Assignment

* 2

**Assignment 3 -- Introduction to Springboot and Spring Framework**

The design is complete,

and

it's time to implement. Using Java, Spring Boot, Spring Web, Spring

Data JPA, and H2(in- memory database)

,

implement a microservice(studentms) to manage students with all CRUD operations.

1.An API to fetch all students

2.An API to fetch a single student

3. An API to create a new student

4. An API to delete a student

5. An API to update a student

Hint: Service registry/discovery, load balancing, circuit breaker

**Solution:**

To implement the **studentms** microservice using Java, Spring Boot, Spring Web, Spring Data Mongo, and Momgo DB we will create a basic Spring Boot application with the following steps:

1. Set up Spring Boot dependencies.
2. Create a Student entity to represent the student data.
3. Set up a repository for CRUD operations.
4. Implement a service layer for business logic.
5. Create RESTful APIs to handle CRUD operations.
6. Configure a basic H2 database. – **We have choosen Mongo DB**
7. Implement basic service registry/discovery, load balancing, and circuit breaker concepts.

**1. Project Setup**

We can use **Spring Initializr** to generate a Spring Boot project with the required dependencies:

* **Spring Web**
* **Spring Data Mongo**
* **Mongo DB**
* **Eureka Client** (for service registry and discovery)
* **Spring Cloud Circuit Breaker** (for circuit breaker)

You can generate the project from [Spring Initializr](https://start.spring.io/) or directly set it up in your IDE.

**2. Application Dependencies**

In pom.xml, include these dependencies:

<dependencies>

<!-- Spring Boot Starter Web for REST APIs -->

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-web</artifactId>  
</dependency>

<!-- Spring Boot Starter Data JPA for interacting with the database -->

<dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-mongodb</artifactId>  
</dependency>

<!-- Eureka Client for Service Discovery -->

<dependency>  
 <groupId>org.springframework.cloud</groupId>  
 <artifactId>spring-cloud-starter-netflix-eureka-client</artifactId>  
</dependency>

<!-- Spring Boot Starter Test (For Unit Testing) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

**3. application.properties Configuration**

Configure the H2 database and Eureka client in the application.properties file:

# Database Configuration for H2

spring.application.name=EeduexcellenceStudentMS  
spring.data.mongodb.database=student-management-db  
spring.data.mongodb.host=studentmanagementdb  
spring.data.mongodb.port=27017

# Eureka client settings

spring.application.name= EeduexcellenceStudentMS

eureka.client.serviceUrl.defaultZone=http://localhost:8761/eureka

**4. Create the Student Entity**

package com.eduexcellence.studentms.model;  
  
  
  
import org.springframework.data.annotation.Id;  
import org.springframework.data.annotation.Transient;  
import org.springframework.data.mongodb.core.mapping.Document;  
  
@Document(collection = "Student")  
public class Student {  
  
  
 @Transient  
 public static final String SEQUENCE\_NAME = "users\_sequence";  
 @Id  
// @GeneratedValue(strategy = GenerationType.IDENTITY)  
 private int id;  
 private int rollNo;  
// @NotNull(message = "First Name must not be null")  
 private String firstName;  
// @NotNull(message = "Last Name must not be null")  
 private String lastName;  
// @NotNull(message = "Address must not be null")  
 private String address;  
// @NotNull(message = "Phone No must not be null")  
 private String phoneNo;  
// @NotNull(message = "Grade must not be null")  
// @Min(value = 1, message = "Grade must be between 1-12")  
// @Max(value = 12, message = "Grade must be between 1-12")  
 private int grade;  
// @OneToOne(mappedBy = "student", cascade = CascadeType.ALL)  
   
 public int getId() {  
 return id;  
 }  
 public void setId(int id) {  
 this.id = id;  
 }  
 public int getRollNo() {  
 return rollNo;  
 }  
 public void setRollNo(int rollNo) {  
 this.rollNo = rollNo;  
 }  
 public String getFirstName() {  
 return firstName;  
 }  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
 public String getLastName() {  
 return lastName;  
 }  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
 public String getAddress() {  
 return address;  
 }  
 public void setAddress(String address) {  
 this.address = address;  
 }  
 public String getPhoneNo() {  
 return phoneNo;  
 }  
 public void setPhoneNo(String phoneNo) {  
 this.phoneNo = phoneNo;  
 }  
 public int getGrade() {  
 return grade;  
 }  
 public void setGrade(int grade) {  
 this.grade = grade;  
 }  
  
  
}

**5. Create the StudentRepository**

package com.eduexcellence.studentms.repository;  
  
import java.util.List;  
  
import org.springframework.data.mongodb.repository.MongoRepository;  
  
import org.springframework.stereotype.Repository;  
  
import com.eduexcellence.studentms.model.Student;  
  
@Repository  
public interface StudentRepository extends MongoRepository<Student, Integer> {  
 public List<Student> findByGrade(int grade);  
}

**6. Create the StudentService Layer**

package com.eduexcellence.studentms.service;  
  
import java.util.List;  
import java.util.Random;  
  
import org.springframework.http.ResponseEntity;  
  
import com.eduexcellence.studentms.model.Student;  
  
  
  
public interface StudentService {  
  
 public ResponseEntity<List<Student>> getAllStudent();  
  
 public ResponseEntity<String> addNewStudent(Student student);  
  
 public ResponseEntity<Student> getStudentById(int id);  
  
 public ResponseEntity<String> updateStudentDetails(Student student);  
  
 public ResponseEntity<String> deleteStudent(int id);  
  
}

**7. Create the StudentController for REST APIs**

package com.eduexcellence.studentms.controller;  
  
import java.util.List;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.http.ResponseEntity;  
import org.springframework.web.bind.annotation.DeleteMapping;  
import org.springframework.web.bind.annotation.GetMapping;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.PostMapping;  
import org.springframework.web.bind.annotation.PutMapping;  
import org.springframework.web.bind.annotation.RequestBody;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RestController;  
import org.springframework.web.reactive.function.client.WebClient;  
  
import com.eduexcellence.studentms.model.FeeDetails;  
import com.eduexcellence.studentms.model.Student;  
import com.eduexcellence.studentms.service.SequenceGeneratorService;  
import com.eduexcellence.studentms.service.StudentService;  
  
  
@RestController  
@RequestMapping("/api/v1")  
public class StudentController {  
  
 @Autowired  
 WebClient.Builder webClientBuilder;  
   
 @Autowired  
 private StudentService studentService;  
 @Autowired  
 private SequenceGeneratorService sequenceGeneratorService;  
  
  
 @GetMapping("/getMsg")  
 public String getMessage() {  
 return "Docker Test Successful";  
 }  
 @GetMapping(value="/student/get/all")  
 public ResponseEntity<List<Student>> getAllStudent(){  
 return studentService.getAllStudent();  
 }  
   
 @GetMapping(value="/student/get/id/{id}")  
 public ResponseEntity<Student> getStudentById(@PathVariable("id") int id){  
 return studentService.getStudentById(id);  
 }  
   
 @PutMapping(value="/student/update/details")  
 public ResponseEntity<String> updateStudentDetails(@RequestBody Student student){  
 return studentService.updateStudentDetails(student);  
 }  
   
 @DeleteMapping(value="/student/delete/{id}")  
 public ResponseEntity<String> deleteStudent(@PathVariable("id") int id){  
 return studentService.deleteStudent(id);  
 }  
   
 @PostMapping(value="/student/add/new")  
 public ResponseEntity<String> addNewStudent(@RequestBody Student student){  
 student.setId(sequenceGeneratorService.generateSequence(Student.*SEQUENCE\_NAME*));  
 return studentService.addNewStudent(student);  
 }  
   
 @GetMapping(value = "/student/get/fee/details/{id}")  
 public ResponseEntity<List<FeeDetails>> getFeesPaidByStudent(@PathVariable("id") int studentId){  
 return ResponseEntity.*ok*(webClientBuilder.build().get().uri("http://localhost:8080/api/v1/fee/get/details/"+studentId).retrieve().bodyToMono(List.class).block());  
 }  
}

**9. Main Application Class**

package com.eduexcellence.studentms;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.cloud.netflix.eureka.EnableEurekaClient;

@SpringBootApplication

@EnableEurekaClient

public class StudentMsApplication {

public static void main(String[] args) {

SpringApplication.run(StudentMsApplication.class, args);

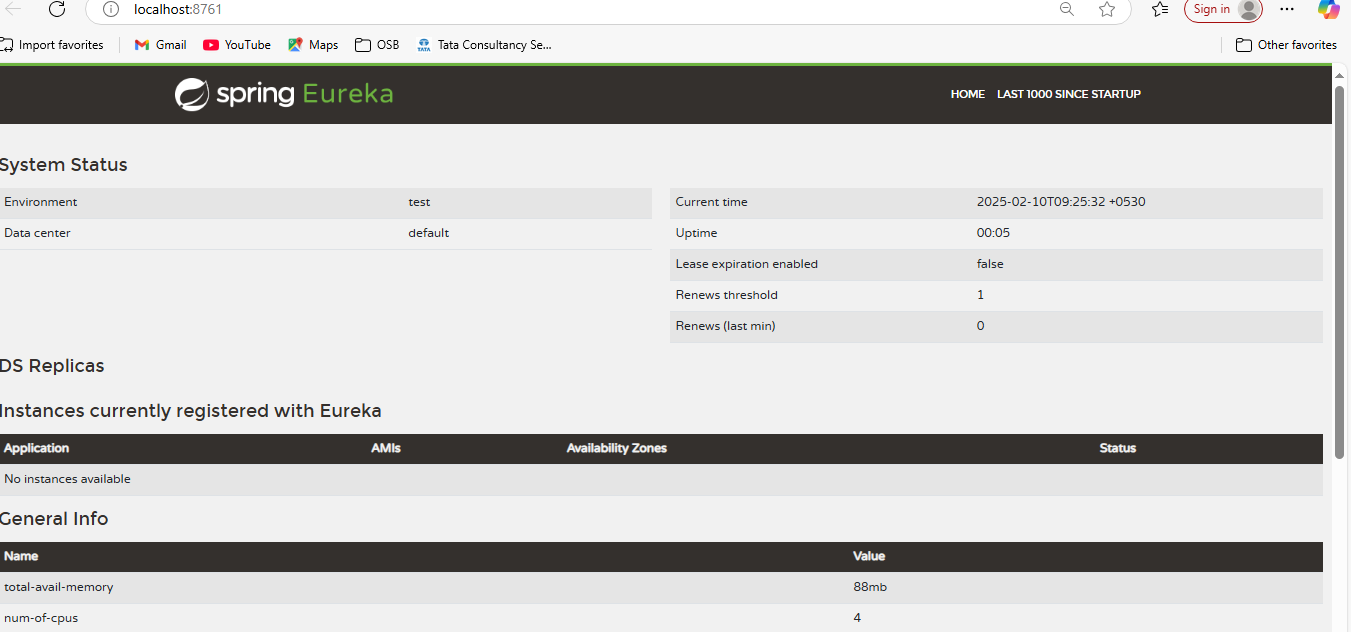
}

}

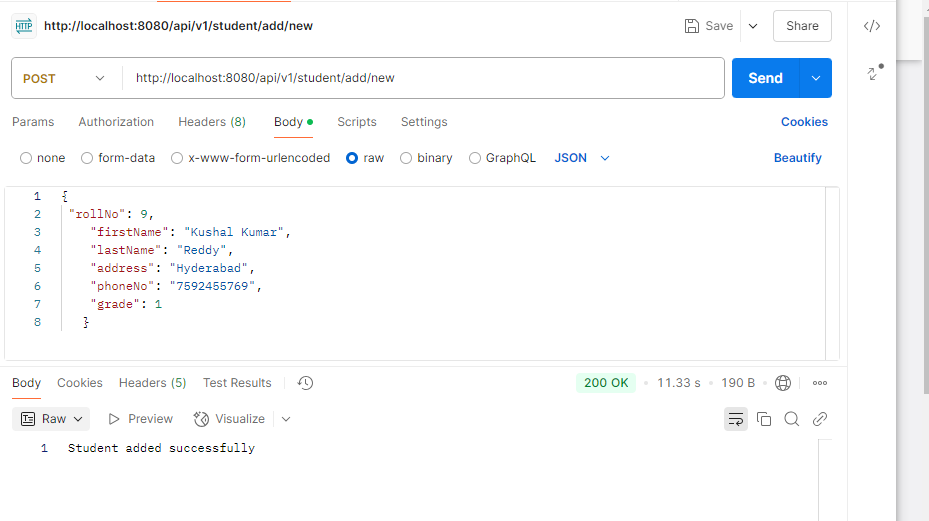
**10. Eureka Server Setup (for Service Discovery)**

You also need to set up a Eureka server for service discovery, which can be done by creating another Spring Boot project using **Spring Cloud Eureka Server**.

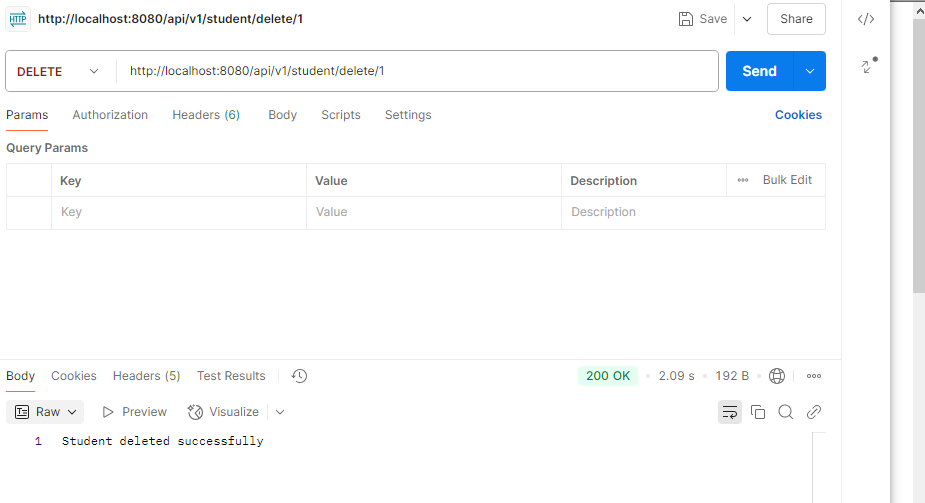
* Add the @EnableEurekaServer annotation to the EurekaServerApplication class.
* Configure the Eureka server in application.properties with server.port=8761.



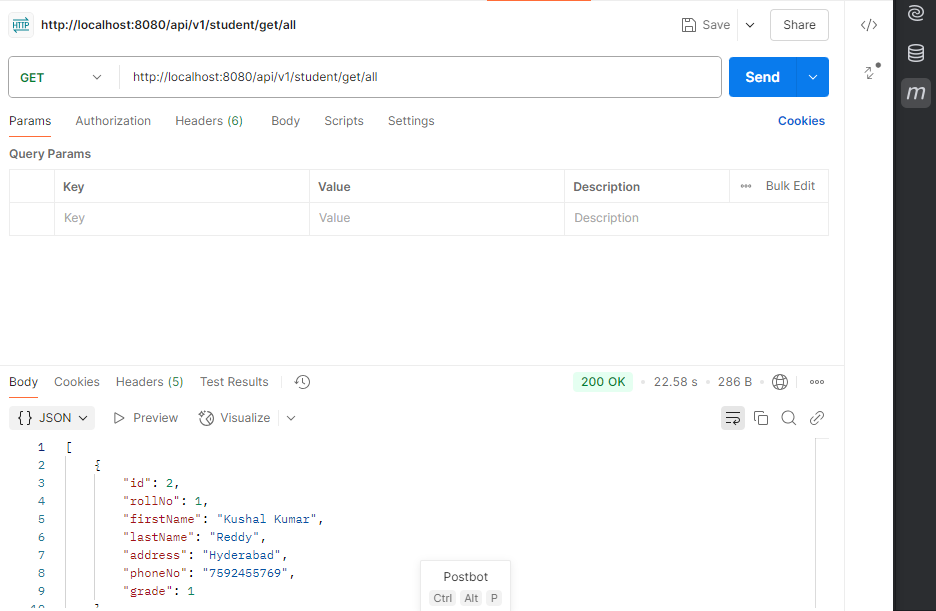
POST : http://localhost:8080/api/v1/student/add/new



http://localhost:8080/api/v1/student/delete/1



http://localhost:8080/api/v1/student/get/all



**11. Conclusion**

In the above implementation, we created a microservice to manage students with CRUD operations, including circuit breaker implementation using Resilience4j, service discovery with Eureka, and basic load balancing. You can test the API using Postman or curl commands to interact with the microservice.