springboot integration tests example

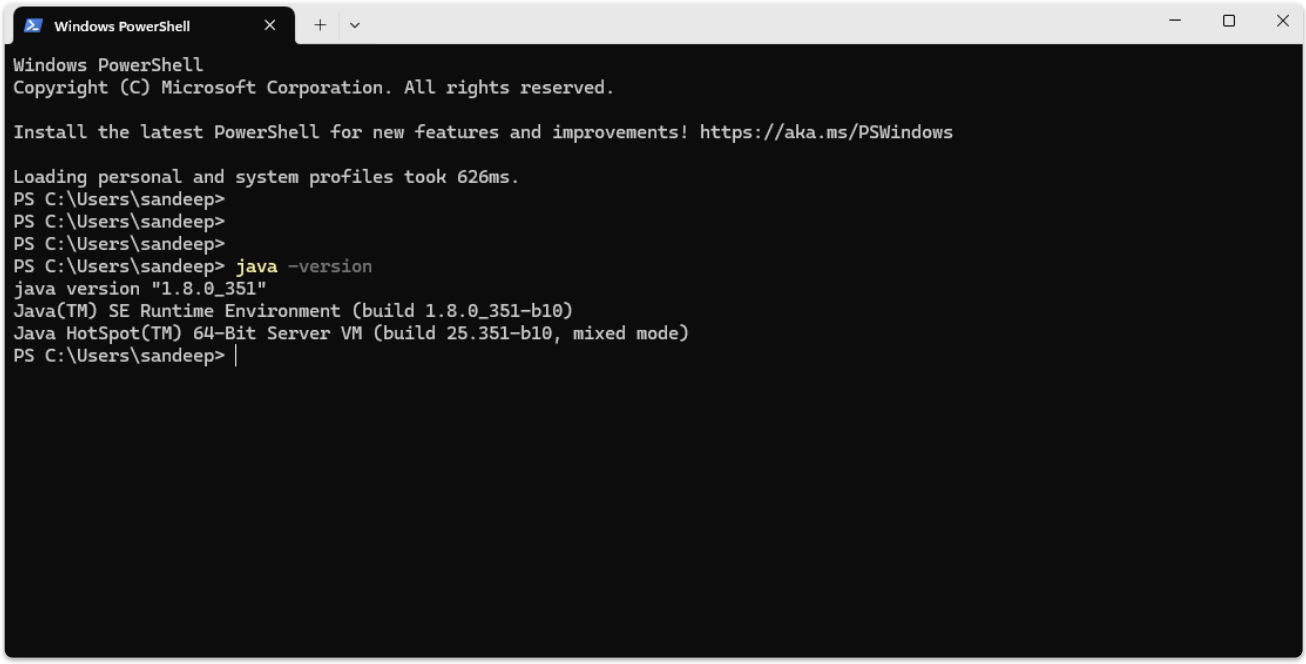
Blogpost : [Writing Integration Tests for Rest Services with Spring Boot | Spring Boot Tutorial]

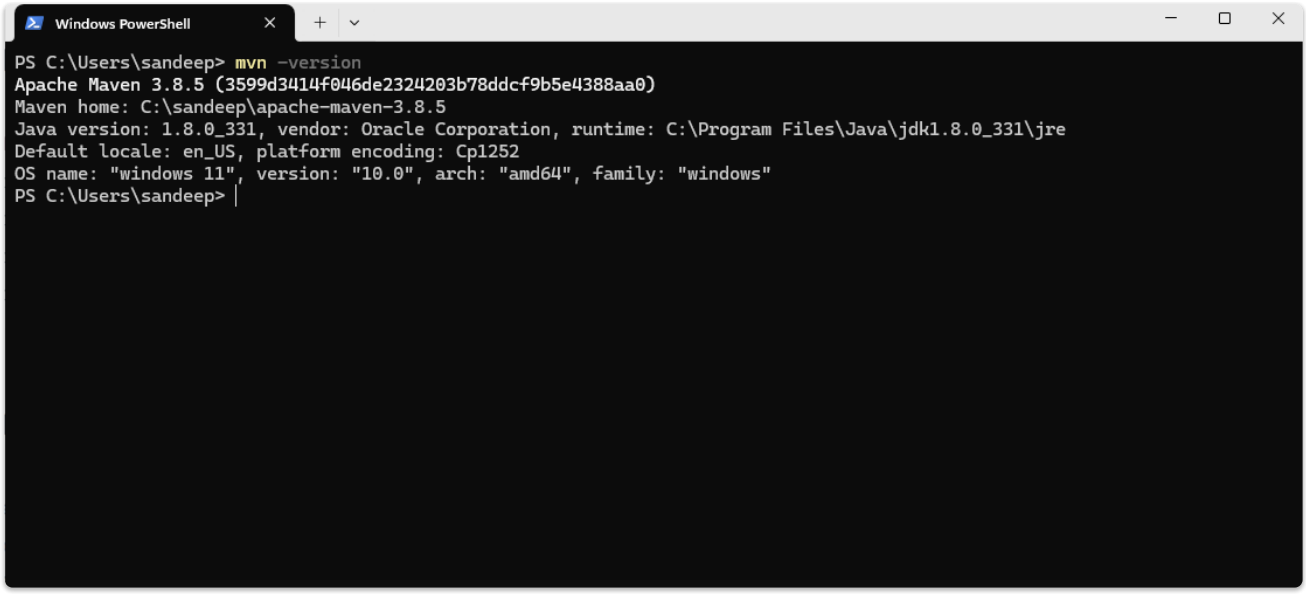
What we will learn in this blogposts

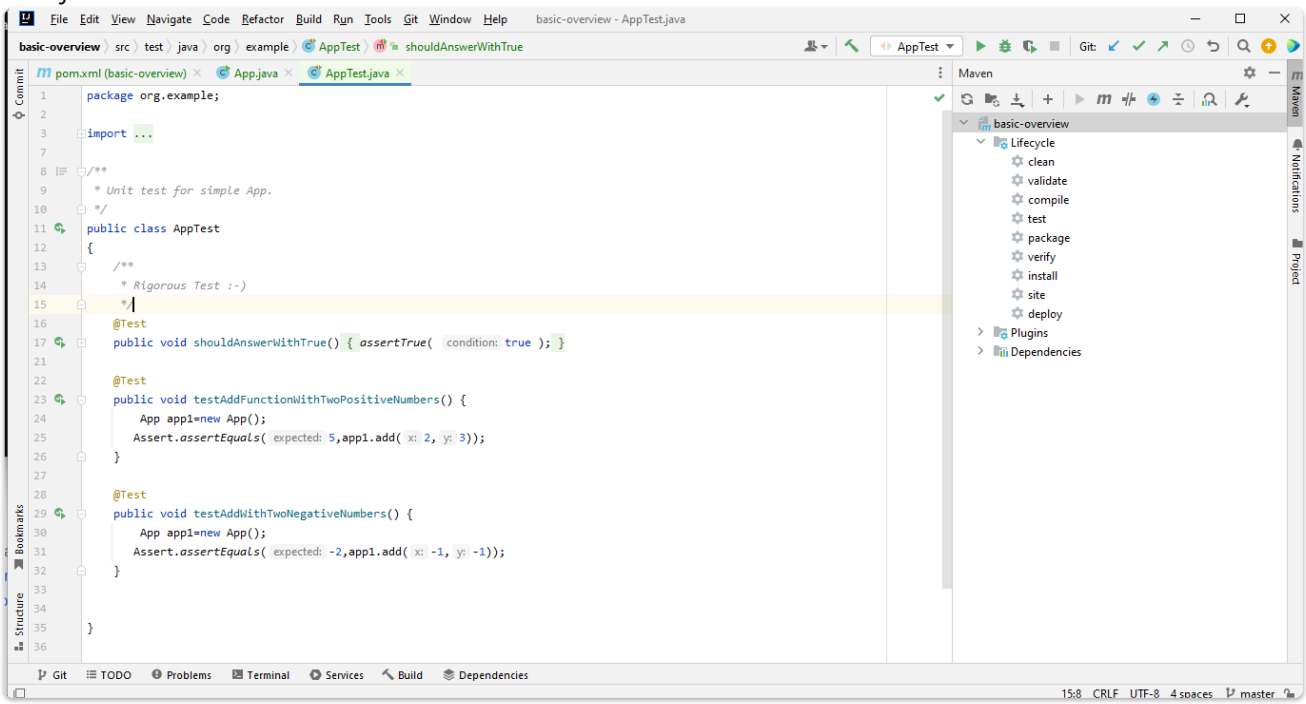
- What is Integration Testing?
- How to create a Get REST Service for retrieving the courses that a student registered for?
- How to write a integration test for Get REST Service?
- How to create a Post REST Service for registering a course for student?
- How to write a integration test for the POST Service?

Prerequisites

1. JDK 8
2. MAVEN
3. Intellj
4. git
5. Java 8



6. 

7. Intellj


Overview

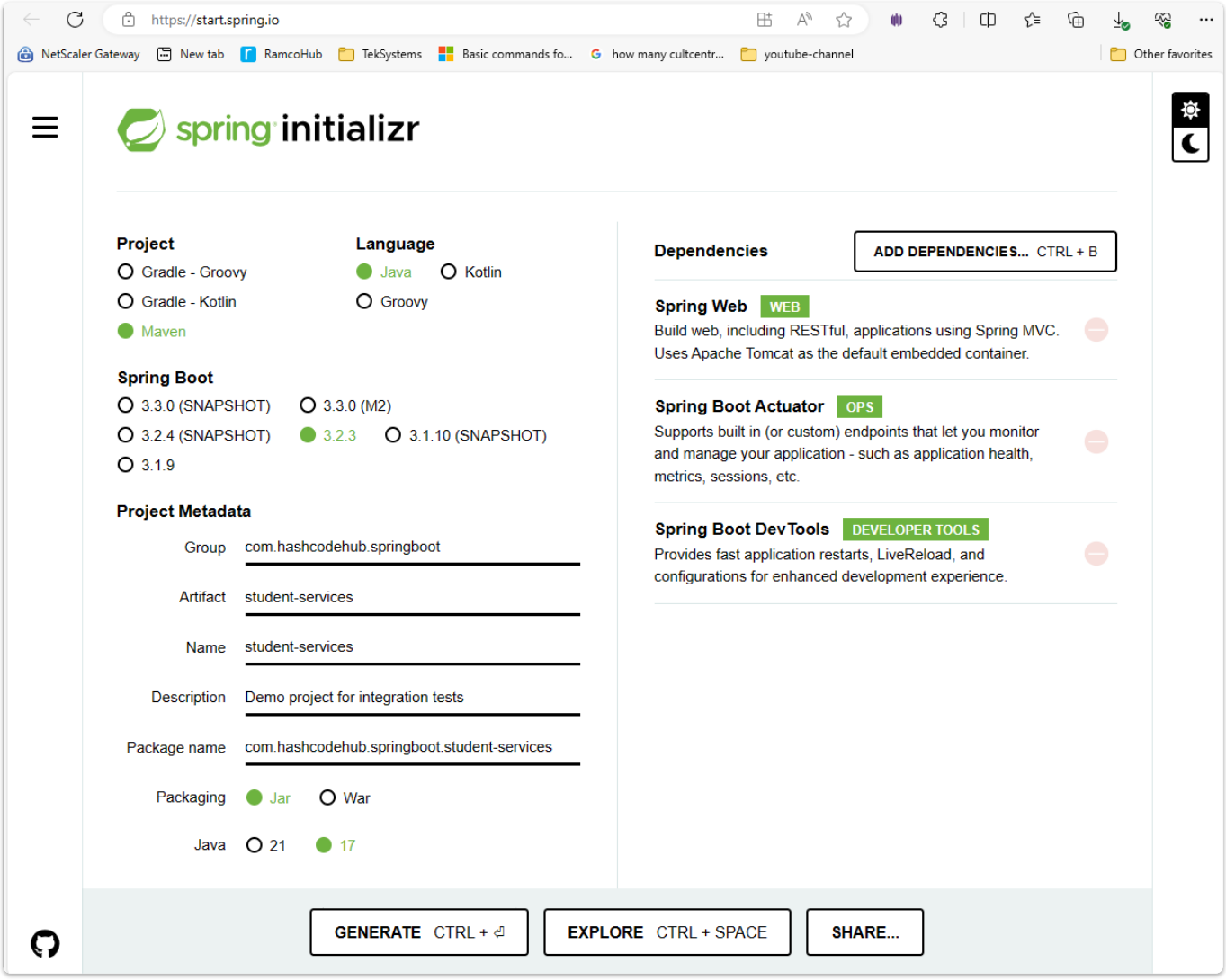
1. Create a project using spring initiazer
2. Push the project to github
3. Create business layer for get API call
4. Create controller for get API call
5. Create POST method similarly
6. Create API Integration tests

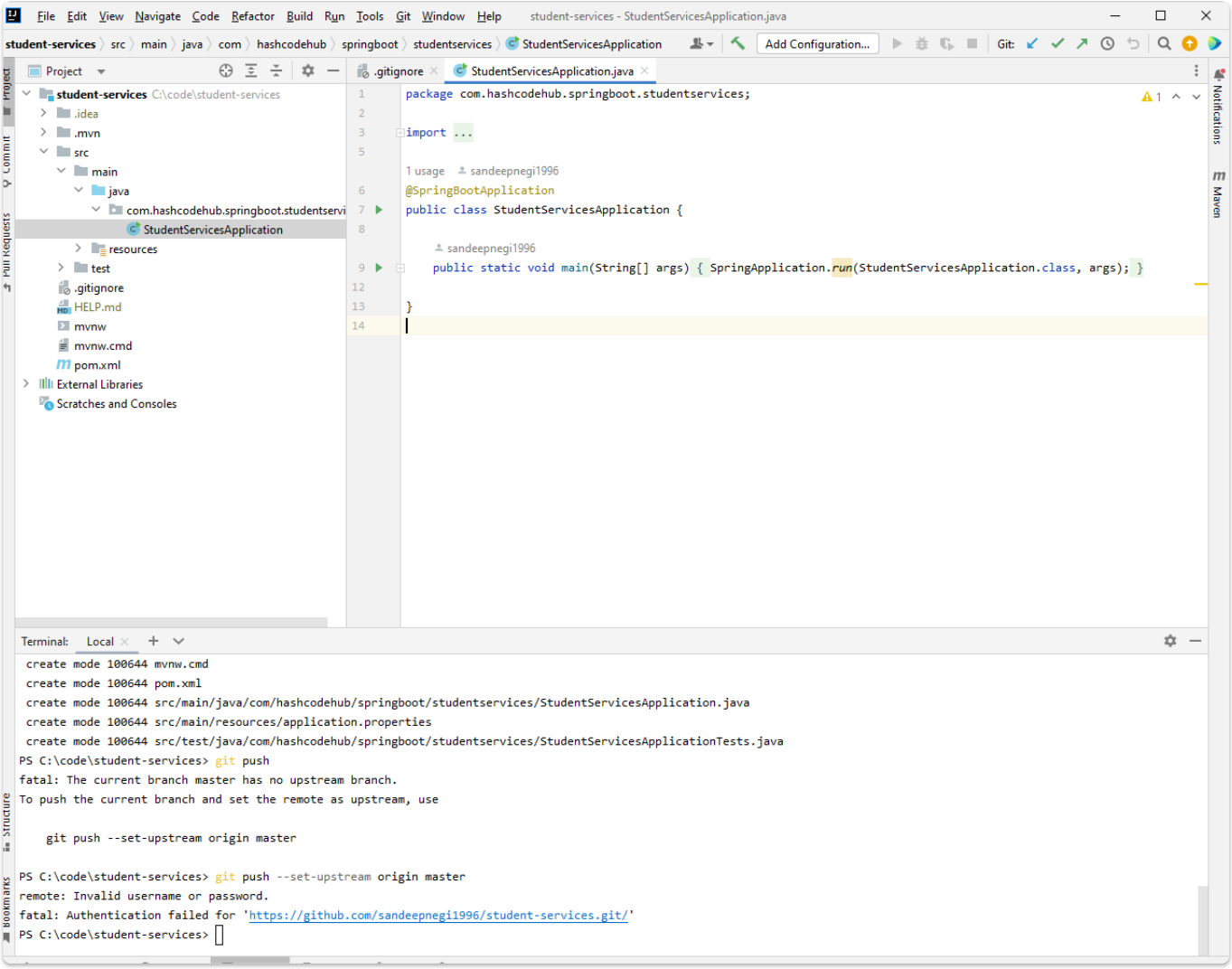
1. Create Project using Spring Initialzer

below mentioned are all the details of the spring initializer

1. download the zip

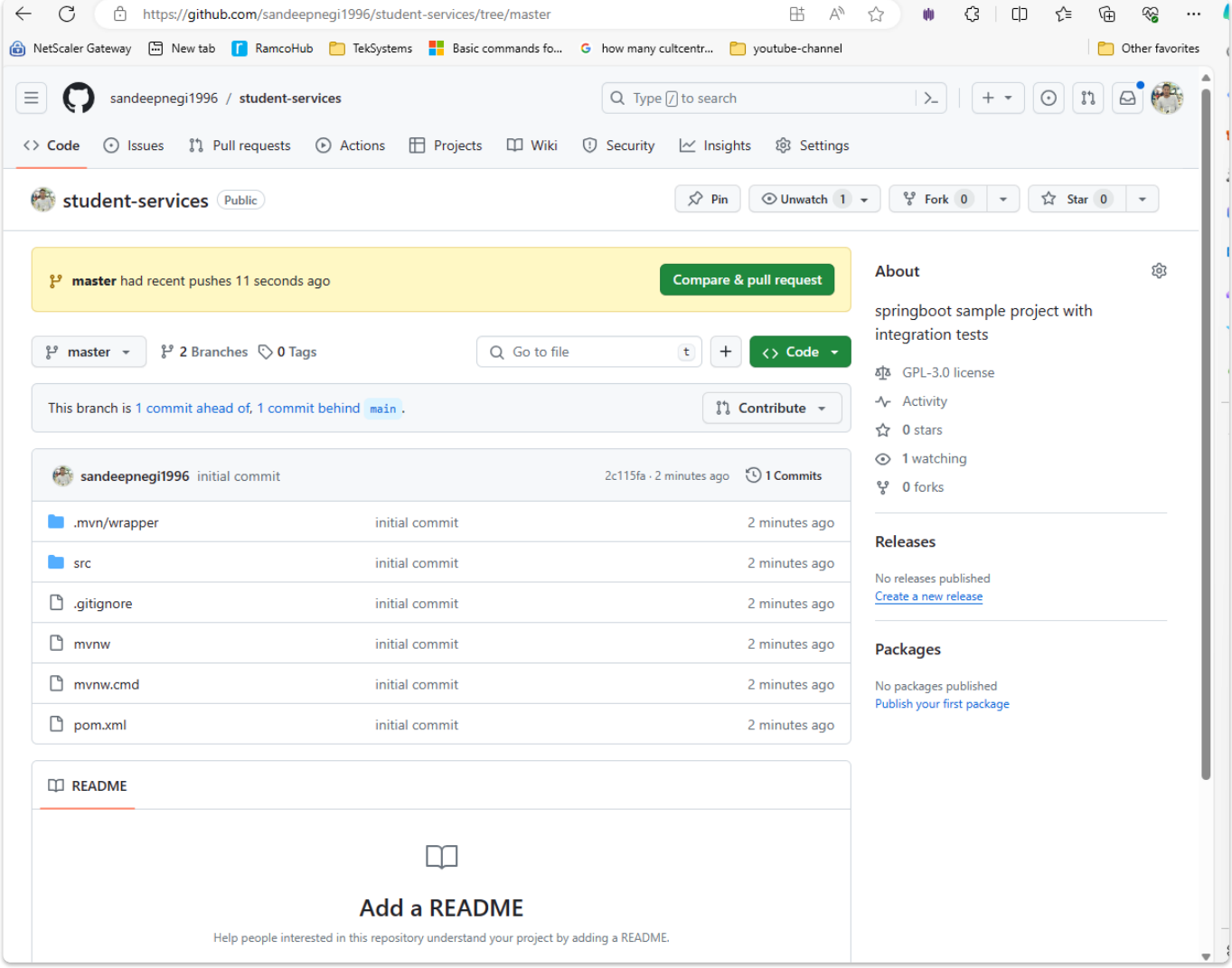
2. extract and open the same in IDE (intellj)
3.





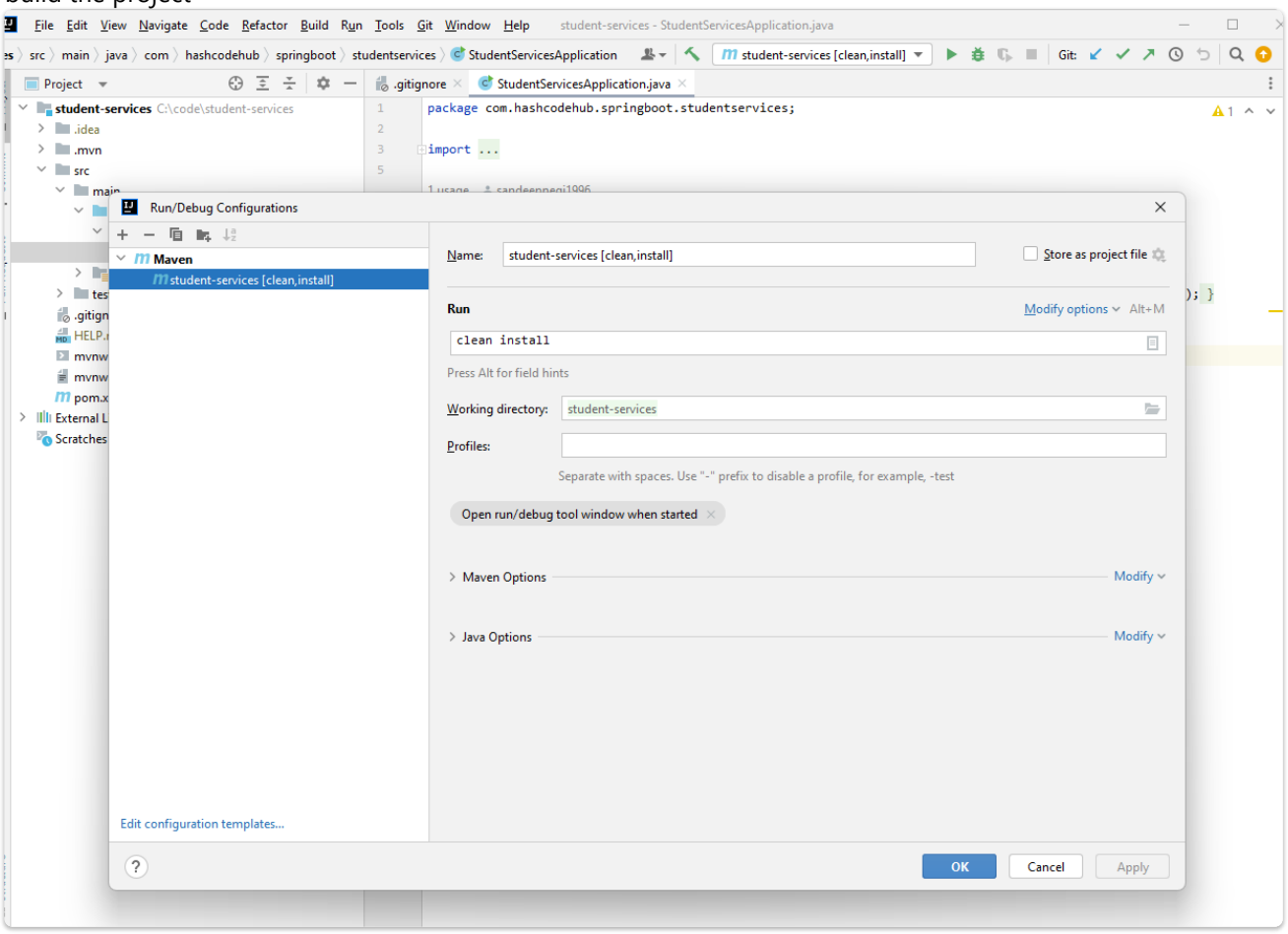
2. Push the code to github

1. git init
2. git add .
3. git commit -m "inital commit"
4. git remote add origin "from github"
5. git push

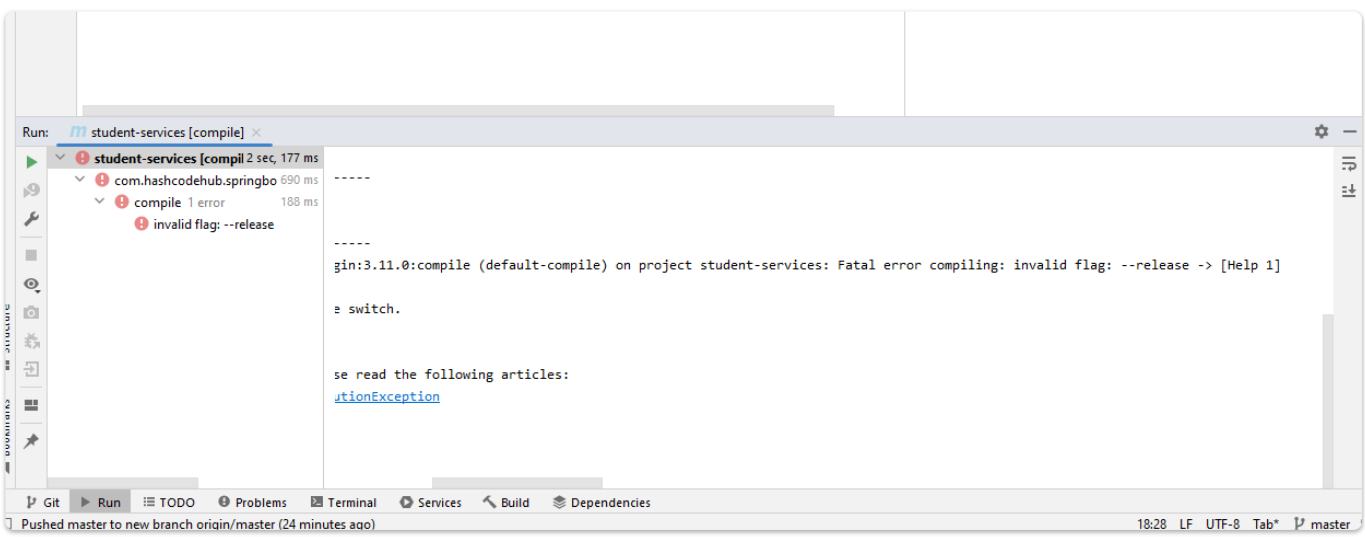


3. Run the project Locally

1. build the project



2. Run the project
   compile error

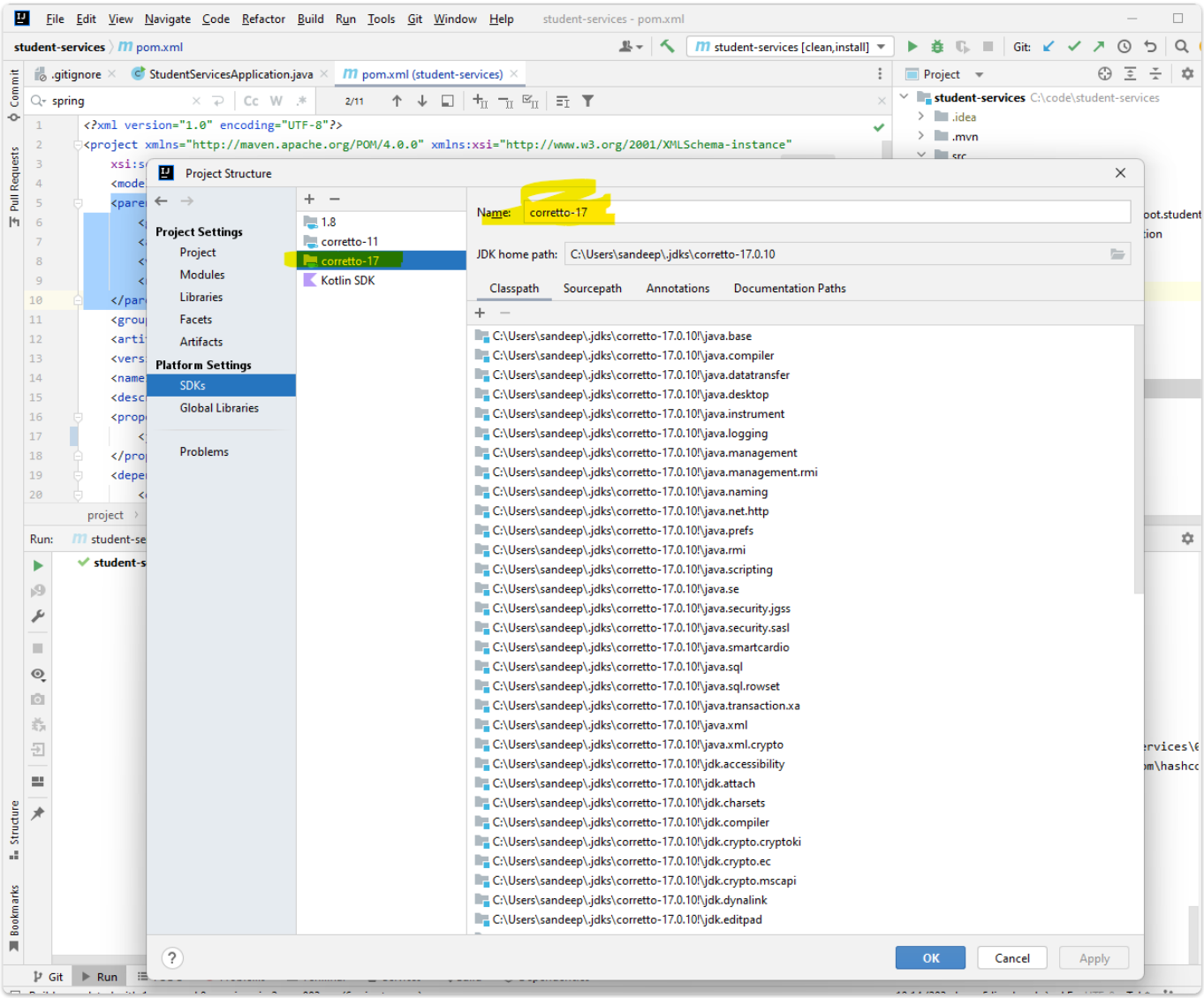


- resolution since we are using spring version

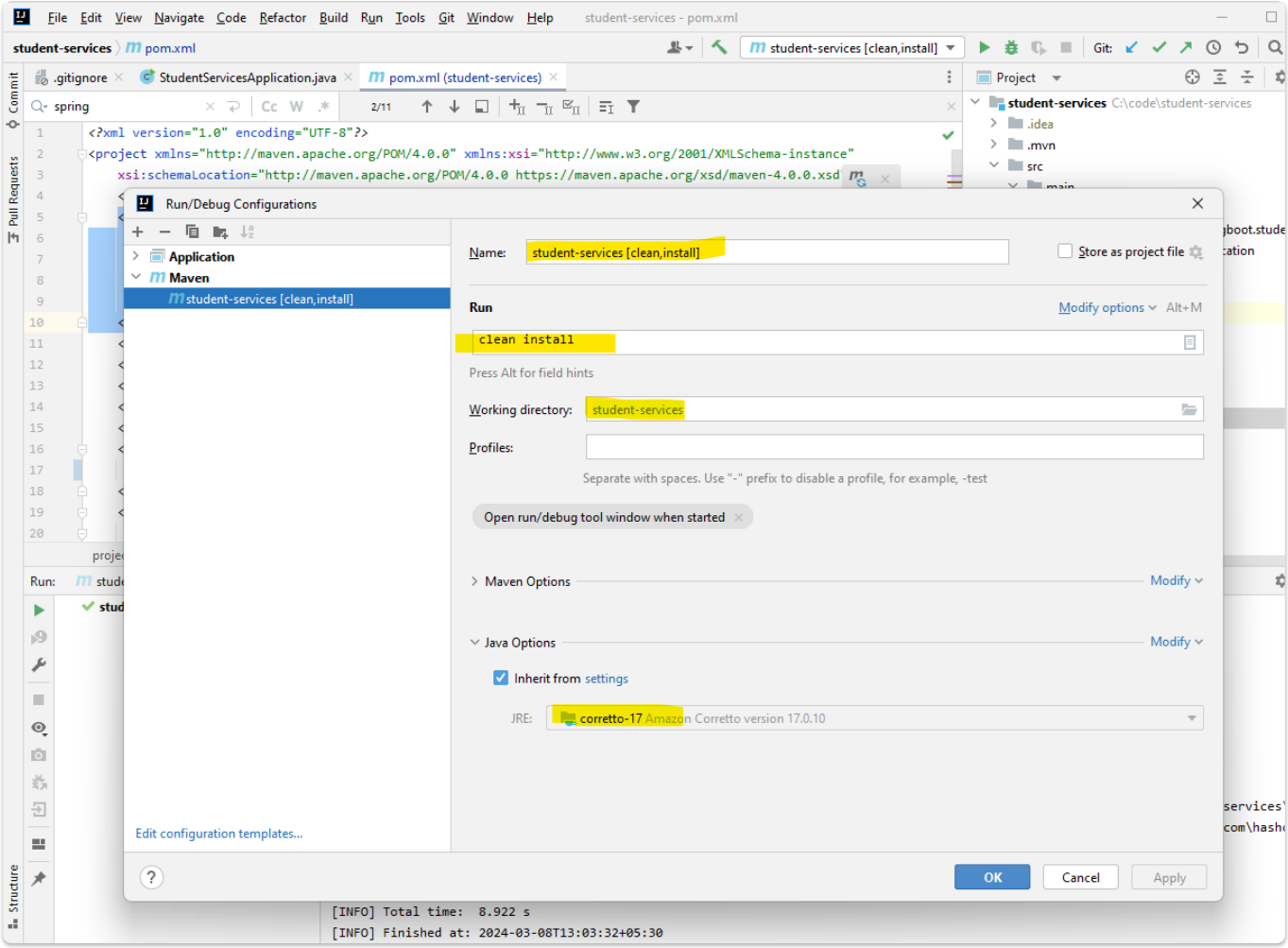
```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.3</version>
```

```
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
```

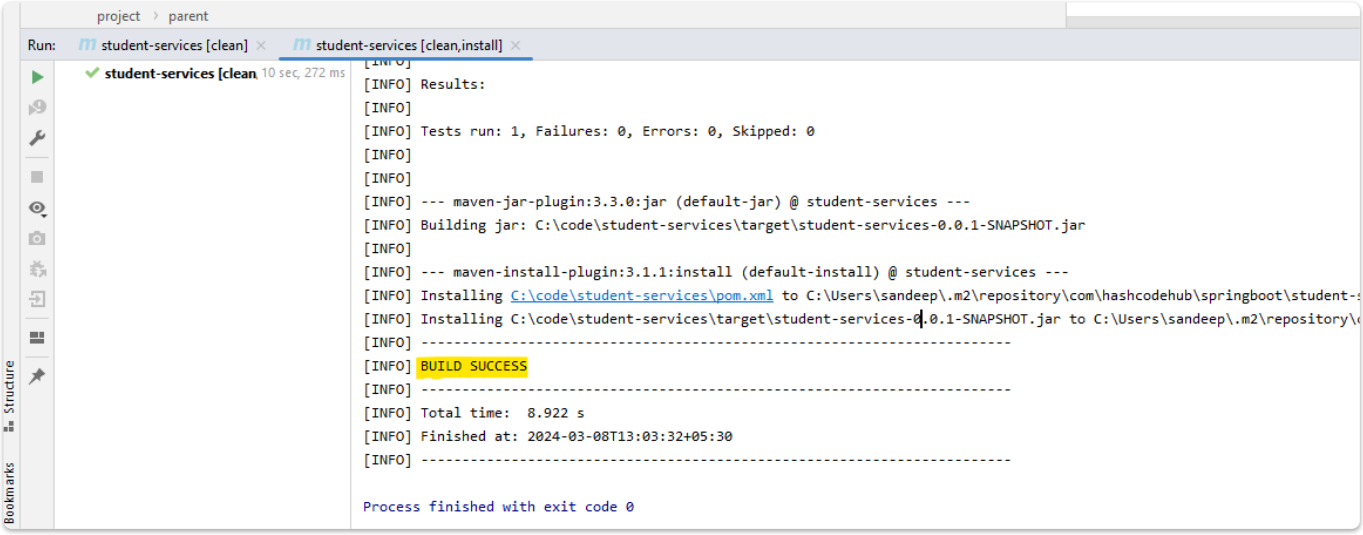
we need to use the java 17 version -> I have downloaded the SDK in intellj and used that while compiling



use the above mentioned one in the build settings



Build Got Success



Now Run the project -> just right click the file with main method and run the project

Note

1. Use this version for java 8 `<version>2.5.9</version>`

4. Adding Business Layer

1. we will use arraylist for storage
2. Course -> id, name, description, action to complete the course
3. Student -> id, name, description , list of courses registered

5. StudentController and StudentService

StudentController

```
package com.hashcodehub.springboot.studentservices.Controller;

import com.hashcodehub.springboot.studentservices.Model.Course;
import com.hashcodehub.springboot.studentservices.Service.StudentService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.servlet.support.ServletUriComponentsBuilder;

import java.net.URI;
import java.util.List;

@RestController
@RequestMapping("/students/{studentId}/courses")
public class StudentController {


    @Autowired
    private StudentService studentService;


    @GetMapping()
    public List<Course> retreiveCoursesForStudent(@PathVariable String studentId) {
        return studentService.retreiveCourses(studentId);
    }


    @GetMapping("/{courseId}")
    public Course retreiveDetailsForCourse(@PathVariable String studentId,@PathVariable String courseId
) {
        return studentService.retreiveCourse(studentId,courseId);
    }


    @PostMapping()
    public ResponseEntity<Void> registerStudentForCourse(@PathVariable String studentId, @RequestBody
Course newCourse) {

        Course course=studentService.addCourse(studentId,newCourse);

        if(course==null) {
            return ResponseEntity.noContent().build();
        }

        URI location=
ServletUriComponentsBuilder.fromCurrentRequest().path("/{studentId}").buildAndExpand(course.getId()).to
```

```
Uri();
        return ResponseEntity.created(location).build();
    }



}
```

- SudentService.java

```
package com.hashcodehub.springboot.studentservices.Service;

import com.hashcodehub.springboot.studentservices.Model.Course;
import com.hashcodehub.springboot.studentservices.Model.Student;
import org.springframework.stereotype.Service;

import java.math.BigInteger;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;

@Service
public class StudentService {

    //creating static data just to store and work with it

    private static final List<Student> students=new ArrayList<>();
    private final SecureRandom random = new SecureRandom();

    static {
        //Initialize Data

        Course courseOne=new Course("Course1","spring","10 Steps",
                Arrays.asList("Learn Maven", "Import Project", "First Example", "Second Example"));

        Course courseTwo=new Course("Course2","spring MVC","10 Steps",
                Arrays.asList("Learn Maven", "Import Project", "First Example", "Second Example"));


        Course courseThree=new Course("Course3","spring Boot","10 Steps",
                Arrays.asList("Learn Maven", "Import Project", "First Example", "Second Example"));


        Course courseFour=new Course("Course4","Maven","10 Steps",
                Arrays.asList("Learn Maven", "Import Project", "First Example", "Second Example"));


        List<Course> courses= new ArrayList<>();
        courses.add(courseOne);
        courses.add(courseTwo);
        courses.add(courseThree);
        courses.add(courseFour);
```

```
        Student ranga = new Student("Student1", "Ranga Karanam", "Hiker, Programmer and Architect",
               courses);


        Student satish = new Student("Student2", "Satish", "Hiker, Programmer",
                courses);



        students.add(ranga);
        students.add(satish);




    }


//    public List<Course> retrieveCourses(String studentId) {
//        return
//    }

    public List<Student> retreiveAllStudents() {
        return students;
    }

    //retreive  a particular student from the list

    public Student retreiveStudent(String studentId) {
        return students.stream()
                .filter(student -> student.getId().equals(studentId))
                .findAny()
                .orElse(null);
    }

    //retreive all the courses for a student

    public List<Course> retreiveCourses(String studentId) {
        Student student= retreiveStudent(studentId);
      return  student==null ?null : student.getCourses();

    }

    //retrieve a particular couse of a particlar student

    public Course retreiveCourse(String studentId,String courseId) {
        Student student = retreiveStudent(studentId);

        List<Course> courses= student.getCourses();

        return courses.stream()
                .filter(course -> course.getId().equals(courseId))
                .findAny()
                .orElse(null);


    }
```

```
    public Course addCourse(String studentId, Course course) {
        Student student = retreiveStudent(studentId);

        if (student == null) {
            return null;
        }

        String randomId = new BigInteger(130, random).toString(32);
        course.setId(randomId);

        student.getCourses()
                .add(course);

        return course;
    }
}
```

Model Classes

Student

```
package com.hashcodehub.springboot.studentservices.Model;

import org.springframework.stereotype.Component;

import java.util.List;
import java.util.Objects;

public class Student {

    private String id;
    private String name;
    private String description;
    private List<Course> courses;


    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Student student = (Student) o;
        return Objects.equals(id, student.id) && Objects.equals(name, student.name) &&
Objects.equals(description, student.description) && Objects.equals(courses, student.courses);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, description, courses);
    }

    public String getId(){
        return id;
    }
```

```
    public String getName(){
        return name;
    }

    public String getDescription(){
        return description;
    }

    public List<Course> getCourses(){
        return courses;
    }

    public Student(String id,String name,String description,List<Course> courses) {
        this.id=id;
        this.name=name;
        this.description=description;
        this.courses=courses;
    }

    public String toString() {
        return "studentId:  "+this.getId()+"  StudentName:  "+this.getName() +" description:
"+this.getDescription()
                +"Courses "+this.getCourses();

    }


}
```

6. Get and post endpoints postman

Postman

File Edit View Help

Home Workspaces Reports Explore Search Postman Sign In Create Account

Scratch Pad New Import

GET getSingleCourse | GET getAllCourses | StudentService | POST postCourse | No Environment

Collections
APIs
Environments
Mock Servers
Monitors
History

- animeAPI-online
- StudentService
  - GET getAllCourses
  - GET getSingleCourse
  - POST postCourse

StudentService / getAllCourses Save

GET http://localhost:8080/students/Student2/courses Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 14 ms 819 B Save Response

Pretty Raw Preview Visualize JSON


```
            "Import Project",
            "First Example",
            "Second Example"
        ]
    },
    {
        "name": "Maven",
        "id": "Course4",
        "description": "10 Steps",
        "steps": [
            "Learn Maven",
            "Import Project",
            "First Example",
            "Second Example"
        ]
    },
```

Postman

File Edit View Help

Home Workspaces Reports Explore Search Postman Sign In Create Account

Scratch Pad New Import

GET getSingleCourse GET getAllCourses StudentService POST postCourse No Environment

Collections
APIs
Environments
Mock Servers
Monitors
History

- animeAPI-online
- StudentService
  - GET getAllCourses
  - GET getSingleCourse
  - POST postCourse

StudentService / getSingleCourse Save

GET http://localhost:8080/students/Student2/courses/Course2 Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

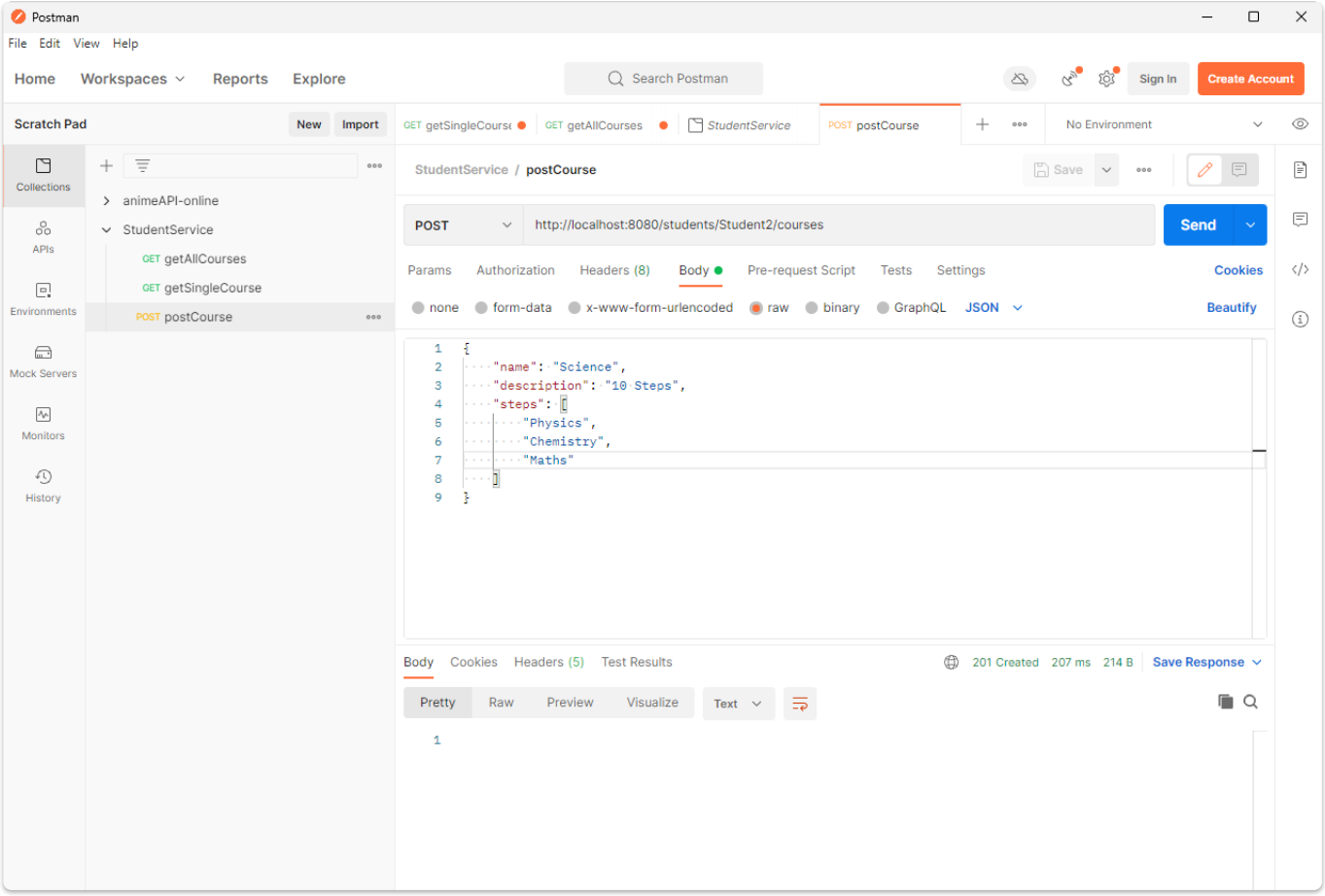
Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 14 ms 299 B Save Response

Pretty Raw Preview Visualize JSON


```
{
    "name": "spring MVC",
    "id": "Course2",
    "description": "10 Steps",
    "steps": [
        "Learn Maven",
        "Import Project",
        "First Example",
        "Second Example"
    ]
}
```



5. Writing Integrations Tests

> *When we are writing an integration test for a rest service, we would want to launch the entire spring context.*

- `@SpringBootTest(classes = StudentServicesApplication.class, webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)` : Launch the entire Spring Boot Application on a Random Port
- `@LocalServerPort private int port;` : Autowire the random port into the variable so that we can use it create the url.
- `createURLWithPort(String uri)` : Utility method to create the url given an uri. It appends the port.
- `HttpEntity<String> entity = new HttpEntity<String>(null, headers);` : We use entity so that we have the flexibility of adding in request headers in future.
- `restTemplate.exchange(createURLWithPort("/students/Student1/courses/Course1"),HttpMethod.GET, entity, String.class)` : Fire a GET request to the specify uri and get the response as a String.
- `JSONAssert.assertEquals(expected, response.getBody(), false)` : Assert that the response contains expected fields.

Example of Testcase

```
package com.hashcodehub.springboot.studentservices.Controller;


import com.hashcodehub.springboot.studentservices.StudentServicesApplication;
import org.json.JSONException;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
```

```java
import org.skyscreamer.jsonassert.JSONAssert;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.boot.test.web.client.TestRestTemplate;
import org.springframework.boot.web.server.LocalServerPort;
import org.springframework.http.HttpEntity;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpMethod;
import org.springframework.http.ResponseEntity;
import org.springframework.test.context.junit.jupiter.SpringExtension;
//
//When we are writing an integration test for a rest service, we would want to launch the entire spring
context.
//
//@SpringBootTest(classes = StudentServicesApplication.class, webEnvironment =
SpringBootTest.WebEnvironment.RANDOM_PORT) : Launch the entire Spring Boot Application on a Random Port
//@LocalServerPort private int port;: Autowire the random port into the variable so that we can use it
create the url.
//        createURLWithPort(String uri) : Utility method to create the url given an uri. It appends the
port.
//        HttpEntity<String> entity = new HttpEntity<String>(null, headers);: We use entity so that we
have the flexibility of adding in request headers in future.
//        restTemplate.exchange(createURLWithPort("/students/Student1/courses/Course1"),HttpMethod.GET,
entity, String.class): Fire a GET request to the specify uri and get the response as a String.
//        JSONAssert.assertEquals(expected, response.getBody(), false) : Assert that the response
contains expected fields.

@ExtendWith(SpringExtension.class)
@SpringBootTest(classes = StudentServicesApplication.class,webEnvironment
=SpringBootTest.WebEnvironment.RANDOM_PORT)
public class StudentControllerIT {

    @LocalServerPort
    private int port;


    TestRestTemplate restTemplate=new TestRestTemplate();

    HttpHeaders headers = new HttpHeaders();

    @Test
    public void testRetrieveStudent1Course1() throws JSONException {

        HttpEntity<String> entity=new HttpEntity<>(null,headers);

       ResponseEntity<String> response =  restTemplate.exchange(
                createURLWithPort("/students/Student1/courses/Course1"),
                HttpMethod.GET,entity,String.class);

       String expected="{\"name\":\"spring\",\"id\":\"Course1\",\"description\":\"10 Steps\",\"steps\":
[\"Learn Maven\",\"Import Project\",\"First Example\",\"Second Example\"]}";
        JSONAssert.assertEquals(expected,response.getBody(),false);

    }

    @Test
    public void testRetrieveStudent2Course2() throws JSONException {
        HttpEntity<String> entity = new HttpEntity<>(null,headers);
```

```
        ResponseEntity<String> response= restTemplate.exchange(
                createURLWithPort("/students/Student2/courses/Course2"),
                HttpMethod.GET,entity,String.class
        );

        String expected="{\"name\":\"spring MVC\",\"id\":\"Course2\",\"description\":\"10
Steps\",\"steps\":[\"Learn Maven\",\"Import Project\",\"First Example\",\"Second Example\"]}";

        JSONAssert.assertEquals(expected,response.getBody(),false);

    }



    private String createURLWithPort(String uri) {
        return "http://localhost:"+port+uri;
    }




}
```