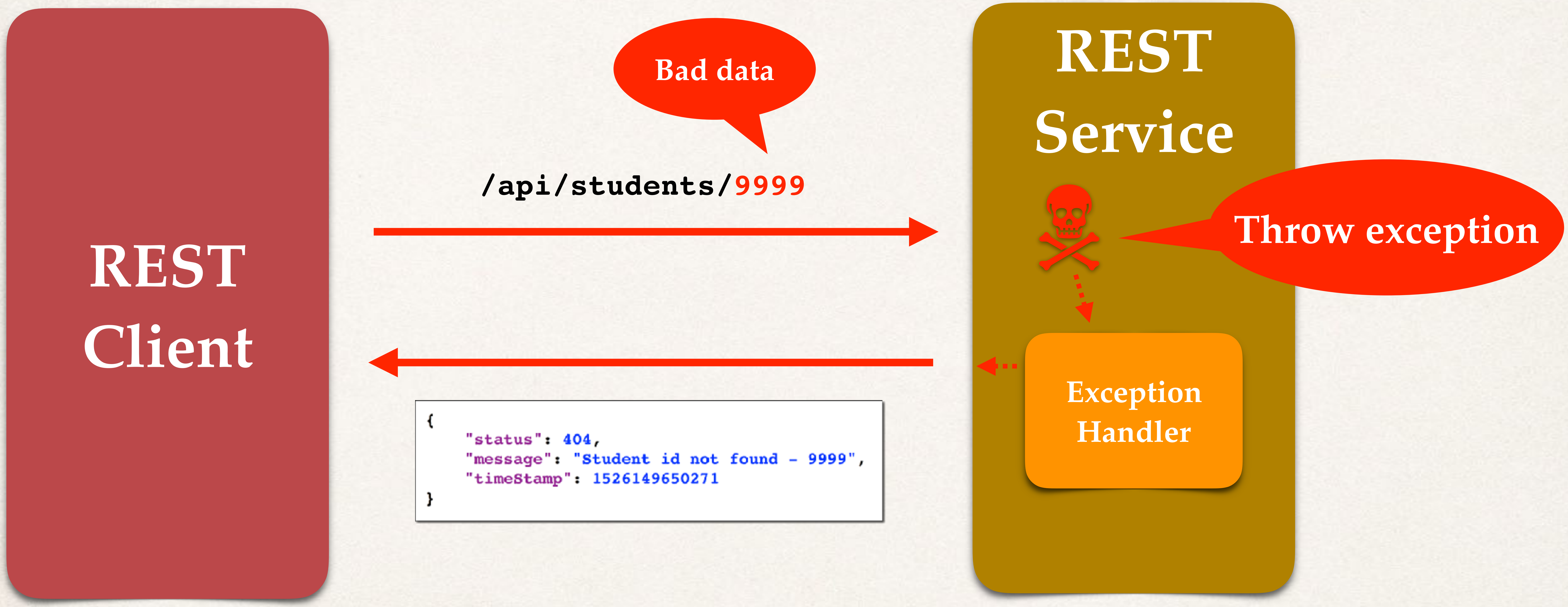
Spring REST - Global Exception Handling

Spring REST Exception Handling



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

luv2code

It works, but …

It works, but …

- Exception handler code is only for the specific REST controller

It works, but …

- Exception handler code is only for the specific REST controller

- Can't be reused by other controllers :-(

It works, but …

- Exception handler code is only for the specific REST controller
- Can't be reused by other controllers :-(

Large projects
will have
multiple controllers

It works, but …

- Exception handler code is only for the specific REST controller

- Can't be reused by other controllers :-(

Large projects will have multiple controllers

- We need **global** exception handlers

It works, but …

- Exception handler code is only for the specific REST controller

- Can't be reused by other controllers :-(

Large projects will have multiple controllers

- We need **global** exception handlers
  - Promotes reuse

It works, but …

- Exception handler code is only for the specific REST controller
- Can't be reused by other controllers :-(

Large projects will have multiple controllers

- We need **global** exception handlers
  - Promotes reuse
  - Centralizes exception handling

Spring @ControllerAdvice

Spring @ControllerAdvice

- `@ControllerAdvice` is similar to an interceptor / filter

Spring @ControllerAdvice

- `@ControllerAdvice` is similar to an interceptor / filter

- Pre-process requests to controllers

Spring @ControllerAdvice

- `@ControllerAdvice` is similar to an interceptor / filter
- Pre-process requests to controllers
- Post-process responses to handle exceptions

Spring @ControllerAdvice

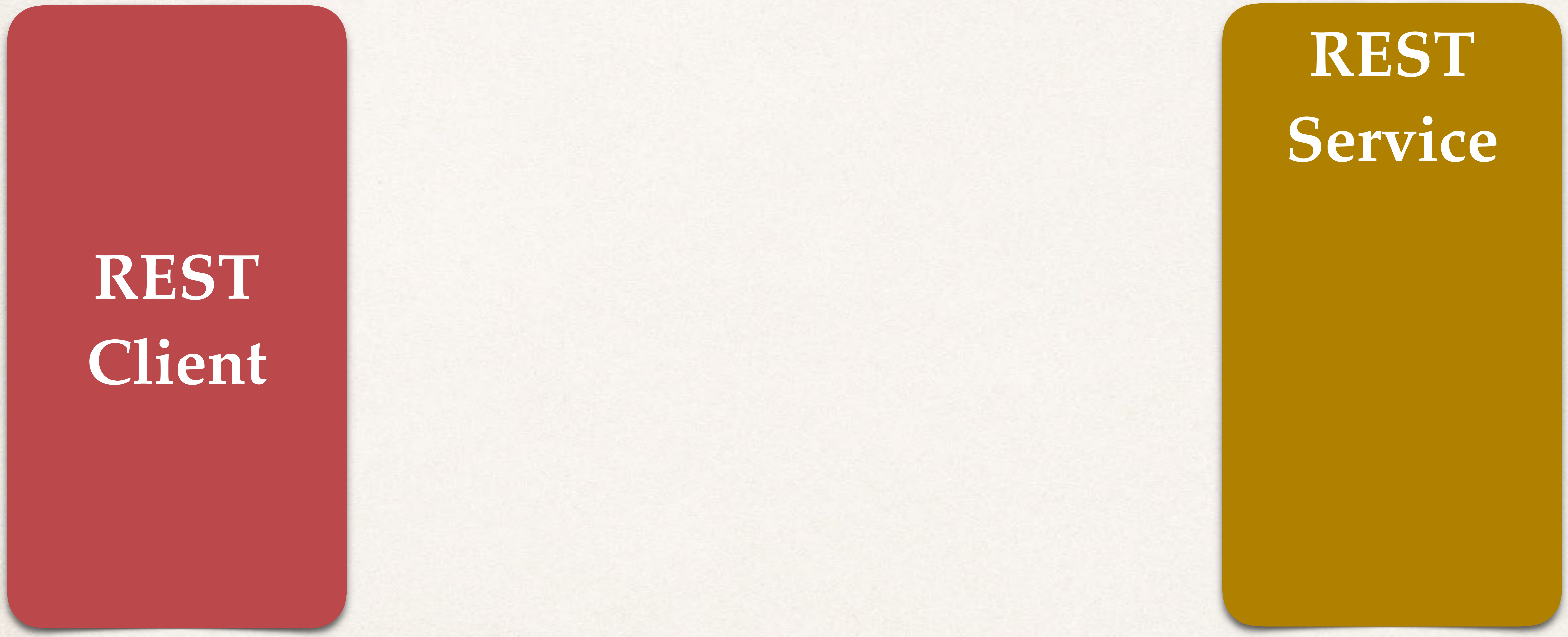
- `@ControllerAdvice` is similar to an interceptor / filter
- Pre-process requests to controllers
- Post-process responses to handle exceptions
- Perfect for global exception handling

Spring @ControllerAdvice

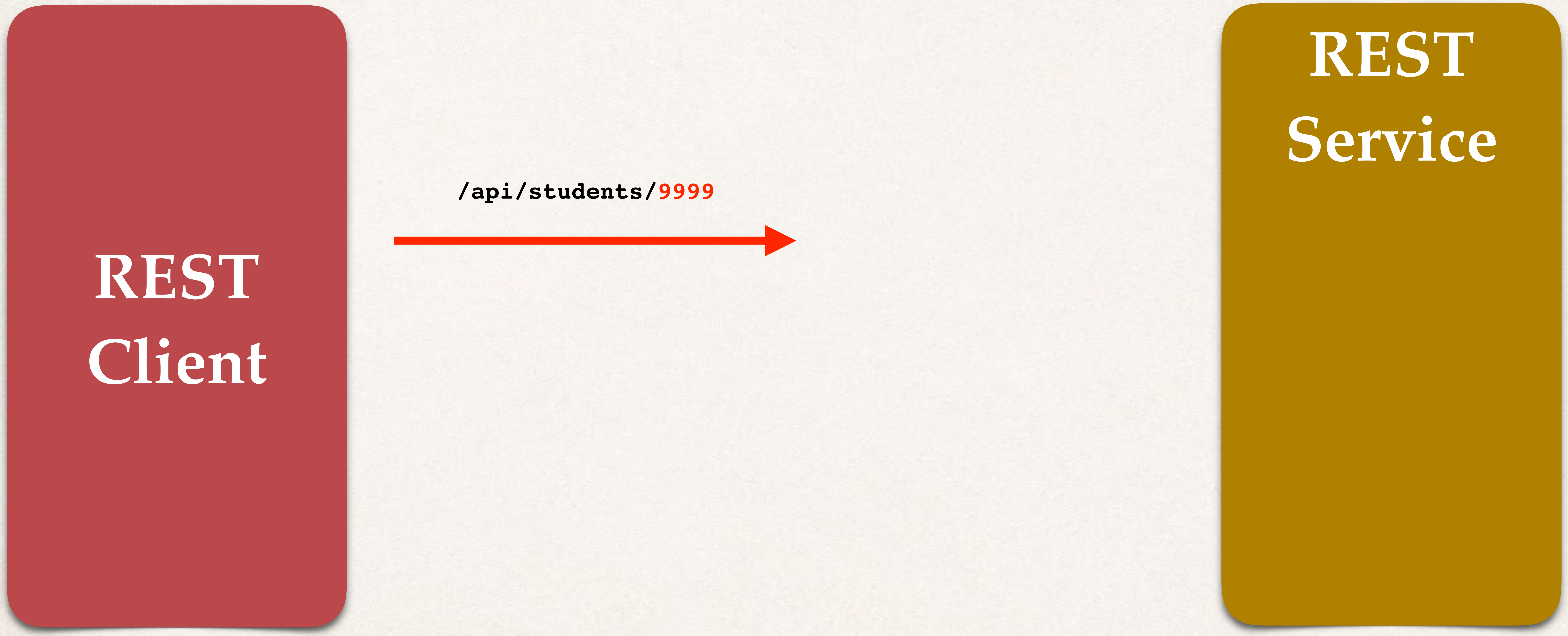
- `@ControllerAdvice` is similar to an interceptor / filter
- Pre-process requests to controllers
- Post-process responses to handle exceptions
- Perfect for global exception handling

Real-time use of AOP

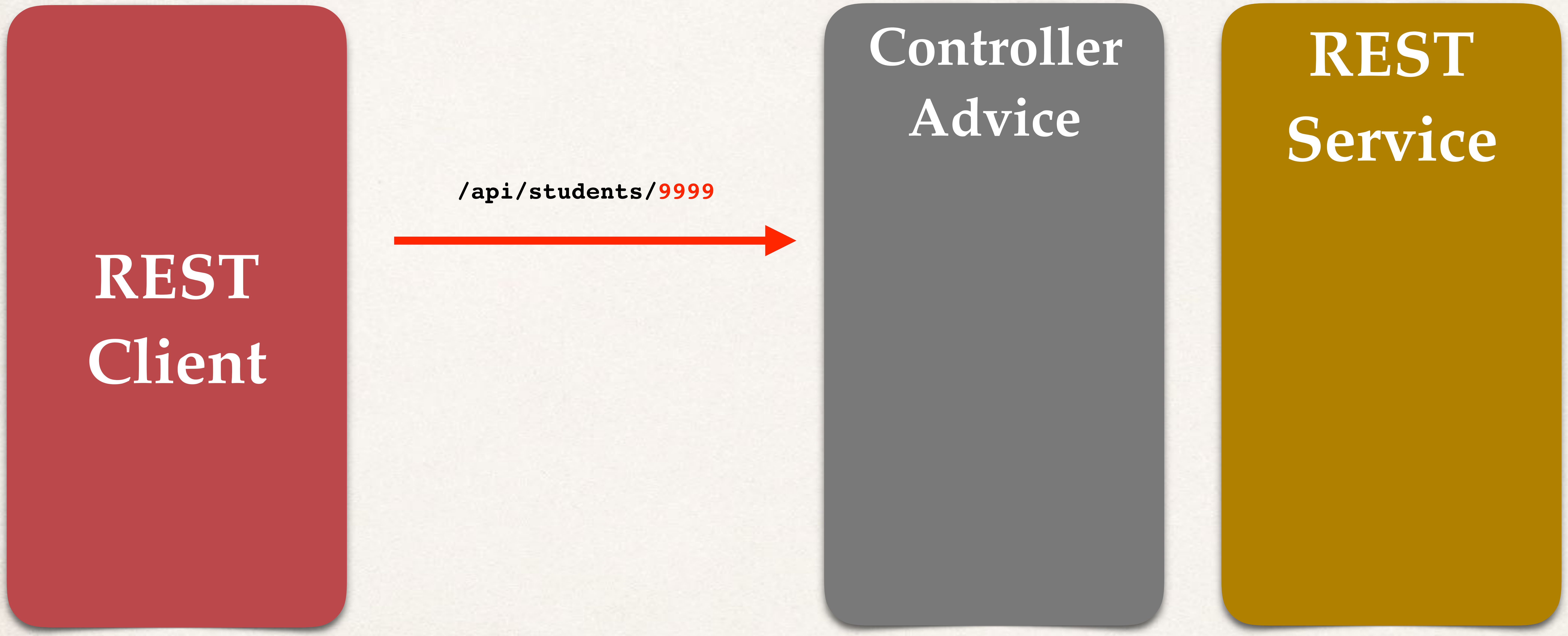
Spring REST Exception Handling



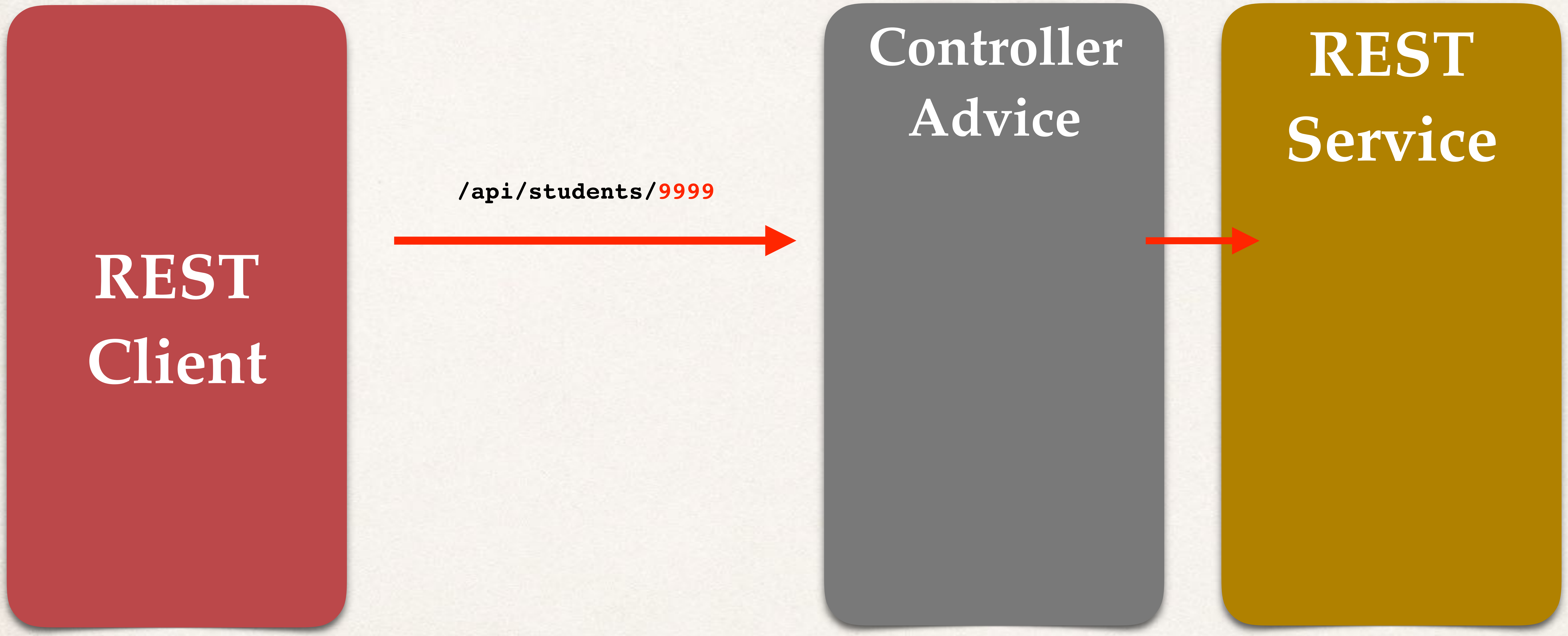
Spring REST Exception Handling



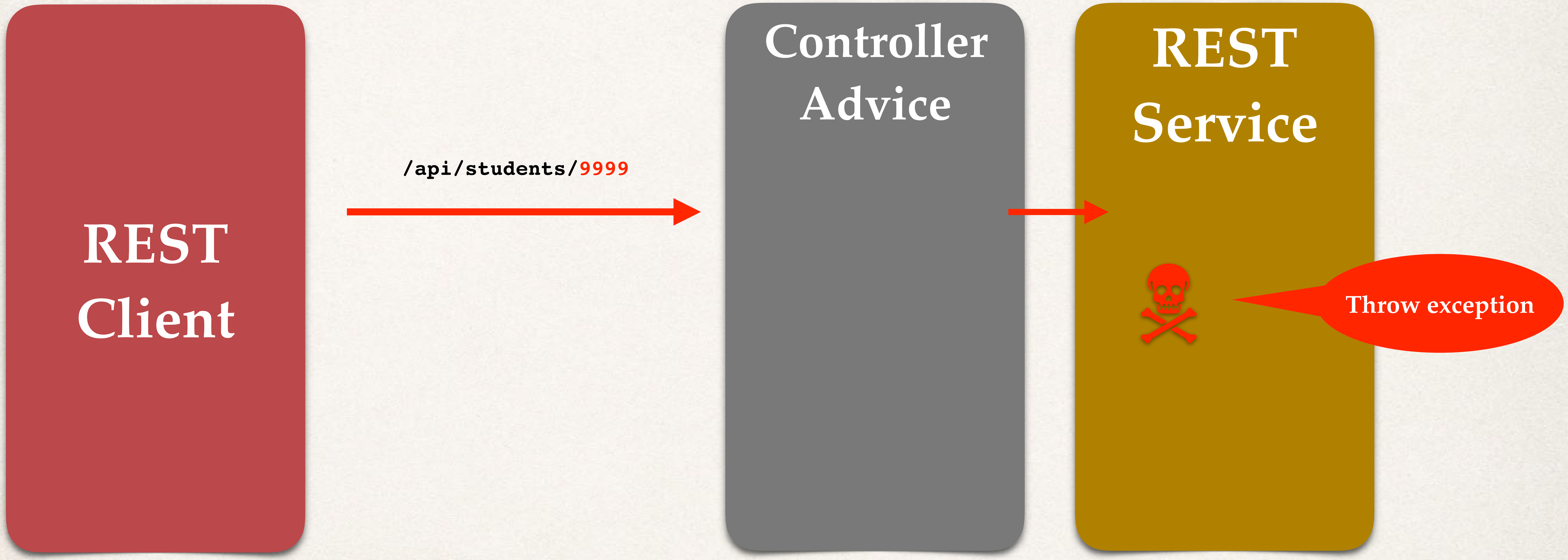
Spring REST Exception Handling



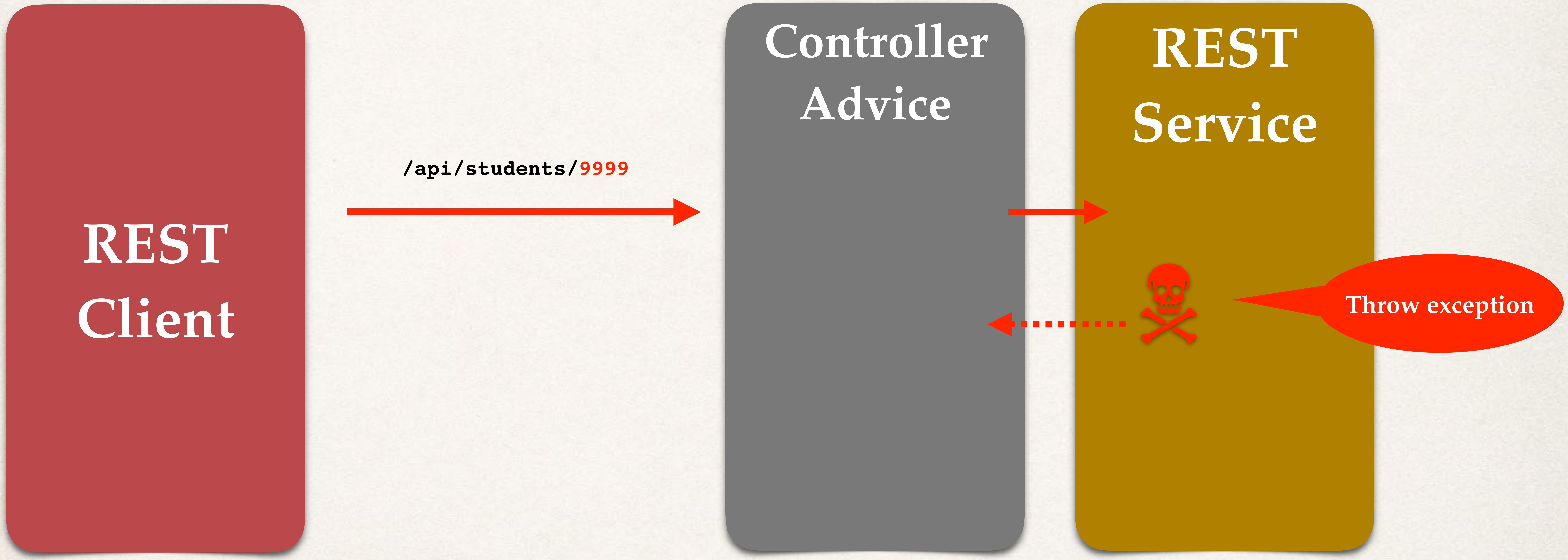
Spring REST Exception Handling



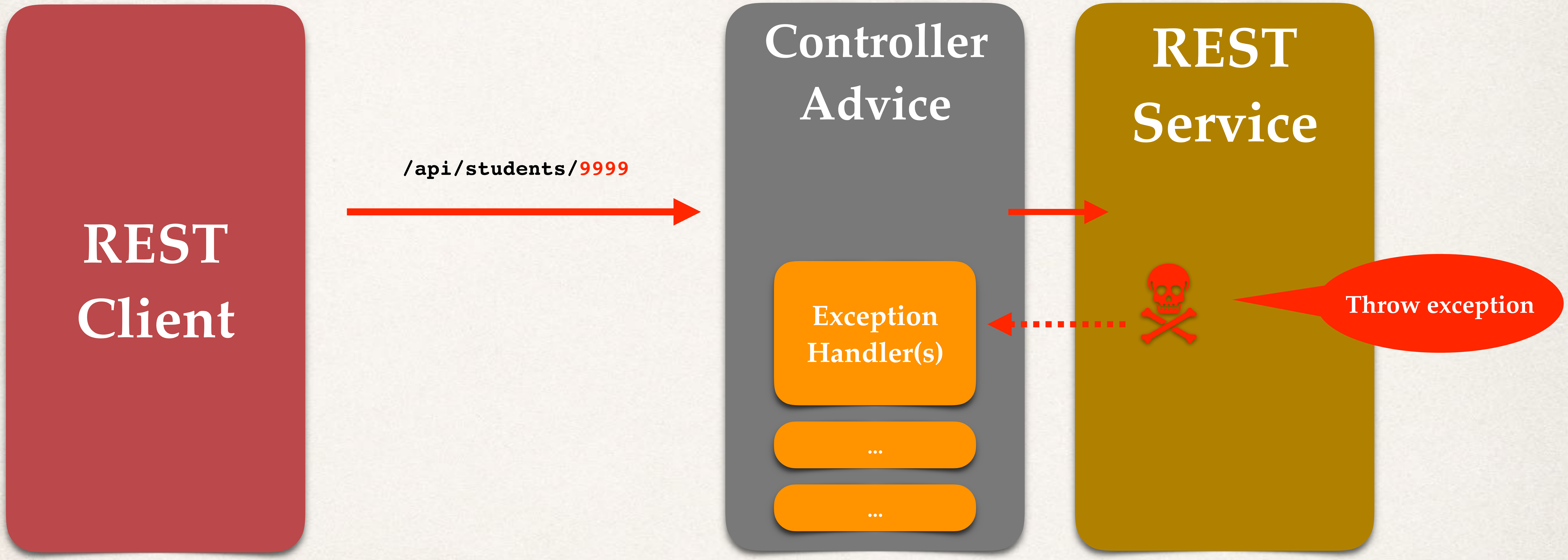
Spring REST Exception Handling



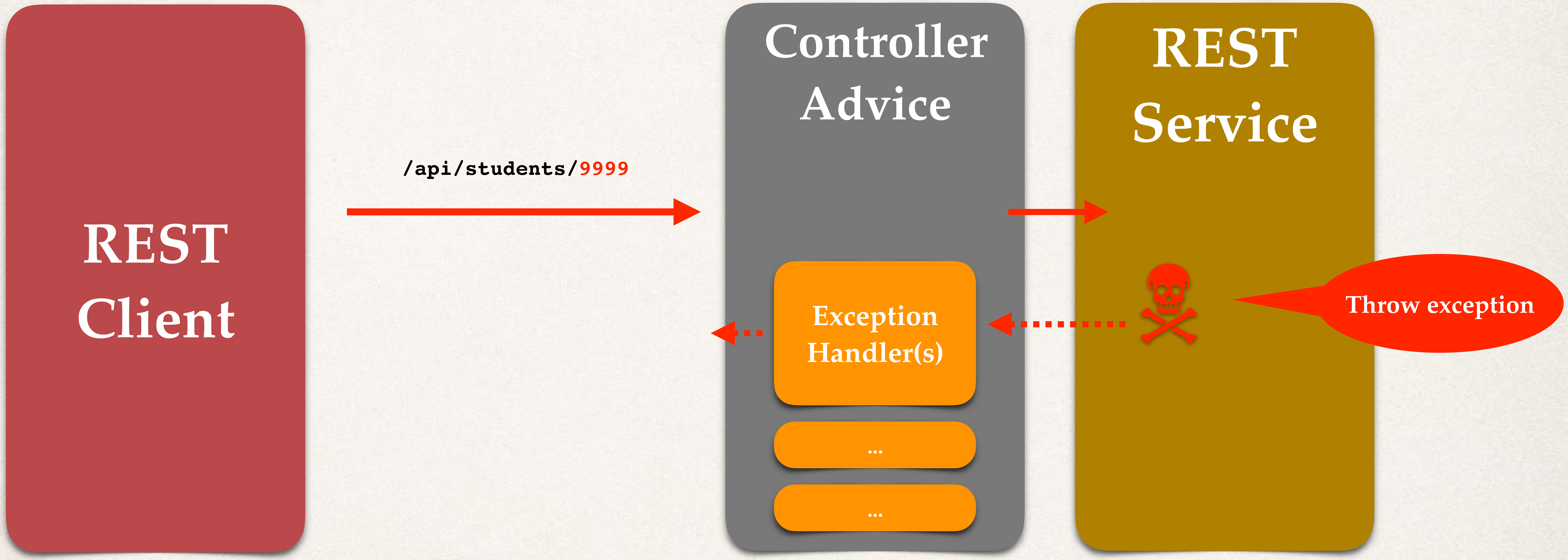
Spring REST Exception Handling



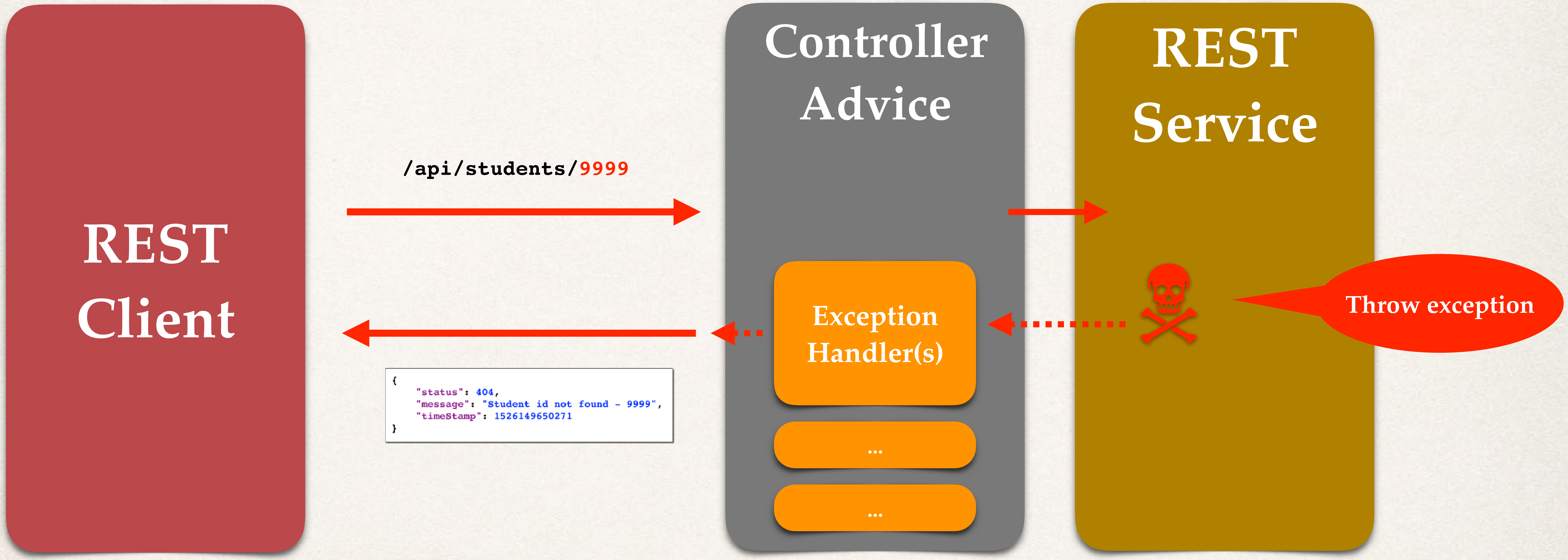
Spring REST Exception Handling



Spring REST Exception Handling



Spring REST Exception Handling



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

Development Process

Step-By-Step

Development Process

Step-By-Step

1. Create new @ControllerAdvice

Development Process

Step-By-Step

1. Create new @ControllerAdvice

2. Refactor REST service … remove exception handling code

Development Process

Step-By-Step

1. Create new @ControllerAdvice

2. Refactor REST service … remove exception handling code

3. Add exception handling code to @ControllerAdvice

Step 1: Create new @ControllerAdvice

File: StudentRestExceptionHandler.java

Step 1: Create new @ControllerAdvice

File: StudentRestExceptionHandler.java

```
@ControllerAdvice
public class StudentRestExceptionHandler {


  …


}
```

Step 2: Refactor - remove exception handling

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 2: Refactor - remove exception handling

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 2: Refactor - remove exception handling

File: StudentRestController.java

Remove this code

```
@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 3: Add exception handler to @ControllerAdvice

File: StudentRestExceptionHandler.java

```
@ControllerAdvice
public class StudentRestExceptionHandler {















}
```

Step 3: Add exception handler to @ControllerAdvice

File: StudentRestExceptionHandler.java

```
@ControllerAdvice
public class StudentRestExceptionHandler {

  @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);
  }

}
```

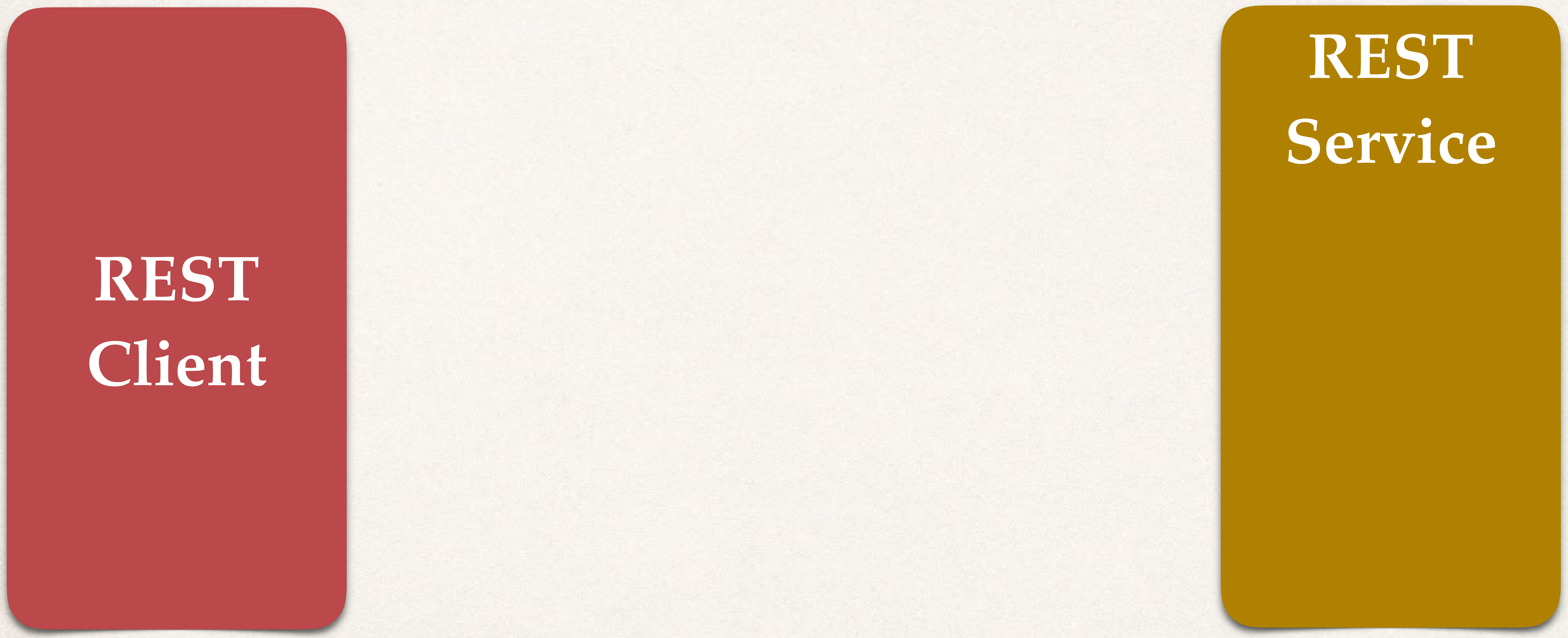
Step 3: Add exception handler to @ControllerAdvice

File: StudentRestExceptionHandler.java

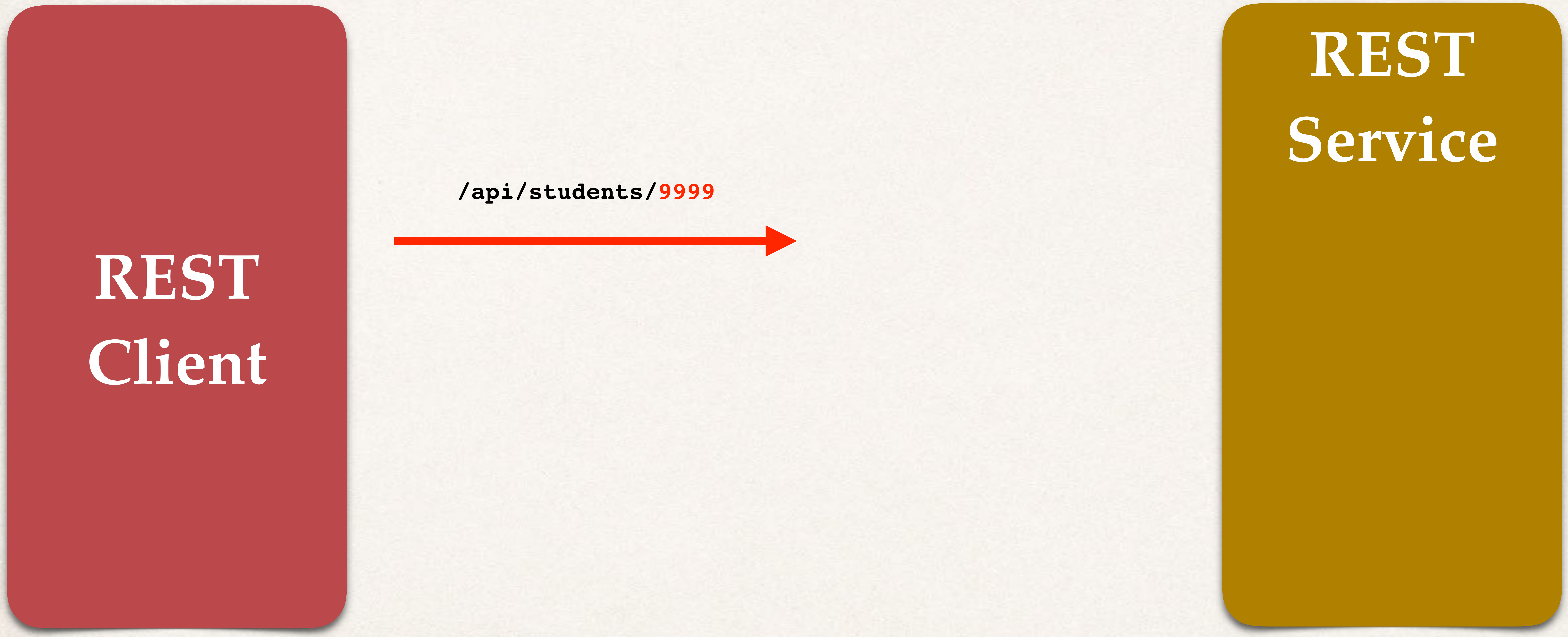
Same code as before

```
@ControllerAdvice
public class StudentRestExceptionHandler {

  @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);
  }

}
```

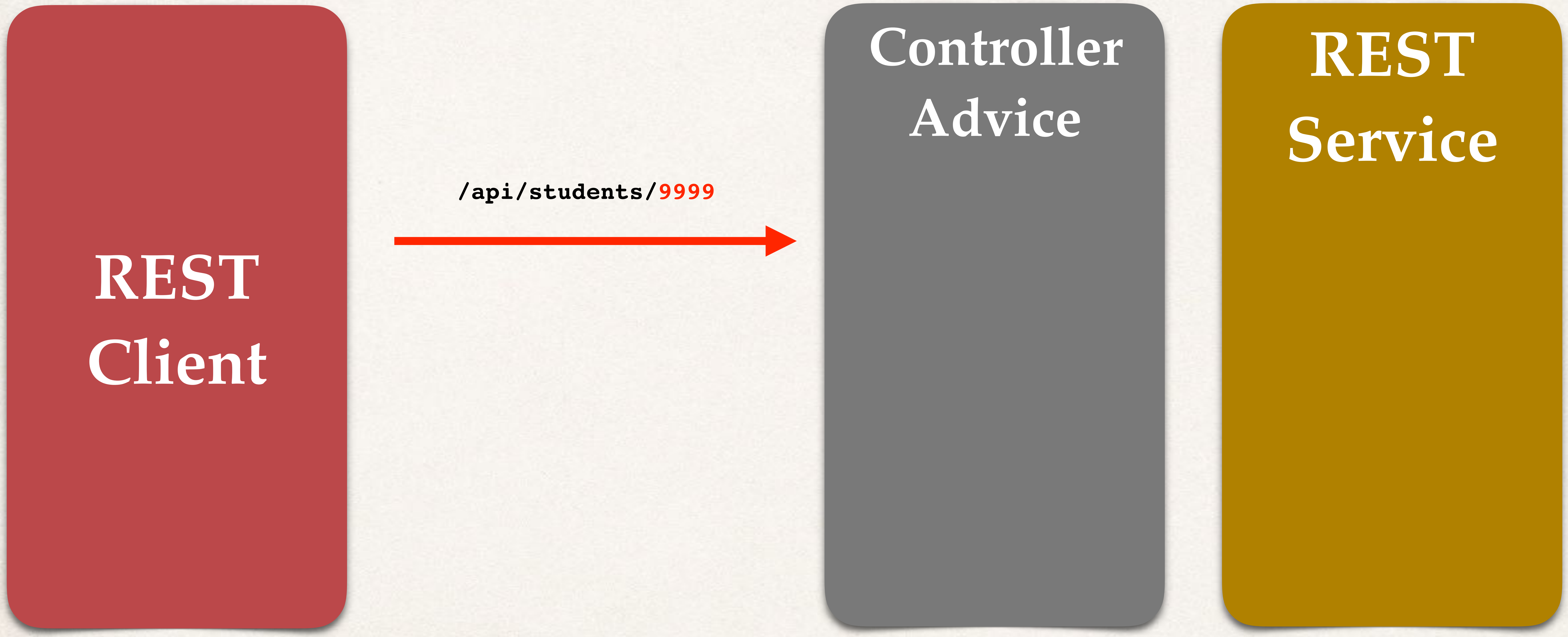
Spring REST Exception Handling



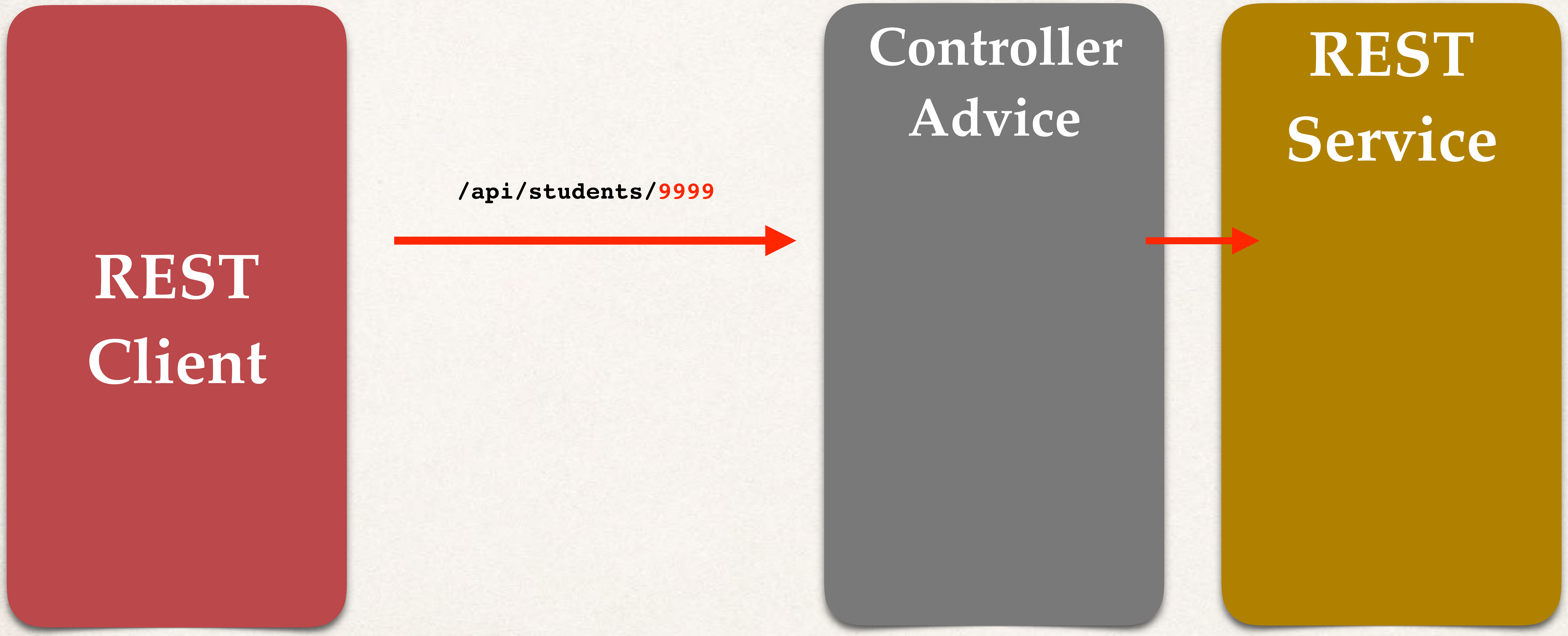
Spring REST Exception Handling



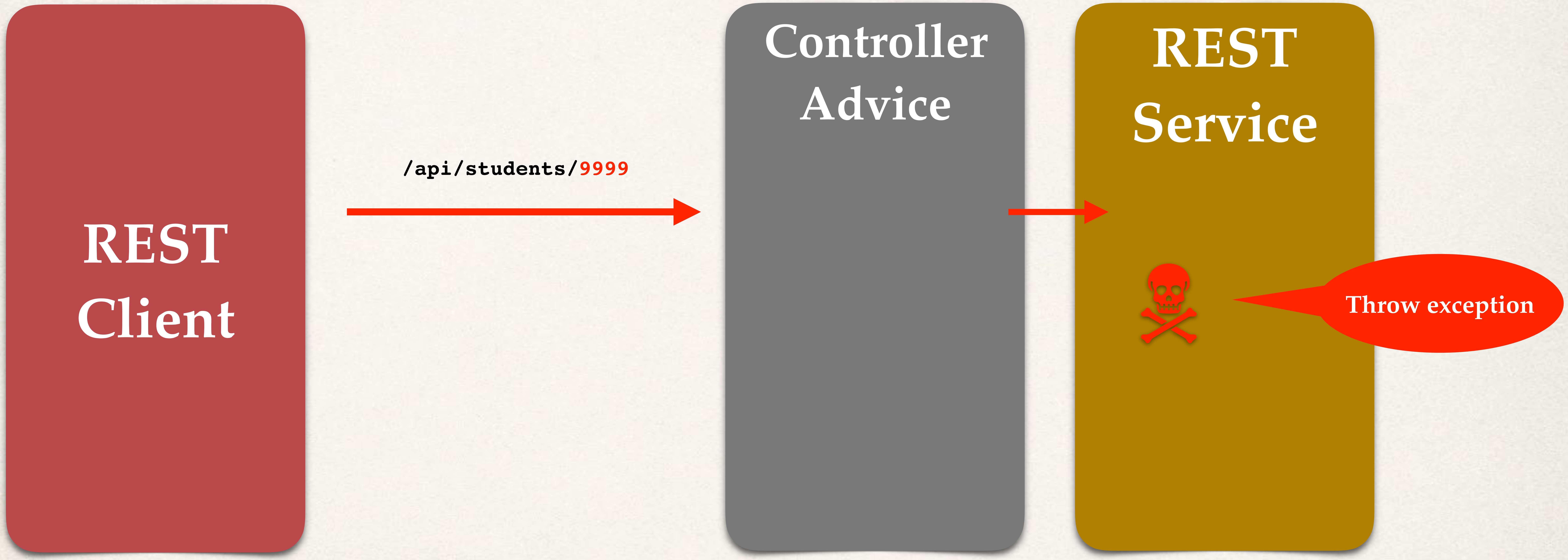
Spring REST Exception Handling



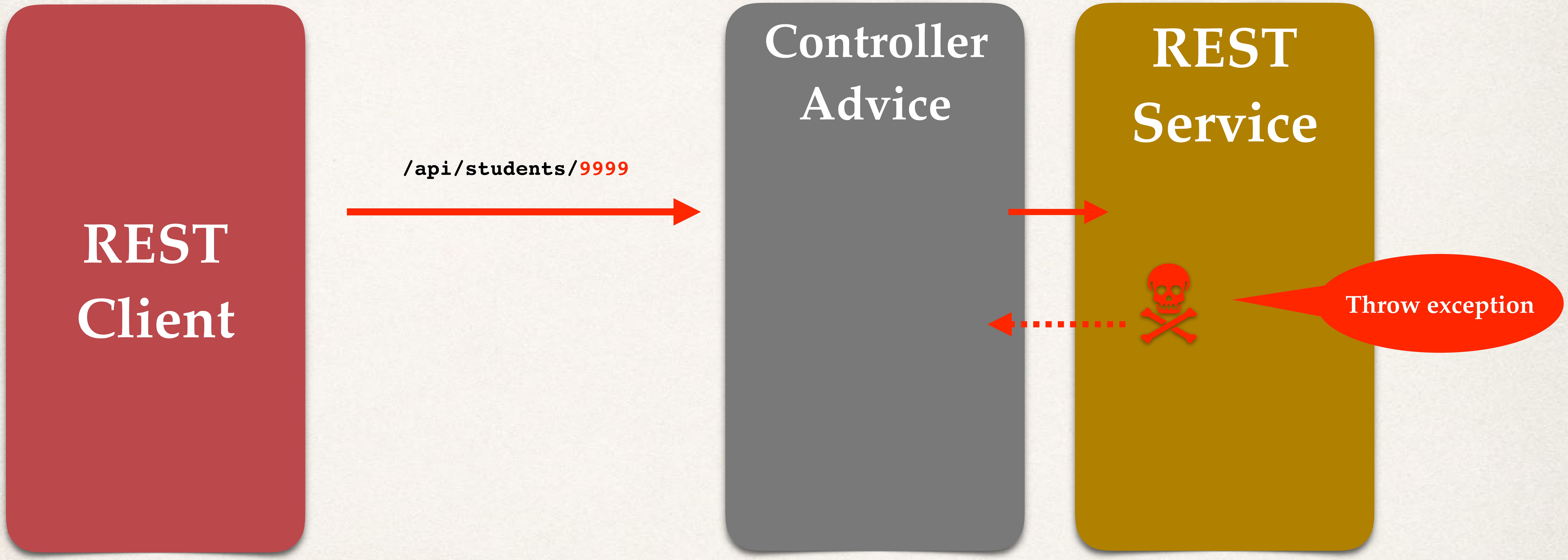
Spring REST Exception Handling



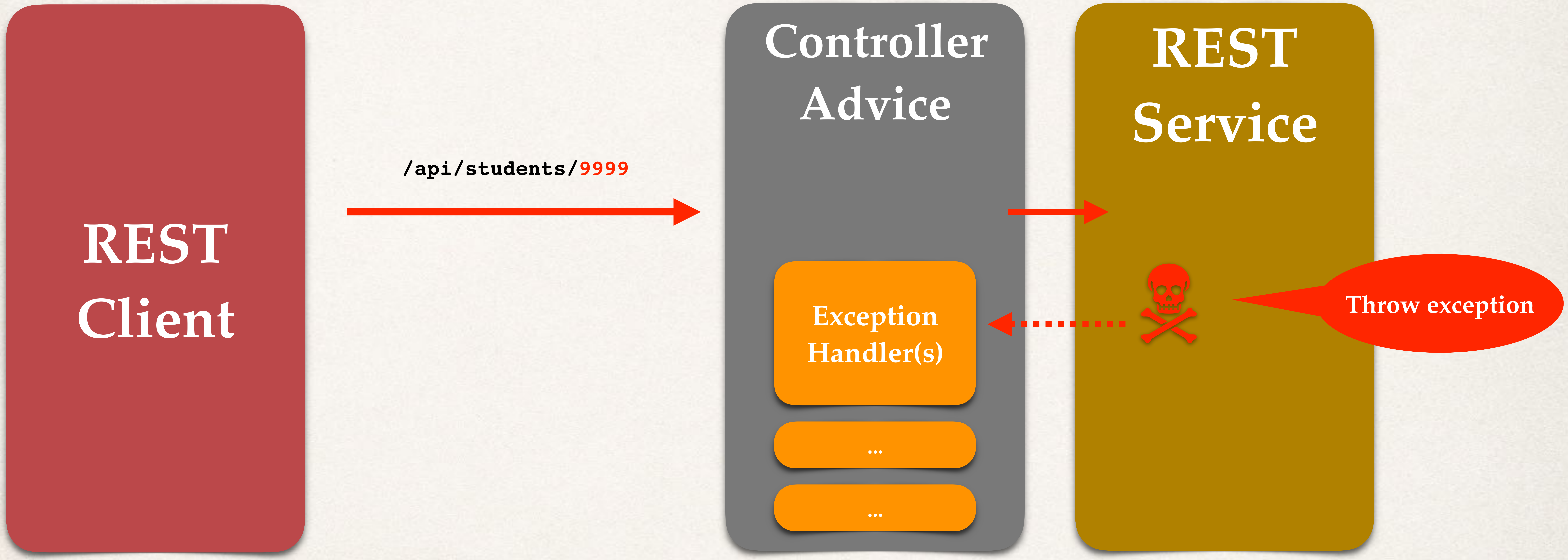
Spring REST Exception Handling



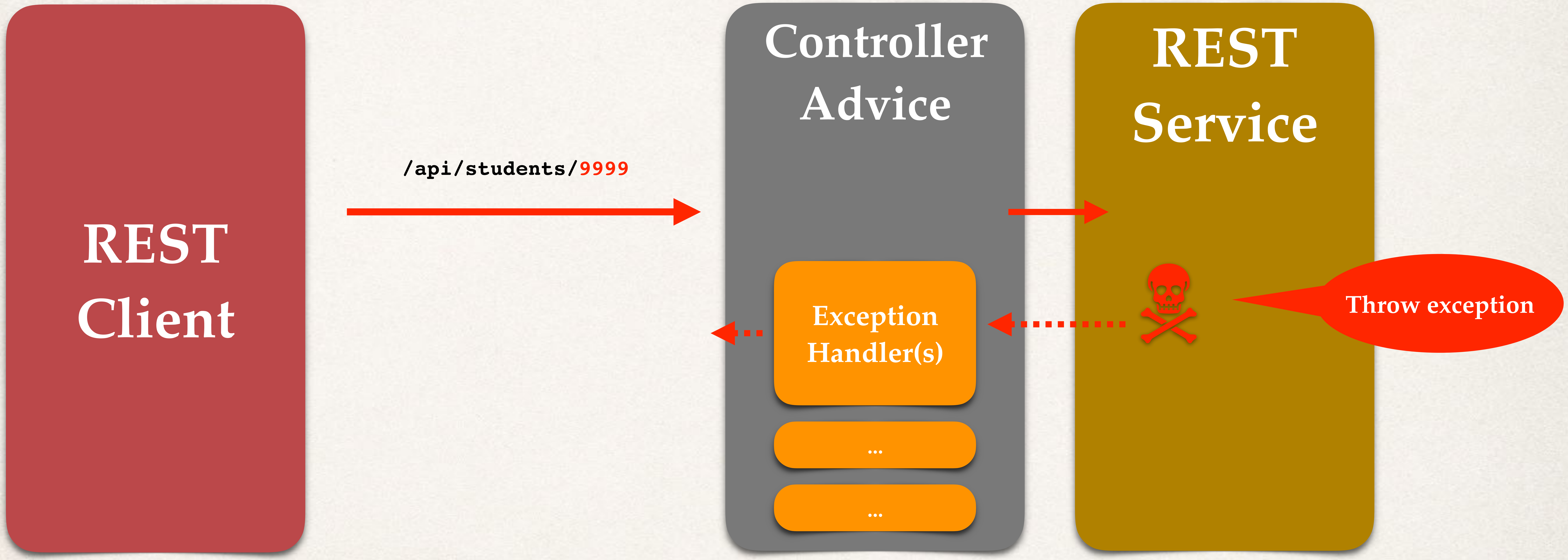
Spring REST Exception Handling



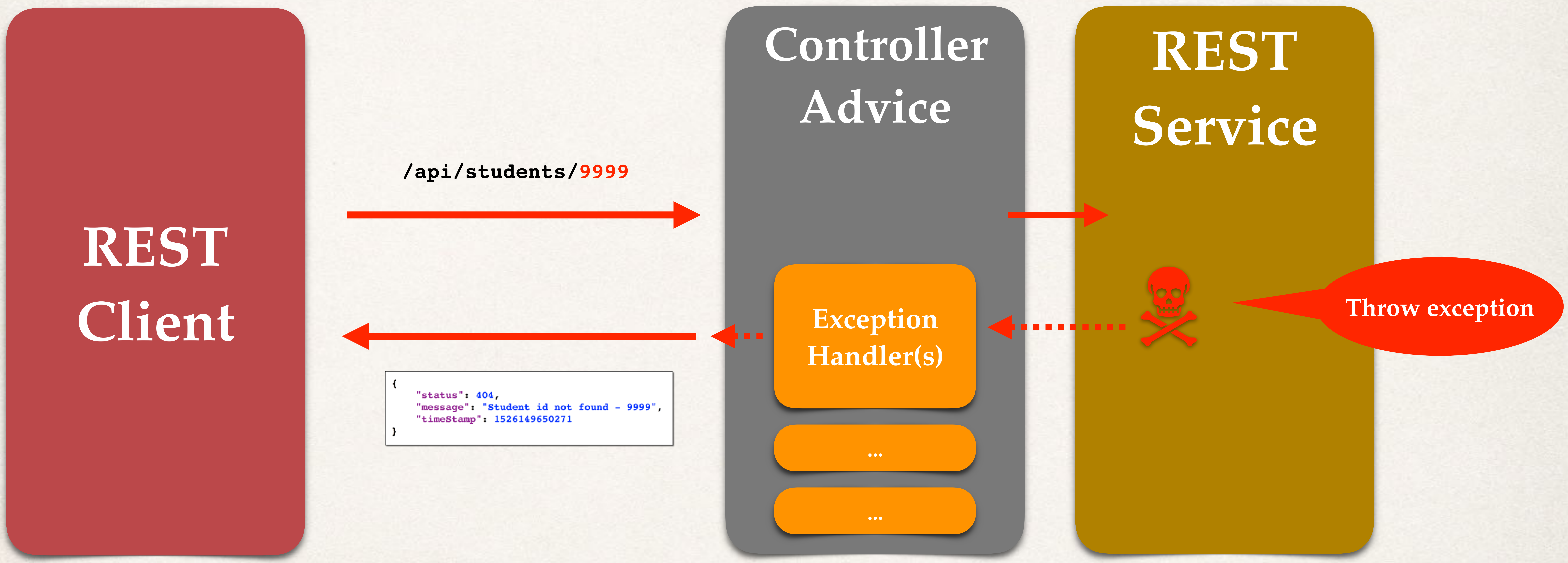
Spring REST Exception Handling



Spring REST Exception Handling



Spring REST Exception Handling



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