Agenda:

~ SpringBoot REST Path Variables

~ SpringBoot REST Exception Handling

* Path Variables :
- Retrieve a single student by id.

GET - /api/students/{studentId}   - Retrieve a single student

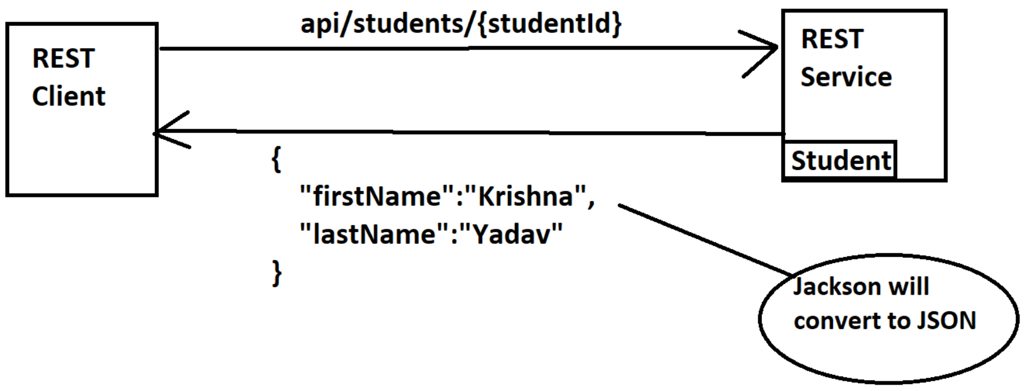
(Path Variable)

/api/students/0

/api/students/1

/api/students/2

Spring REST Service



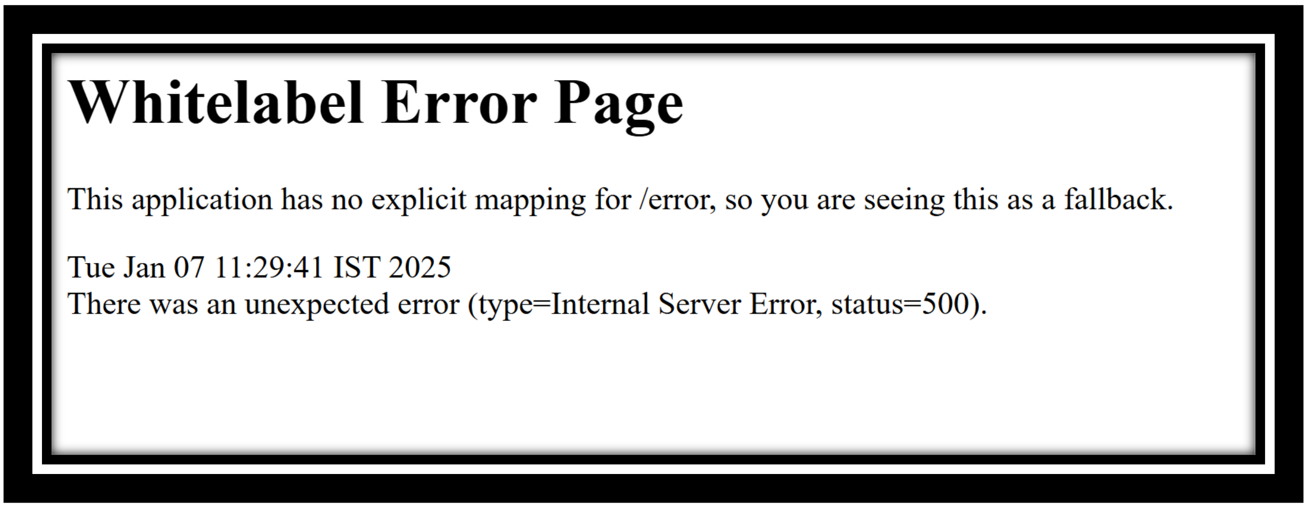
Development Process:

Add request mapping to spring REST service

* Bind path variable to method parameter using @PathVariable

```
@GetMapping( "/students/{studentId}" )
public Student getStudent(@PathVariable int studentId)
{
List<Student> theStudents = new ArrayList<>();
//populate theStudents
}
```

~ SpringBoot REST Exception Handling



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Tue Jan 07 11:29:41 IST 2025
There was an unexpected error (type=Internal Server Error, status=500).

What we want?

- Handle the exception and return error as JSON.

Development Process:

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

Step1> 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

-
  ```
  package com.flynaut.crud.rest;

  public class StudentErrorResponse {
      private int status;
      private String message;
      private long timeStamp;

      public StudentErrorResponse(){}

      public StudentErrorResponse(int status, String
  message, long timeStamp) {
          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;
      }
  }
  ```

Step2> Create custom exception class

```
package com.flynaut.crud.rest;

public class StudentNotFoundException extends
RuntimeException{

    public StudentNotFoundException(String message) {
        super(message); // It will call the super class
constructor
    }

//    public StudentNotFoundException(String message,
Throwable cause) {
//        super(message, cause);
//    }
//
//    public StudentNotFoundException(Throwable cause) {
//        super(cause);
//    }
}
```

Step3> Update REST service to throw an exception if student not found

```
package com.flynaut.crud.rest;

import com.flynaut.crud.entity.Student;
import jakarta.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 java.util.ArrayList;
import java.util.List;

@RestController
@RequestMapping("/api")
public class StudentRestController {
    private List<Student> theStudents;

    //define @PostContruct to load data ... for once
    @PostConstruct
    public void loadData(){
        theStudents = new ArrayList<>();
        theStudents.add(new Student("Krishna","Yadav"));
        theStudents.add(new Student("Gopal","Yadav"));
        theStudents.add(new Student("Govind","Yadav"));
    }

    //define an endpoint for "/students" - return the list of
students
    @GetMapping("/students")
    public List<Student> getStudents(){
        return theStudents;
    }

    // define an endpoint "/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("StudentId not
found- "+studentId);
        }
        return theStudents.get(studentId);
    }
}
```

Step4> Add an exception handler method using @ExceptionHandler

<u>* Define exception handler methods with @ExceptionHandler.</u>

- <u>Exception handler will return a ResponseEntity</u>
- <u>ResponseEntity is a wrapper for the HTTP response object.</u>

```
• package com.flynaut.crud.rest;

  import com.flynaut.crud.entity.Student;
  import jakarta.annotation.PostConstruct;
  import org.springframework.http.HttpStatus;
  import org.springframework.http.ResponseEntity;
  import org.springframework.web.bind.annotation.*;

  import java.util.ArrayList;
  import java.util.List;

  @RestController
  @RequestMapping("/api")
  public class StudentRestController {
      private List<Student> theStudents;
```

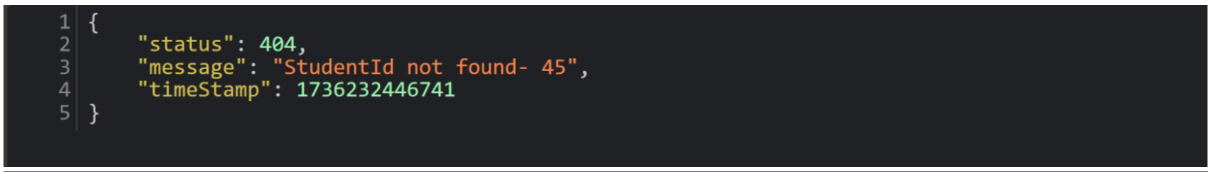
```
    //define @PostContruct to load data ... for once
    @PostConstruct
    public void loadData(){
        theStudents = new ArrayList<>();
        theStudents.add(new Student("Krishna","Yadav"));
        theStudents.add(new Student("Gopal","Yadav"));
        theStudents.add(new Student("Govind","Yadav"));
    }

    @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);
    }

    //define an endpoint for "/students" - return the
list of students
    @GetMapping("/students")
    public List<Student> getStudents(){
        return theStudents;
    }

    // define an endpoint "/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("StudentId
not found- "+studentId);
        }
        return theStudents.get(studentId);
    }
}
```

O/P:



```
{
    "status": 404,
    "message": "StudentId not found- 45",
    "timeStamp": 1736232446741
}
```

Task: http://localhost:8080/api/students/ueue
how to handle this exception.

Global Exception Handler* - (@ControllerAdvice)
@ResponseBody