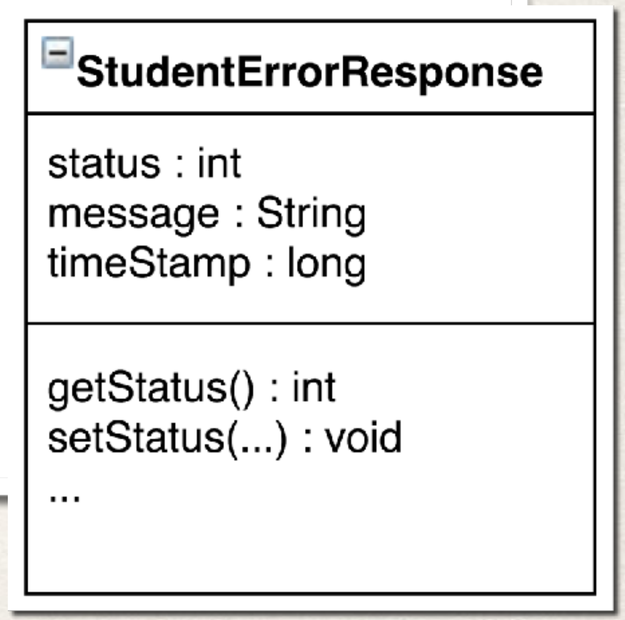
```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
```



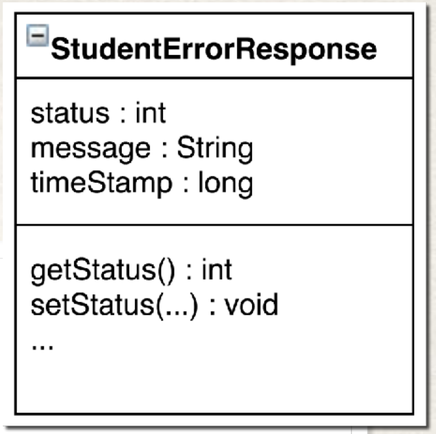
Step 4: Add exception handler method

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());
```



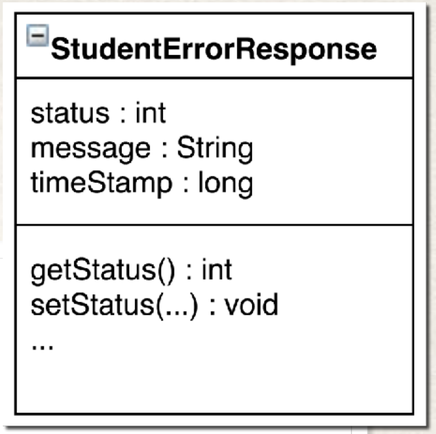
Step 4: Add exception handler method



File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());
```

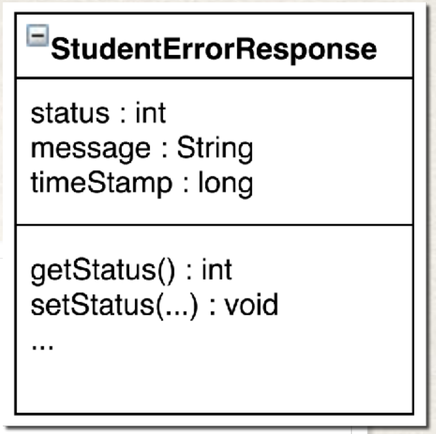
Step 4: Add exception handler method



File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());

    return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
  }

}
```

Step 4: Add exception handler method

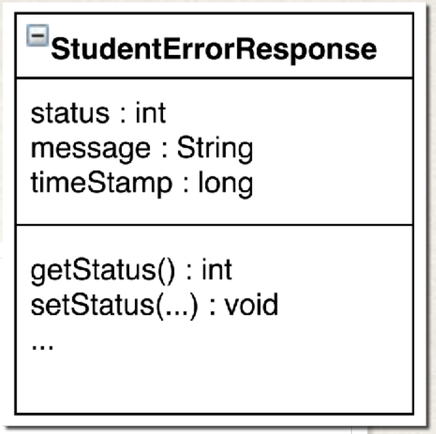


File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());

    return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
  }

}
```

Body

Step 4: Add exception handler method



File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());

    return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
  }

}
```

Body

Status code

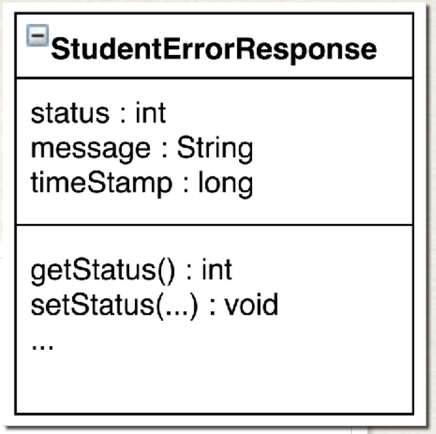
Step 4: Add exception handler method

File: StudentRestController.java

```
@RestController
@RequestMapping("/api")
public class StudentRestController {
  …

  @ExceptionHandler
  public ResponseEntity<StudentErrorResponse> handleException(StudentNotFoundException exc) {

    StudentErrorResponse error = new StudentErrorResponse();

    error.setStatus(HttpStatus.NOT_FOUND.value());
    error.setMessage(exc.getMessage());
    error.setTimeStamp(System.currentTimeMillis());

    return new ResponseEntity<>(error, HttpStatus.NOT_FOUND);
  }

}
```

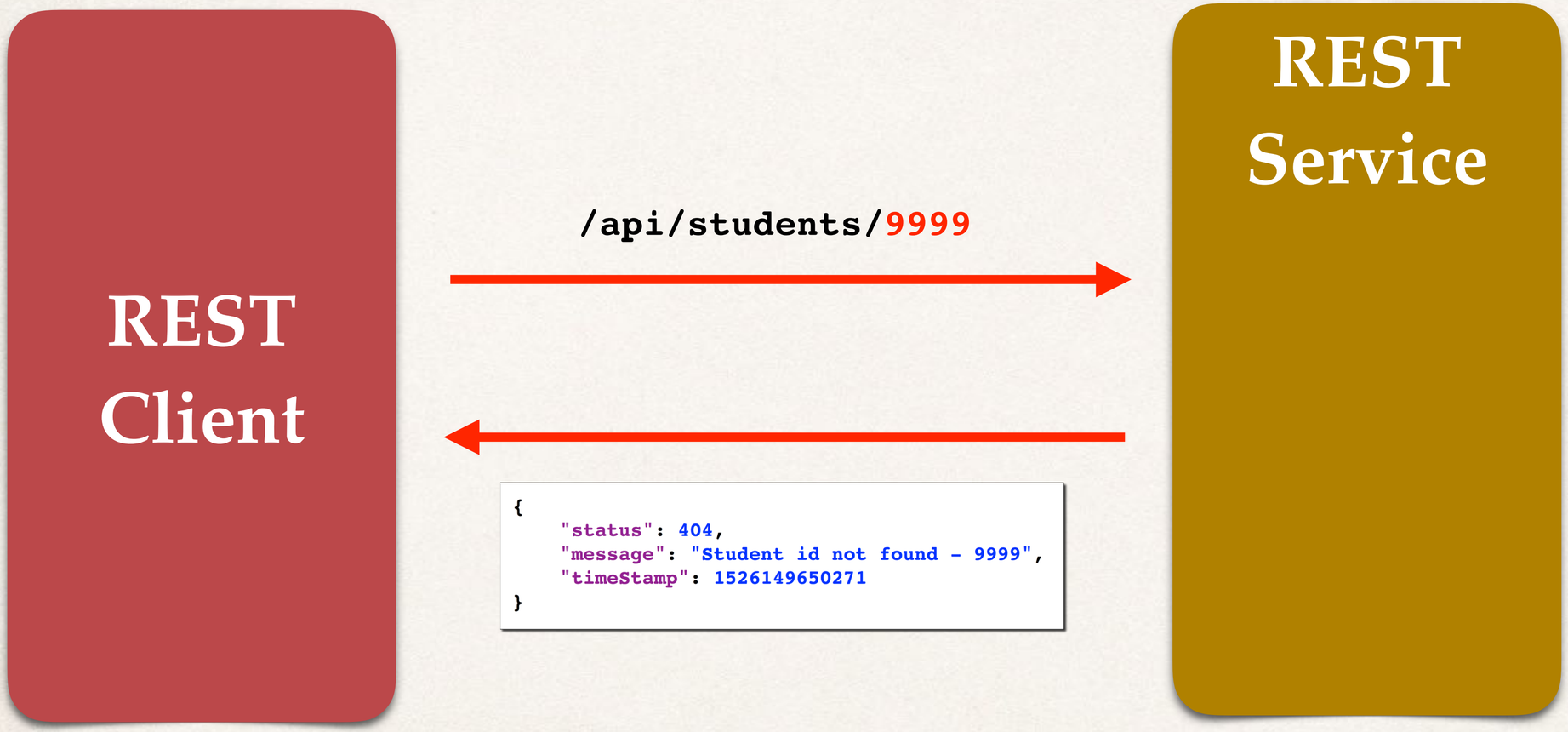
Body

Status code



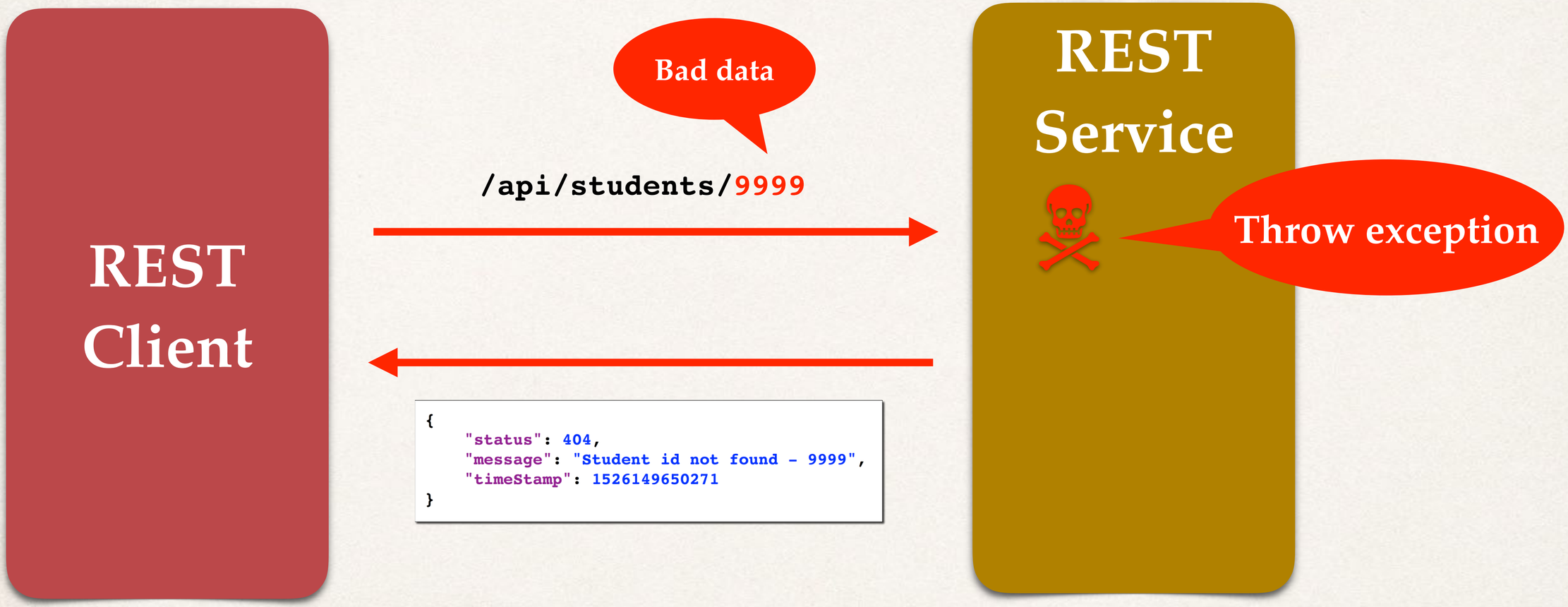

```
{
    "status": 404,
    "message": "Student id not found - 9999",
    "timeStamp": 1526149650271
}
```

Spring REST Exception Handling



Spring REST Exception Handling

Spring REST Exception Handling



Spring REST Exception Handling