

springboot integration tests example

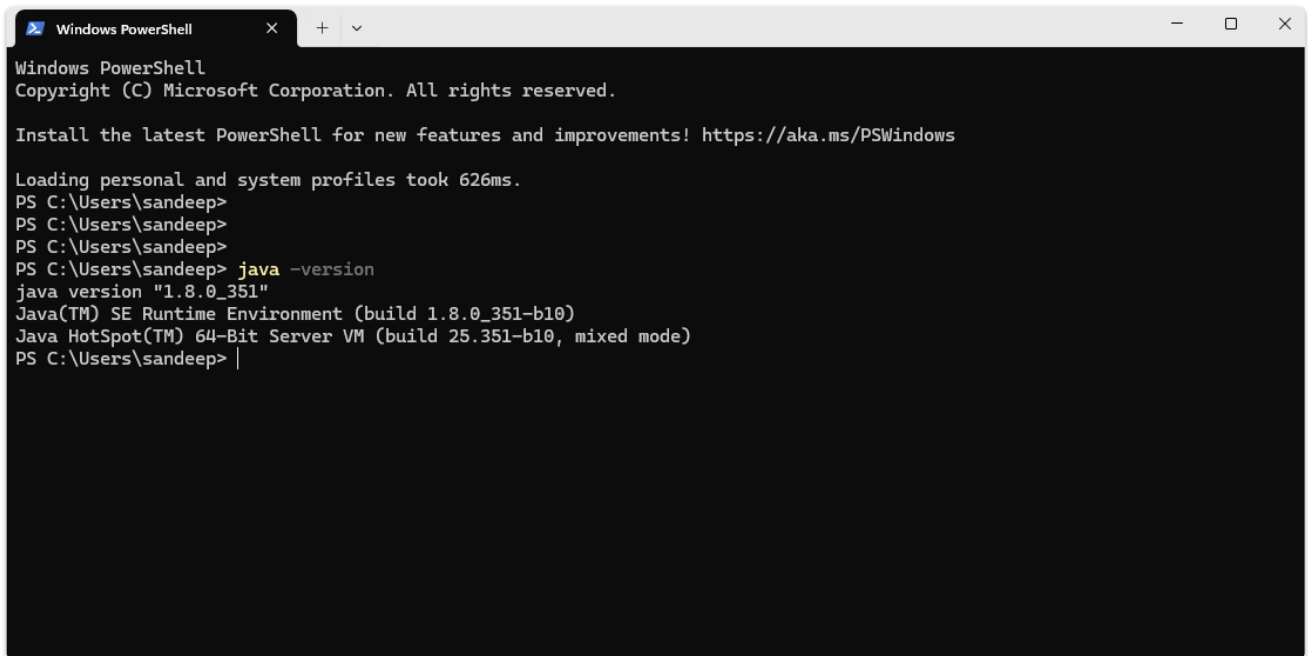
Blogpost : [Writing Integration Tests for Rest Services with Spring Boot](#) | [Spring Boot Tutorial](#)

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. IntelliJ
4. git
5. Java 8



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

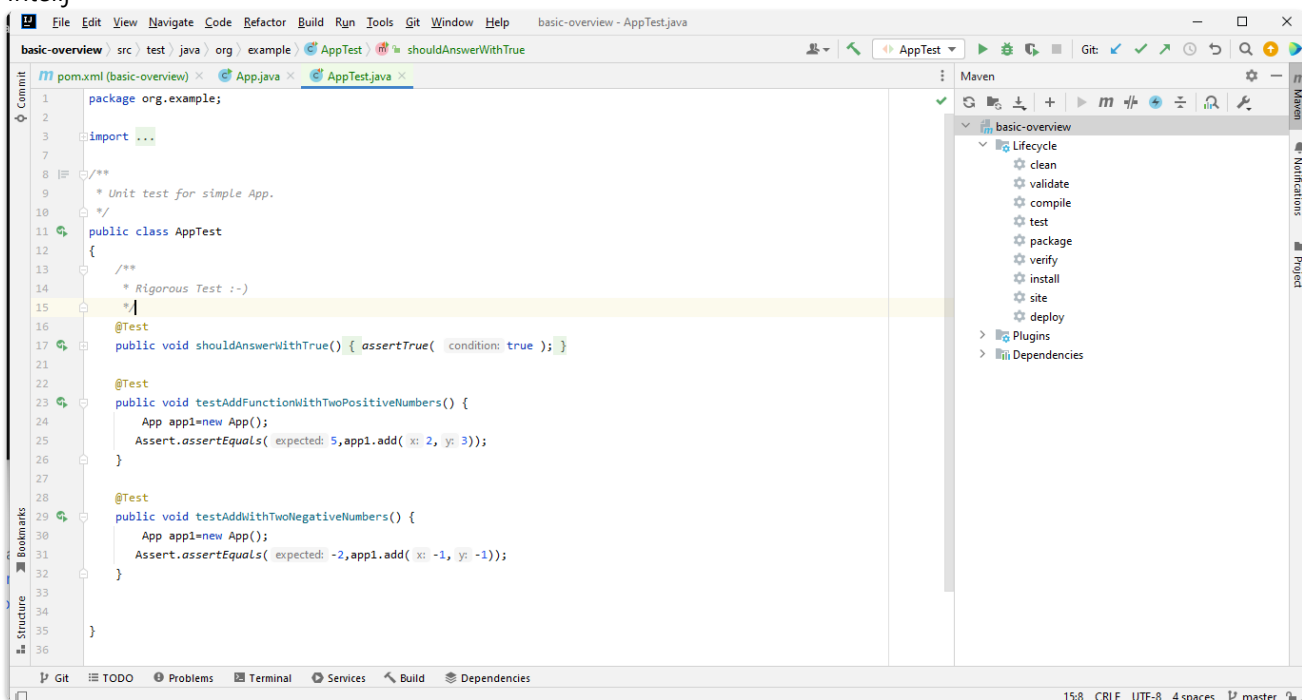
Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

Loading personal and system profiles took 626ms.
PS C:\Users\sandeep>
PS C:\Users\sandeep>
PS C:\Users\sandeep>
PS C:\Users\sandeep> java -version
java version "1.8.0_351"
Java(TM) SE Runtime Environment (build 1.8.0_351-b10)
Java HotSpot(TM) 64-Bit Server VM (build 25.351-b10, mixed mode)
PS C:\Users\sandeep> |
```

```
Windows PowerShell
PS C:\Users\sandeep> mvn -version
Apache Maven 3.8.5 (3599d3414f046de2324203b78ddcf9b5e4388aa0)
Maven home: C:\sandeep\apache-maven-3.8.5
Java version: 1.8.0_331, vendor: Oracle Corporation, runtime: C:\Program Files\Java\jdk1.8.0_331\jre
Default locale: en_US, platform encoding: Cp1252
OS name: "windows 11", version: "10.0", arch: "amd64", family: "windows"
PS C:\Users\sandeep> |
```

6.

7. IntelliJ



Overview

1. Create a project using spring initializer
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 Initializer

below mentioned are all the details of the spring initializer

1. download the zip

2. extract and open the same in IDE (intelliJ)

3.

The screenshot shows the Spring Initializr web application in a browser. The URL is <https://start.spring.io>. The interface is divided into several sections for configuring a new Spring project.

Project

- ☐ Gradle - Groovy
- ☐ Gradle - Kotlin
- ☒ Maven

Language

- ☒ Java
- ☐ Kotlin
- ☐ Groovy

Spring Boot

- ☐ 3.3.0 (SNAPSHOT)
- ☐ 3.3.0 (M2)
- ☐ 3.2.4 (SNAPSHOT)
- ☒ 3.2.3
- ☐ 3.1.10 (SNAPSHOT)
- ☐ 3.1.9

Project Metadata

Group:

Artifact:

Name:

Description:

Package name:

Packaging: ☒ Jar ☐ War

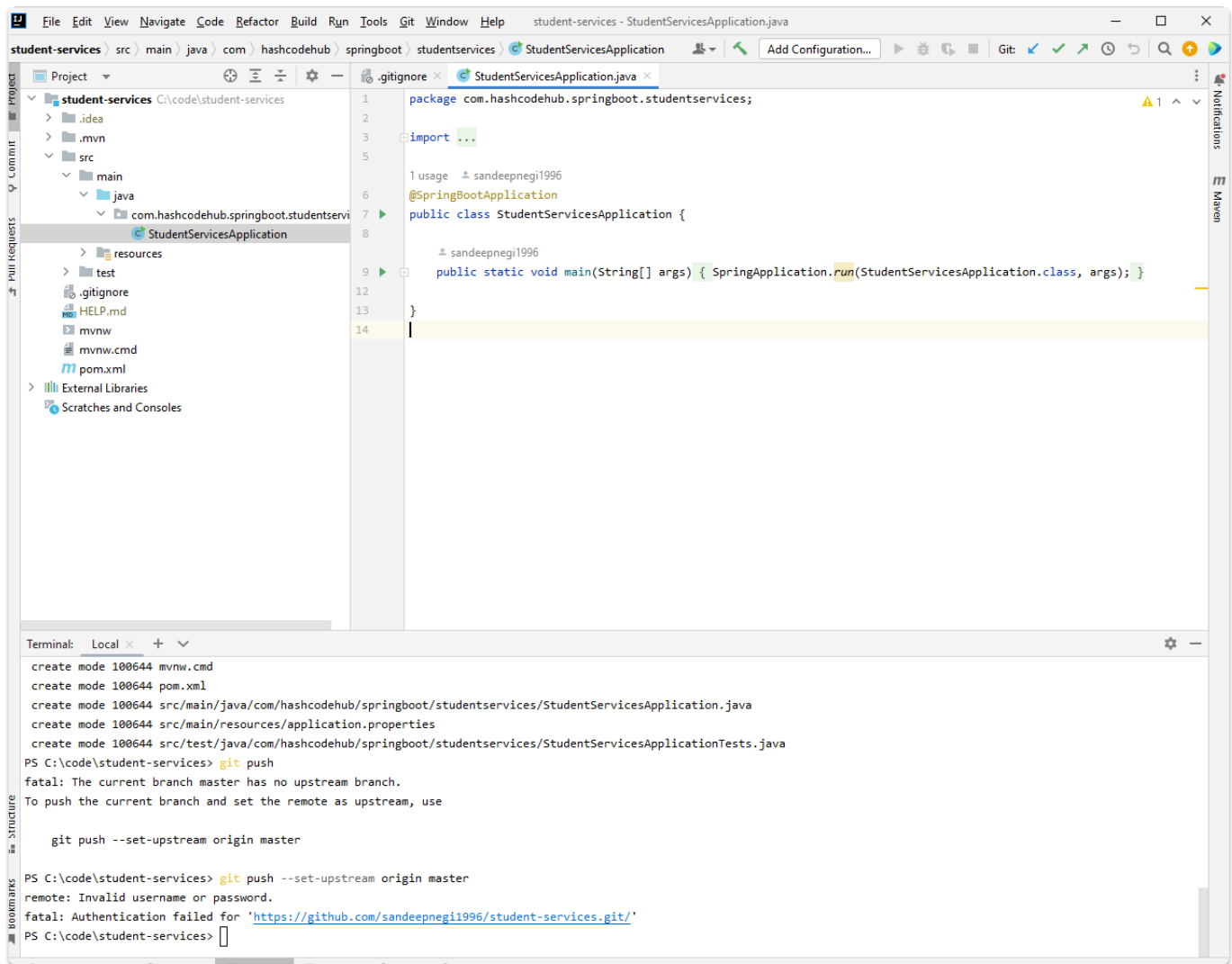
Java: ☐ 21 ☒ 17

Dependencies ADD DEPENDENCIES... CTRL + B

- Spring Web** WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container. +
- Spring Boot Actuator** OPS
Supports built in (or custom) endpoints that let you monitor and manage your application - such as application health, metrics, sessions, etc. +
- Spring Boot DevTools** DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience. +

Buttons:

- GENERATE** CTRL + G
- EXPLORE** CTRL + SPACE
- SHARE...**



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

← ↻ https://github.com/sandeepnegi1996/student-services/tree/master

NetScaler Gateway New tab RamcoHub TekSystems Basic commands fo... how many cultcentr... youtube-channel Other favorites

☰ sandeepnegi1996 / student-services 🔍 Type to search + 🔁 📧 🧑

<> Code ⌚ Issues 🔗 Pull requests ⚙️ Actions 📁 Projects 📖 Wiki 🛡️ Security 📈 Insights ⚙️ Settings

👤 student-services Public 📌 Pin 📺 Unwatch 1 🍴 Fork 0 ⭐ Star 0

🔑 master had recent pushes 11 seconds ago [Compare & pull request](#)

🔑 master 2 Branches 0 Tags 🔍 Go to file + <> Code

This branch is 1 commit ahead of, 1 commit behind main. 📌 Contribute

👤 sandeepnegi1996 initial commit 2c115fa · 2 minutes ago ⌚ 1 Commits

📁 .mvn/wrapper	initial commit	2 minutes ago
📁 src	initial commit	2 minutes ago
📄 .gitignore	initial commit	2 minutes ago
📄 mvnw	initial commit	2 minutes ago
📄 mvnw.cmd	initial commit	2 minutes ago
📄 pom.xml	initial commit	2 minutes ago

📖 README

📖

Add a README

Help people interested in this repository understand your project by adding a README.

About ⚙️

springboot sample project with integration tests

📄 GPL-3.0 license

📈 Activity

⭐ 0 stars

👁️ 1 watching

🍴 0 forks

Releases

No releases published

[Create a new release](#)

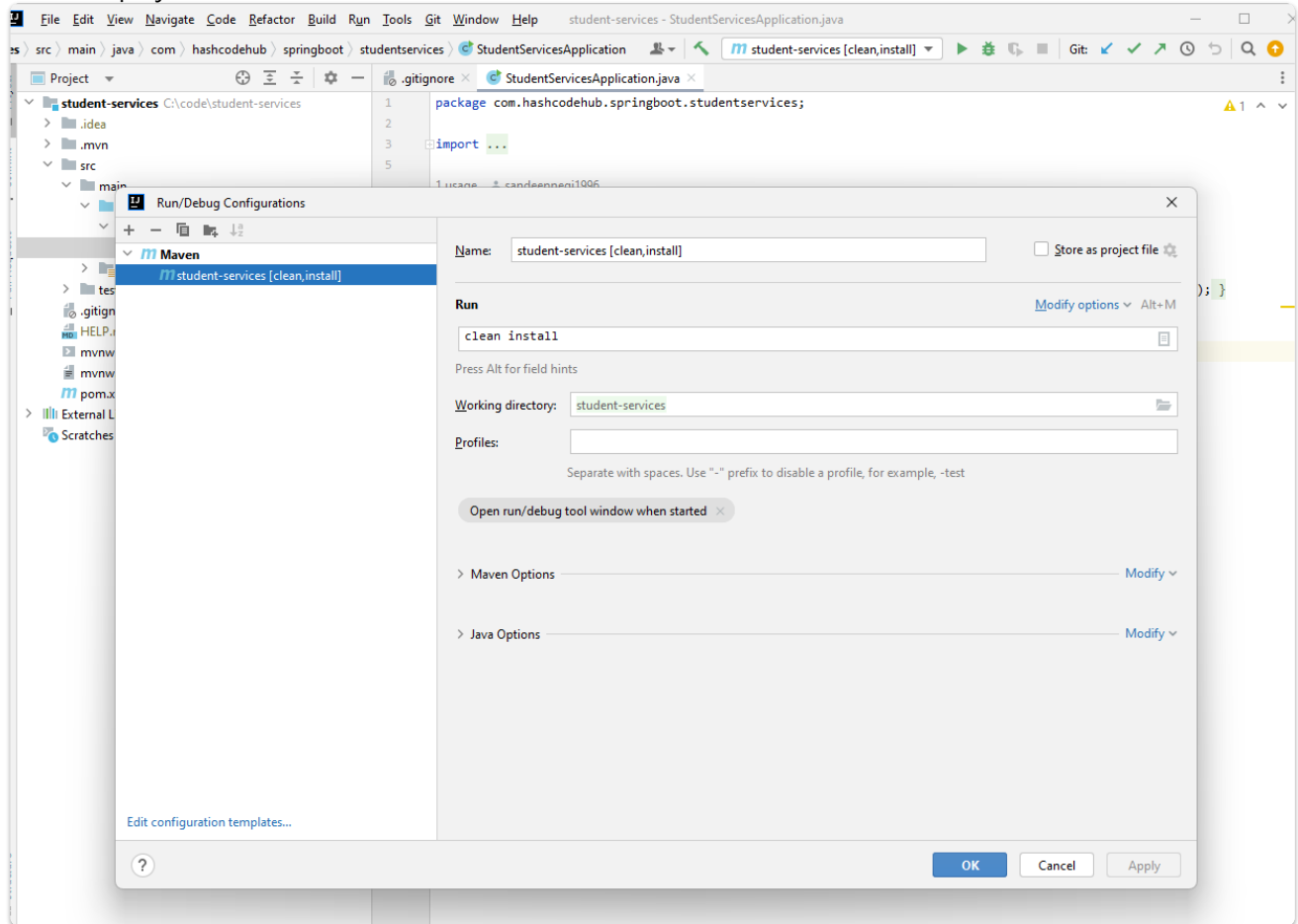
Packages

No packages published

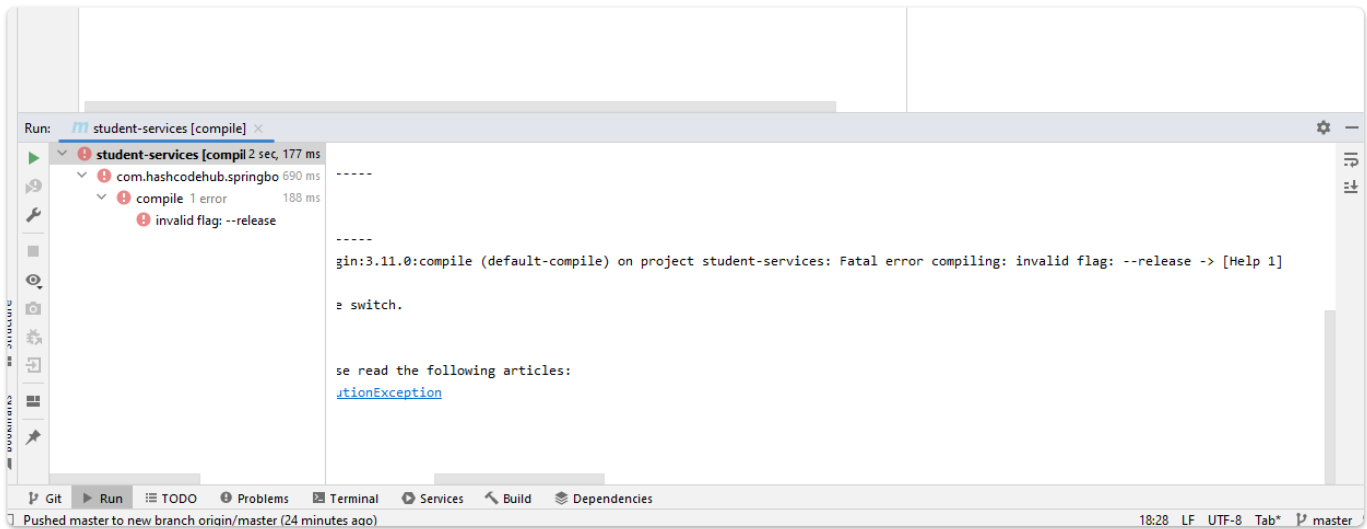
[Publish your first package](#)

3. Run the project Locally

1. build the project



2. Run the project compile error

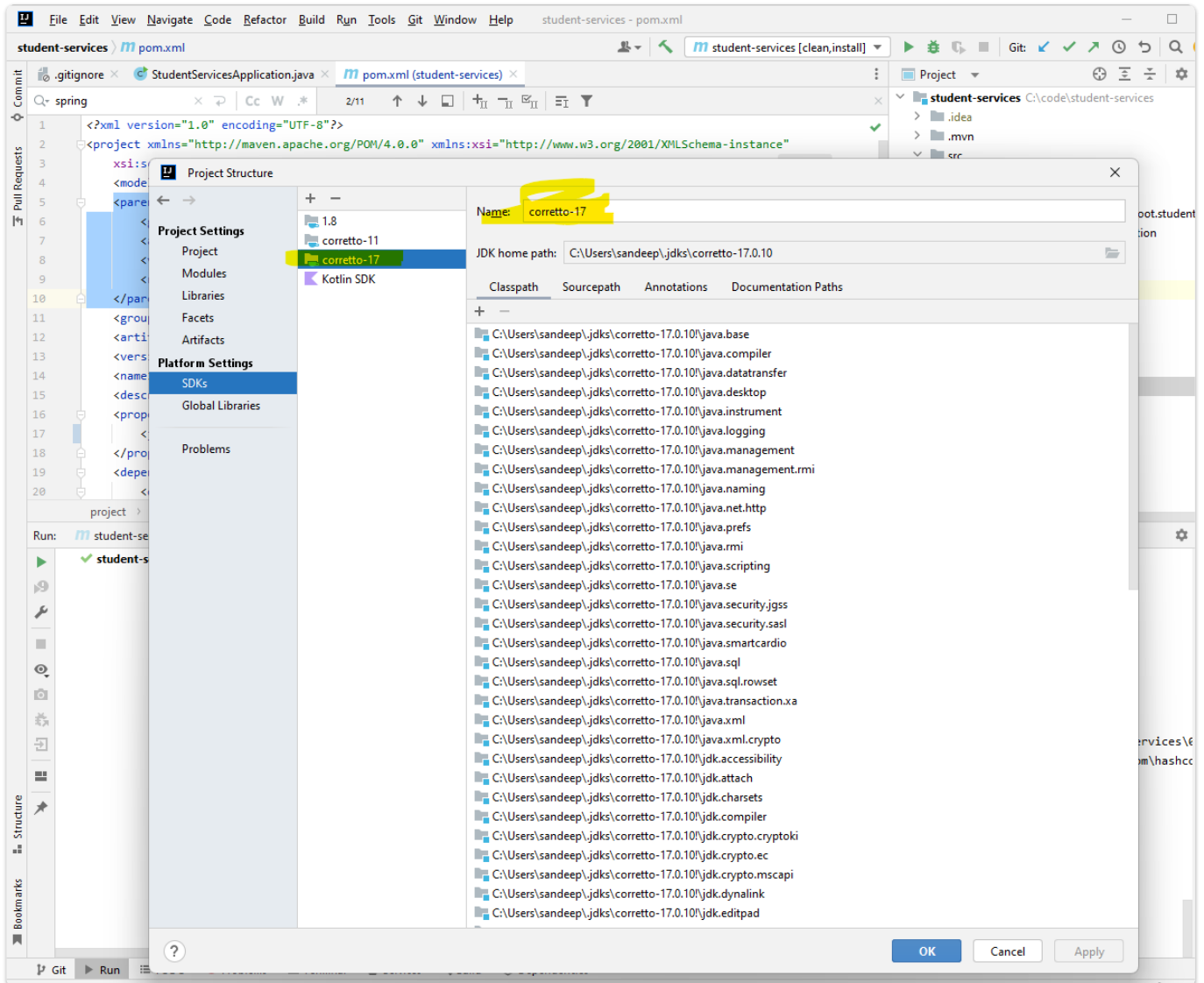


- resolution since we are using spring version

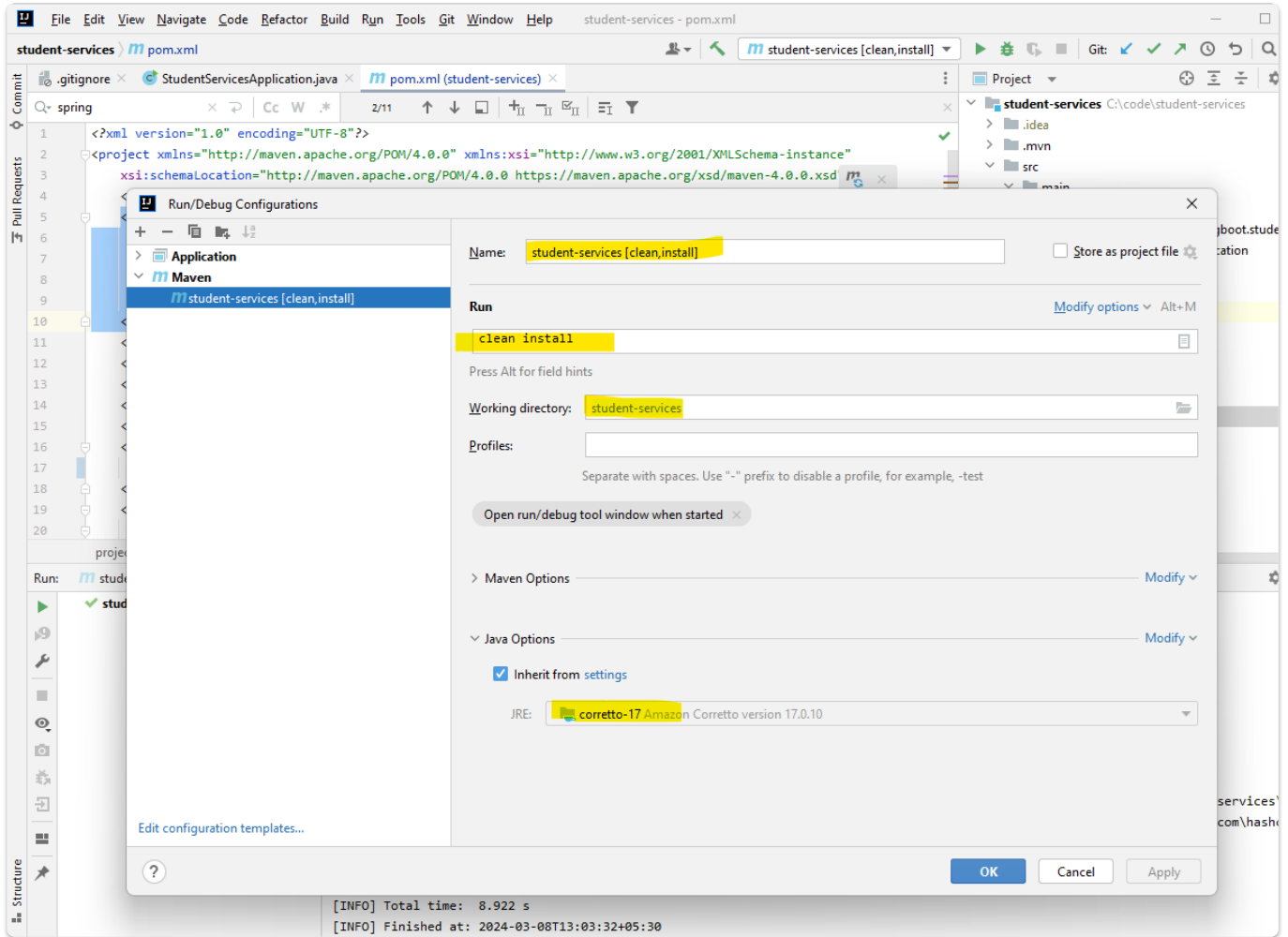
```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.2.3</version>
```

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

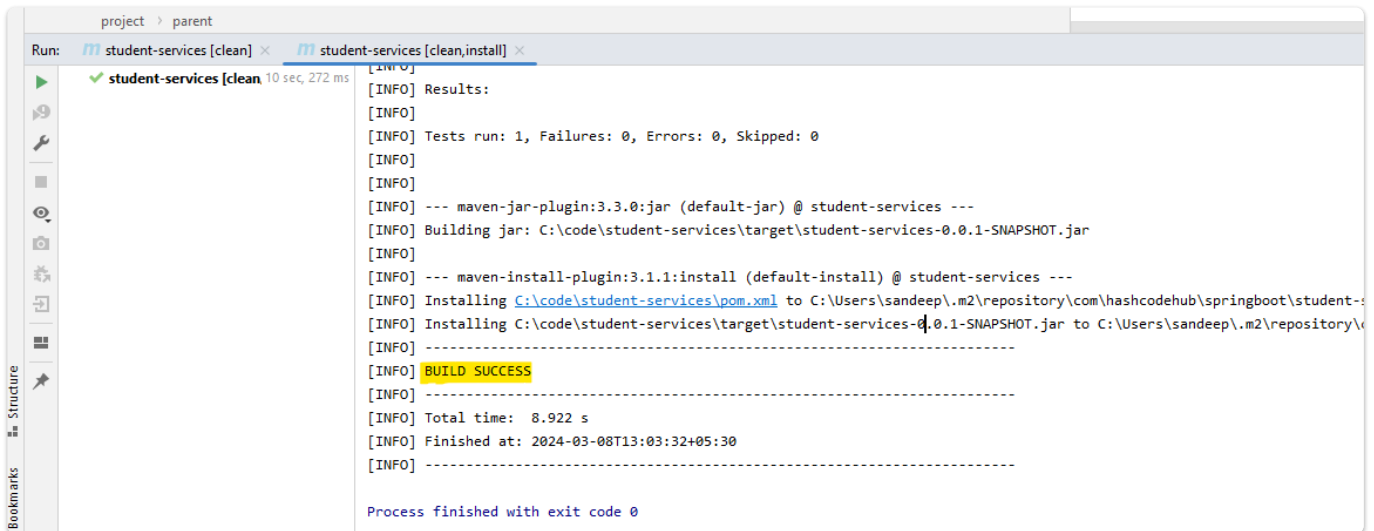
we need to use the java 17 version -> I have downloaded the SDK in intellj and used that while compiling



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> retrieveCoursesForStudent(@PathVariable String studentId) {
        return studentService.retrieveCourses(studentId);
    }

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

}

```

- StudentService.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> retrieveAllStudents() {
        return students;
    }

//retrieve a particular student from the list

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

//retrieve all the courses for a student

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

//retrieve a particular course of a particular student

    public Course retrieveCourse(String studentId,String courseId) {
        Student student = retrieveStudent(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 = retrieveStudent(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

StudentService / getAllCourses

Save

Send

GET

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

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

```
30      "Import Project",
31      "First Example",
32      "Second Example"
33    ],
34  },
35  {
36    "name": "Maven",
37    "id": "Course4",
38    "description": "10 Steps",
39    "steps": [
40      "Learn Maven",
41      "Import Project",
42      "First Example",
43      "Second Example"
44    ]
45  },
46  ]
```

Postman

File Edit View Help

HomeWorkspacesReportsExplore

Search Postman

Sign InCreate Account

Scratch Pad

NewImport

GET getSingleCourseGET getAllCoursesStudentServicePOST postCourse

SaveSend

GEThttp://localhost:8080/students/Student2/courses/Course2

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettingsCookies

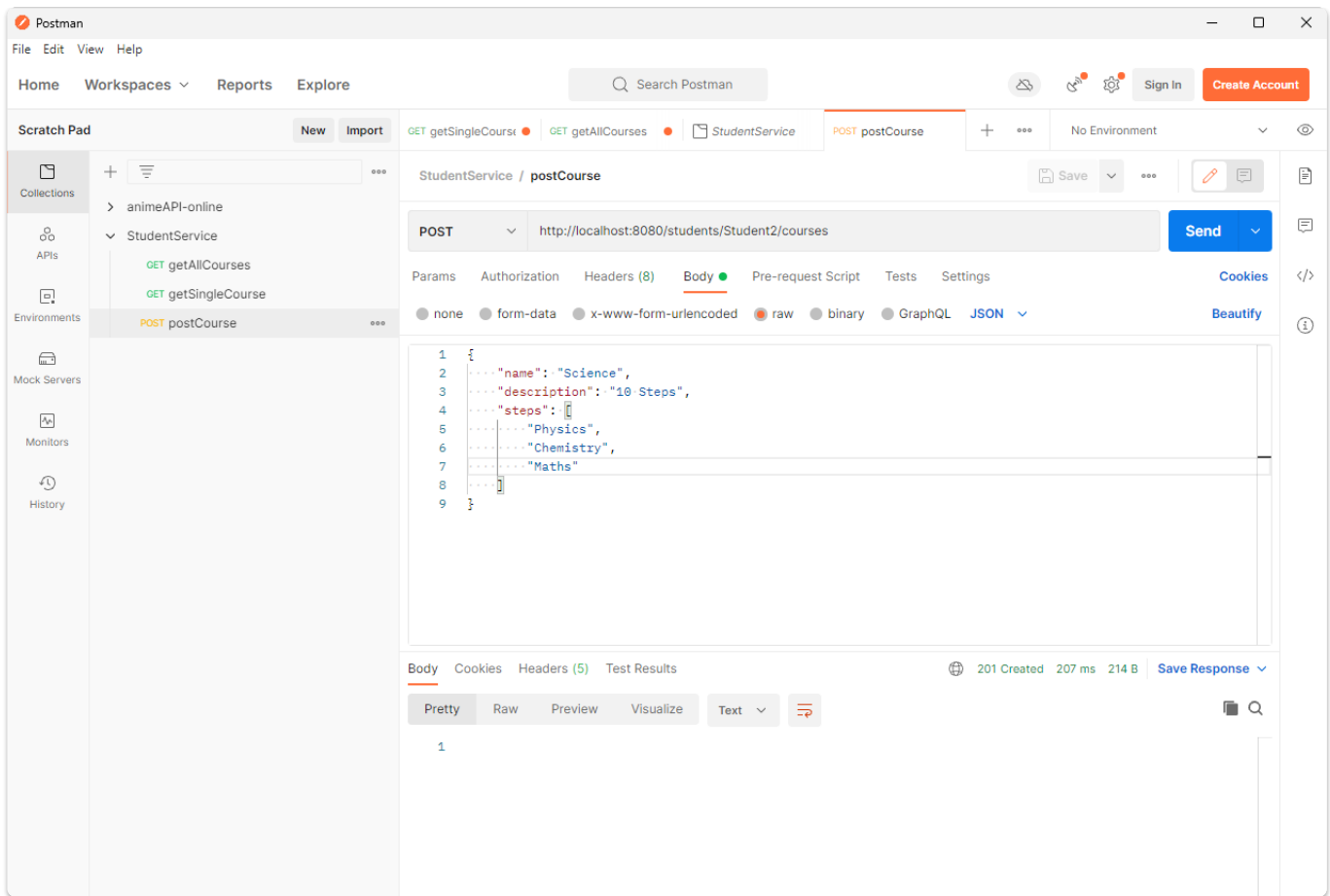
Query Params

KEY	VALUE	DESCRIPTION	Bulk Edit
Key	Value	Description	

BodyCookiesHeaders (5)Test Results200 OK 14 ms 299 BSave Response

PrettyRawPreviewVisualizeJSON

```
1  {
2    "name": "spring MVC",
3    "id": "Course2",
4    "description": "10 Steps",
5    "steps": [
6      "Learn Maven",
7      "Import Project",
8      "First Example",
9      "Second Example"
10   ]
11 }
```



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;
```



```

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;
    }

}
```