TekGain E-Learning (Spring Boot + Microservices)

1.0 Introduction

1.0 Functional Requirements

TekGain is one of the popular training institutes that have branches all over India. Due to covid restrictions, TekGain planned to initiate E-Learning activities that help candidates all over the world to gain benefit. As an initiative they need to develop an application that supports the below business activities.

- Associate Service
- Admission Service

The client wishes to have microservices created for Associate and Admission, so that each team can work on independent services which will help them to deploy the services more quickly.

Help them to automate the above process by developing Rest Service using Maven and incorporate microservices for the same.

2.0 Technical Specifications

REST SERVICE LAYERING

Rest Services:

2.0.1 Associate Management Service

The provided AssociateController which is a RestController, should be created with the below services. All the below endpoints should return the ResponseEntity with below status:

1. For success – HttpStatus.OK
2. For Exception – HttpStatus.NOT_FOUND

Request Method	Request Url	Description
Post	/addAssociate	This service should add the associate details by using the addAsscoiate method of the AssociateServiceImpl class
Put	/updateAssociate/{associateId,associateAddr}	This service should modify the associate address of the given associate id and should return the updated associate details , by using the updateAssociate method of the AssociateServiceImpl class
Get	/viewByAssociateId/{associateId}	This service should retrieve the associate information of the given associate id by using the viewByAssociateId method of the AssociateServiceImpl class

GET	/viewAll	This service returns the list of associates using the viewAll method the AssociateServiceImpl class

2.0.3 Admission Service

The provided AdmissionController which is a RestController, should be created with the below services. All the below endpoints should return the ResponseEntity with below status:

1. For success – HttpStatus.OK
2. For Exception – HttpStatus.NOT_FOUND

Request Method	Request Url	Method in Controller	Description
post	/register/{registrationId}/{associateId}/{courseId}	registerAssociateForCourse	This service should register the given associate for the given course by using the registerAssociateForCourse method of the AdmissionServiceImpl class. **Note:1.** **This service should be implemented by performing intermicroservice communication with the Associate Service for checking the existence of the Associate.**
post	/feedback/{regNo,feedback}	addFeedback	This service should add the feedback for the course associated with the given regNo by using the addFeedback method of the AdmissionServiceImpl which internally uses the AdmissionRepository class.
get	/viewFeedbackByCourseId/{courseId}	viewFeedbackByCourseId	This service should return the feedback as a list for the given course id by using the viewFeedbackByCourseId method of the AdmissionServiceImpl
Delete	/deactivate/{courseId}	deactivateAdmission	This service should deactivate the admissions made for the given course by using the deactivateAdmission method of the AdmissionServiceImpl class.
get	/viewAll	viewAll	This service returns the list of associates using the viewAll

			method the AdmissionServiceImpl class

Service Layer

Service Layer for Associate Management Service

Refer to the IAssociateService interface provided as part of the code skeleton. The AssociateServiceImpl class which is provided as part of the code skeleton must realize all the methods in the IAssociateService interface.

AssociateServiceImpl class should be configured via annotation as Service.

For each method, use SLF4J along with Spring AOP to log the success and failure message into the log file named "log/mylog.log". For the success case, log the message as "The method <<methodname> has completed successfully and the log level should be info. For the Exception scenario, log the message given in each method's exception column and the log level should be error. Note: **Use Lombok annotations for logging.**

Requirements	Method in Service	Description	Exception
Requirement 1	addAssociate	This method should add the associate details to the Collection.	
Requirement 2	updateAssociate	This method should update the associate address for the given associate id by retrieving the existing associate information from the collection and update the same to the collection. The method should then return the associate information with the updated address.	If the given associate id does not exist, then throw AssociateInvalidException with the message **"Invalid Associate Id".**
Requirement 3	viewByAssociateId	This method should return the associate information available in the collection for the given associateId.	If the given associate id does not exist, then throw AssociateInvalidException with the message **"Invalid Associate Id".**
Requirement 4	viewAll	This method should return the list of associates available in the collection	

Service Layer for Admission Management Service

Refer to the IAdmissionService interface provided as part of the code skeleton. The AdmissionServiceImpl class which is provided as part of the code skeleton must realize all the methods in the IAdmissionService interface. **AdmissionServiceImpl** class should be configured via annotation as Service.

For each method, use SLF4J along with Spring AOP to log the success and failure message into the log file named "log/mylog.log". For the success case, log the message as "The method <<methodname> has completed successfully and the log level should be info. For the Exception scenario, log the message given in each method's exception column and the log level should be error. Note: **Use Lombok annotations for logging**

Requirements	Method in Service	Description	Exception
Requirement 1	registerAssociateForCourse	This method should add the admission object to the collection and return the Admission object.	
Requirement 2	addFeedback	This method should add the feedback for the given registration number into the collection and return the Admission object	If the given registration number does not exist, then throw AdmissionInvalidException with the message **"Invalid Registration Id".**
Requirement 3	viewFeedbackByCourseId	This method should return the feedback from the collection for the given course id.	If the given course id does not exist, then throw AdmissionInvalidException with the message **"Invalid Course Id".**
Requirement 4	deactivateAdmission	This method should deactivate the admission registered for the given course in the collection.	
Requirement 5	viewAll	This method should return the list of admissions available in the collection	

Model Class

Refer to the Associate class provided as part of the code skeleton.

Attributes	DataType
associateId	String
associateName	String
associateAddress	String
associateEmailId	String

Validations for Associate Service

Inbuilt Validation

Rule	Message when validation fails
associateName should not be null	Provide value for associate name
associateId should not be null	Provide value for associate id
associateAddress should not be null	Provide value for associate address

Perform the above Validations using annotations that are available in javax.validation.constraints package

Refer to the Admission class provided as part of the code skeleton.

Attributes	DataType
registrationId	Integer
courseId	Integer
associateId	Integer
Fees	Integer
feedback	String

Validations for Admission Management Service

Inbuilt Validation

Rule	Message when validation fails
associateId should not be null	Provide value for associate name
registrationId should not be null	Provide value for associate id
courseId should not be null	Provide value for associate address
fees should not ve negative or zero	Provide value greater than or equal to zero

Note:

- **Use Lombok to generate no-arg-constructor, AllArgsConstructor, getters and setters for the attributes.**
- **Do not change the datatype or the attribute name provided as part of the code skeleton.**

Eureka Registry

- The AssociateService and AdmissionService should be registered in the Eureka Registry. The Registry should run in the port number 8761.

API Gateway:

- Implement API Gateway using Spring cloud API Gateway
- The API gate way server port(8777)must be specified in the “application.properties” file.
- The Gateway should be configured to route only the services of admissionapp and associateapp.

GLOBAL EXCEPTION HANDLING

To incorporate the messages related to Exception as ResponseEntity, you are provided with the below classes

- Make use of ExceptionResponse class that is already provided as part of the code skeleton to send the customized response error message.
- CustomizedResponseEntityExceptionHandler class to handle all the exceptions that has occurred in the application(both user defined and pre-defined)

Method Name	**Explanation**
handleAllExceptions	This method should generally handle all the exceptions. It should return ResponseEntitywhich include ExceptionResponse object and the status as HttpStatus.INTERNAL_SERVER_ERROR
handleNotFoundException	This method should handle and provide customized error message using Exception Response class for user defined exception CourseInvalidException for CourseService, AdmissionInvalidException for AdmissionService and AssociateInvalidException for AssociateService It should return ResponseEntity which include ExceptionResponse object and the status as **HttpStatus.NOT_FOUND**
handleMethodArgumentNotValid	This method should handle and provide customized error message

	using Exception Response class for both pre-defined and user defined validations. It should return ResponseEntity which include ExceptionResponse object and the status as **HttpStatus.BAD_REQUEST**

Note:

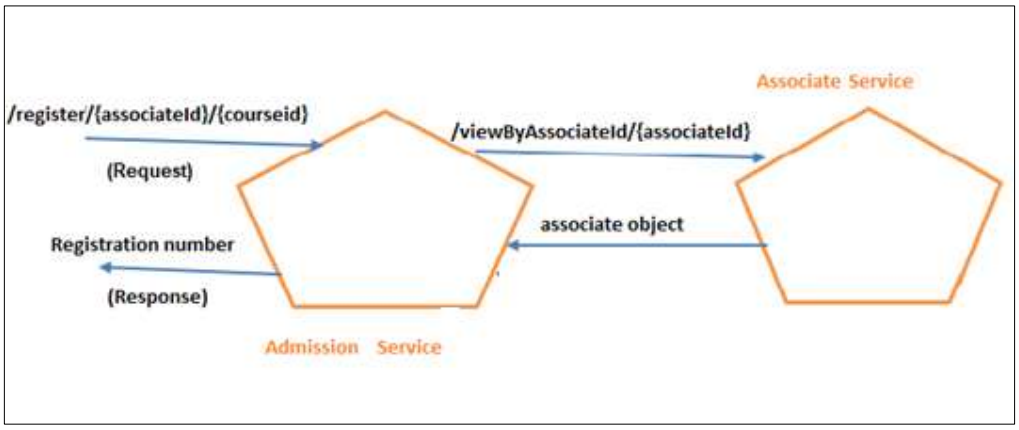
In CustomizedResponseEntityExceptionHandler class, log all the error messages using SLF4J.Use Lombok annotations for logging.

Logging

The LoggingAspect class is an AOP class, which is used to log the success and the failure messages. To log the message use Slf4j. The LoggingAspect class should be annotated with @Aspect.

Intermicroservice Communication

- register request for AdmissionService

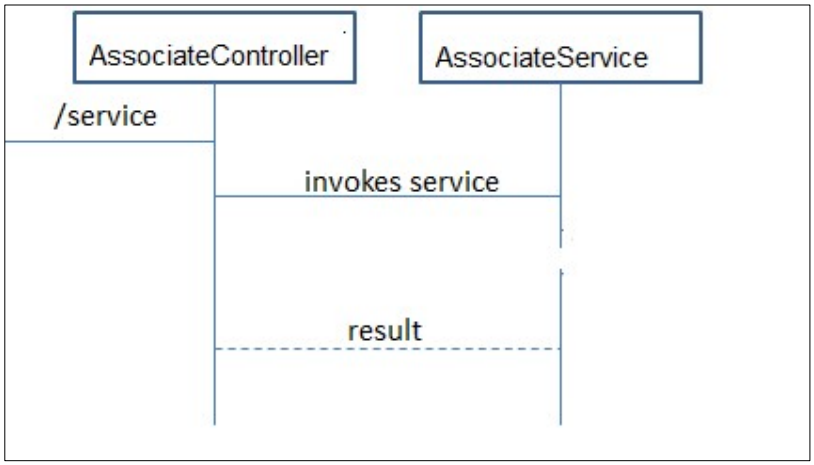


The url of the Associate service is specified in the application.properties file of Admission service as given below:

associateService.url=http://localhost:9092

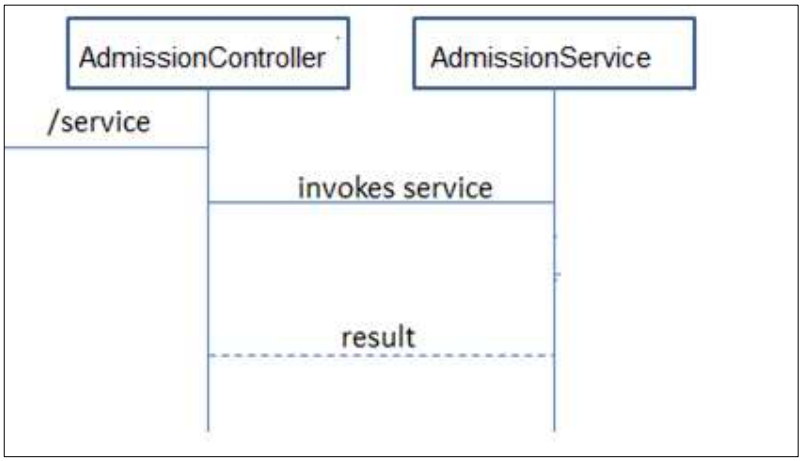
You can change the value of the associateService.url, but do not change the name.

Process Flow for Associate Service



- Client invokes the required service.
- AssociateController invokes the method of the AssociateServiceImpl.
- AssociateServiceImpl performs the service and returns the data back.
- To perform **logging**, you are provided with LogingAspect class. Log the success message and failure message using Slf4j and AOP. Configure logging using **Slf4j** annotation with Lombok in LoggingAspect class.

Process Flow for Admission Service



- Client invokes the required service.
- AdmissionController invokes the method of the AdmissionServiceImpl.
- AdmissionServiceImplperforms the service and returns the data back.
- To perform **logging**, you are provided with LogingAspect class. Log the success message and failure message using Slf4j and AOP. Configure logging using **Slf4j** annotation with Lombok in LoggingAspect class.

Overall Design Constraints

1) **Do not change the property name given in the application.properties files, you can change the value and you can include additional property if needed.**
2) **In the pom.xml you are provided with all the dependencies needed for developing the application. Do not change or add new dependencies.**

3) Adhere to the design specifications mentioned in the case study.
4) Do not change or delete the class/method names or return types which are provided to you as a part of the base code skeleton.
5) Developer will create 4 applications:
    - Spring Boot (**Name of the application Should be CourseService,AssociateService**)
    - Spring Cloud (Eureka-Server, spring-cloud-starter-eureka-server) (**Name of the application Should be cms-registry-server**)
    - Spring Cloud API GateWay(spring-cloud-starter-gateway) **(Name of the application Should be Gateway**)