Part B

Handson Problem (1 question)

TcsInterns is an application used by a company for managing their internship program. This application has following modules :

1. Allocate project : This module allocates a new project to a mentor.
2. Get mentor: This module fetches mentor details based on number of projects mentored.
3. Update mentor: This module updates mentor of a project.
4. Delete project : This module deletes project details.

You have to implement this application according to the description given below.

1. Entity classes (com.tcs.tcsinterns.entity)

   Class: Mentor
   - mentorId: Integer
   - mentorName: String
   - numberOfProjectsMentored: Integer

   Annotate this class with proper annotation to declare it as an entity class with
   mentorId as primary key.

   Map this class with mentor table.

   Class: Project
   - projectId: Integer
   - projectName: String
   - ideaOwner: Integer
   - releaseDate: LocalDate
   - Mentor: Mentor

Annotate this class with proper annotation to declare it as an entity class with
projectId as primary key.

Map this class with project table.

Generate the projectId using the IDENTITY strategy

There is Many to One unidirectional relationship from Project to Mentor

In the project table, mentor_id is the foreign key column referencing mentor table

2. DTO classes (com.tcs.tcsinterns.dto)

Class: MentorDTO
- mentorId: Integer
- mentorName: String
- numberOfProjectsMentored: Integer

This is a DTO (Data Transfer Object) class that holds mentor details such as
mentor Id, mentor name and number of projects mentored. Use appropriate
annotations from the Java Bean Validation API for validating mentor Id attribute of this class. (Refer below allocateProject method of ProjectAllocationAPI class for validation rules).

Class: ProjectDTO
- projectId: Integer
- projectName: String
- ideaOwner: Integer
- releaseDate: LocalDate
- mentorDTO: MentorDTO

This is a DTO (Data Transfer Object) class that holds project details such as project Id, project name, idea owner, release date and the mentor details.

Use appropriate annotations from the Java Bean Validation API for validating project name, idea owner, release date and the mentor details attributes of this class. (Refer below allocateProject method of ProjectAllocationAPI class for validation rules).

3. Repository classes (com.tcs.tcsinterns.repository)

Class: MentorRepository

- This interface should extend the appropriate repository interface.
- Add the following method to the interface

  The method should accept the number of projects mentored and return details of all the mentors who are mentoring the given number of projects.

Class: ProjectRepository

- This interface should extend the appropriate repository interface

4. Service classes (com.tcs.tcsinterns.service)

Class: ProjectAllocationServiceImpl

- projectRepository: ProjectRepository
- mentorRepository: MentorRepository

This is service class of the application. Implement this class according to given class diagram and description.

Inject the ProjectRepository and MentorRepository objects using the proper annotation.

Annotate it with appropriate stereotype annotation with value as “projectService”.

Annotate it with appropriate annotation for managing transactions.

Method Description

public Integer allocateProject(ProjectDTO project)

- This method is used to allocate a new project to an existing mentor.
- Invoke appropriate method of MentorRepository which will retrieve the mentor details using the given mentorId (available in ProjectDTO).
- If mentor does not exist, then throw an object of TcsInternException with message “Service.MENTOR_NOT_FOUND”
- If mentor exists and the number of projects mentored by retrieved mentor is greater than or equal to 3, then throw an object of TcsInternException with message “Service.CANNOT_ALLOCATE_PROJECT”
- Else, create a new Project object and populate it with received ProjectDTO details.
- The above created Project should be allocated to the retrieved Mentor. After successful allocation, the numberOfProjectsMentored by the corresponding mentor must be incremented by 1.
- The project details should be saved to the database and return the generated projectId.

public List<MentorDTO> getMentors(Integer numberOfProjectsMentored)

- This method is used to fetch Mentor details who have mentored projects equal to numberOfProjectsMentored.
- Invoke appropriate method of MentorRepository, to fetch mentor details based on numberOfProjectsMentored which in turn returns a list.

- If the list is empty, throw an object of TcsInternException with message “Service.MENTOR_NOT_FOUND” .
- Else, for each mentor found, create and populate the MentorDTO object and add it to a List<MentorDTO>.
- Return the above obtained list

public void updateProjectMentor (Integer projectId, Integer mentorId)

- This method is used to update the mentor of a project.
- Invoke appropriate method of MentorRepository to retrieve the mentor details using the given mentorId.
- If mentor does not exist, then throw an object of TcsInternException with message “Service.MENTOR_NOT_FOUND”
- If the numberOfProjectsMentored of the retrieved Mentor is greater than or equal to 3, then throw an object of TcsInternException with message “Service.CANNOT_ALLOCATE_PROJECT”
- Invoke appropriate method of ProjectRepository which will retrieve the project details using the given projectId.
- If project does not exist, then throw an object of TcsInternException with message “Service.PROJECT_NOT_FOUND”
- Else, update the mentor of the retrieved project with the above retrieved mentor. Also, increment the numberOfProjectsMentored of the corresponding retrieved Mentor by 1.

public void deleteProject(Integer projectId)

- This method is used to delete the project details only and not the mentor who is allocated the project.
- Invoke appropriate method of ProjectRepository which will retrieve the project details using the given projectId.
- If the retrieved project does not exist, then throw an object of TcsInternException with message “Service.PROJECT_NOT_FOUND”

- If the project is not allocated to any mentor, then delete the project details by invoking appropriate method of ProjectRepository.
- If the project is allocated to a mentor, then deallocate the project from the mentor and delete the project only. Also, the numberOfProjectsMentored of the corresponding mentor must be decremented by 1.

5. API Classes (com.tcs.tcsinterns.api)

Class: ProjectAllocationAPI

- projectService: ProjectAllocationService
- environment: Environment

This is API class of application. Implement this class according to the given class diagram and description.

Annotate it with proper annotation to declare it as REST controller class.

Annotate it with proper annotation enable path variable validation support.

Annotate it with proper annotation so that all its methods are mapped with “tcsinterns” as base URI.

Inject ProjectAllocationService and Environment objects using appropriate annotation.

Method Description

public ResponseEntity<String> allocateProject (ProjectDTO project)

- This is a REST controller method to add new project with mentor allocation details.
- It accepts POST request with URI “/project”.
- It receives project details in the form of JSON. The JSON data should be populated in ProjectDTO parameter of this method after successful validation.

- The project details will consist of project name, idea owner, release date and the mentor details. Using Bean Validation API validate the details according to the description given below. (Note: The validation message should not be hard-coded. It should be read from the ValidationMessages.properties file)
  - The projectName should not be null. If null set validation message as Please provide project nameThe ideaOwner should not be null. If null set validation message as Please provide idea owner name
  - The releaseDate should not be null. If null set validation message as Please provide project release date
  - The mentorDTO should not be null. If null set validation message as Please provide mentor details
  - Also, MentorDTO object should be validated, annotate mentorDTO with proper annotation for bean validation.
  - The mentorId of MentorDTO should not be null. If null set validation message as Please provide mentor id.
  - The mentorId of MentorDTO should be of 4 digits. If invalid set validation message as Mentor id should be of 4 digits.
- Invoke the allocateProject method of ProjectAllocationServiceImpl by passing the ProjectDTO object received as parameter, which in turn returns a projectId.
- Retrieve the success message associated with property API.ALLOCATION_SUCCESS from application.properties files using environment and append it to projectId as <success message>: projectId.
- Create an object of ResponseEntity using the above created string as message and HTTP status code as *CREATED* and return it.

public List<MentorDTO> getMentors (Integer numberOfProjectsMentored)

- This is a REST controller method to fetch mentor details based on the given numberOfProjectsMentored.
- It accepts GET request with URI “mentor/{ numberOfProjectsMentored }”.

- Invoke the getMentors method of ProjectAllocationServiceImpl by passing the numberOfProjectsMentored received, which in turn returns a List<MentorDTO>
- Create an object of ResponseEntity using the above List<MentorDTO>, with HttpStatus as *OK* and return it.

public ResponseEntity<String> updateProjectMentor (Integer projectId, Integer mentorId)

- This is a REST controller method to update the mentor details of a given project.
- It accepts PUT request with URI “project/{ projectId }/{ mentorId }”
- Using Bean Validation API validate the path variable mentorId according to the description given below. (Note: The validation message should not be hard-coded. It should be read from the ValidationMessages.properties file)
  - mentorId should be of 4 digits. If invalid mentorId, set validation message as Mentor id should be of 4 digits.
- Invoke the updateProjectMentor method of ProjectServiceImpl by passing the projectId and mentorId received.
- Retrieve the success message associated with property API.PROJECT_UPDATE_SUCCESS from application.properties files using environment.
- Create an object of ResponseEntity using the above created string as message and HTTP status code as *OK* and return it.

public ResponseEntity<String> deleteProject (Integer projectId)

- This is a REST controller method to delete a project.
- It accepts DELETE request with URI “project/{ projectId }"
- Invoke the deleteProject method of ProjectService by passing the projectId received.

- Retrieve the success message associated with property API.PROJECT_DELETE _SUCCESS from application.properties files using environment.
- Create an object of ResponseEntity using the above created string as message and HTTP status code as *OK* and return it.

6. Utility classes (com.tcs.tcsinterns.utility)

Class: LoggingAspect

- LOGGER: Log

- This is AOP aspect class to log the exceptions. Implement this class according to the given class diagram and description.
- Annotate this class with proper annotation to declare and use it as AOP aspect.

Method Description

- public logServiceException(TcsInternException exception)
    - Mark this method with annotation to declare it as an after throwing advice
    - The method should log all the exceptions thrown from the methods of com.tcs.service.ProjectAllocationServiceImpl class at error level

Class: ExceptionControllerAdvice

- LOGGER: Log
- environment: Environment

This is exception handler class to log and handle the exceptions. Implement this class according to given class diagram and description.

Annotate this class with proper annotation to declare it as controller advice.

Inject Environment object using proper annotation.

The body of the methods have been implemented.

Method Description

public ResponseEntity<ErrorInfo> meetingSchedulerExceptionHandler (TcsInternException exception)

- This method is for handling TcsInternException exception
- Add appropriate annotation to make this method as exception handler for the TcsInternException

public ResponseEntity<ErrorInfo> generalExceptionHandler (Exception exception)

- This method is for handling general exceptions
- Add appropriate annotation to make this method as exception handler for any type of exceptions

public ResponseEntity<ErrorInfo> validatorExceptionHandler(Exception)

- This method is for handling MethodArgumentNotValidException and ConstraintViolationException exceptions

- Add appropriate annotation to make this method as exception handler method for MethodArgumentNotValidException and ConstraintViolationException exceptions

7. Test class (com.tcs.tcsinterns)

Class: TcsInternsApplicationTests

- mentorRepository: MentorRepository
- projectAllocationService: ProjectAllocationService

This is test class to test the methods of ProjectAllocationServiceImpl class using JUnit 5 and Mockito. Implement this class according to given class diagram and description.

Annotate this test class to execute Spring Boot based test

Annotate mentorRepository and projectAllocationService for mocking using Mockito.

Annotate the methods with appropriate annotation to mark them as JUnit 5 test cases

Method description

public void allocateProjectCannotAllocateTest()

- This test case is used to test for exception if mentorId is already mentoring more than 3 projects.
- Take any invalid value for mentorId and valid values for other member variables.

public void allocateProjectMentorNotFoundTest()

- This test case is used to test for exception if mentorId is not found.

- ○ Take any invalid value for mentorId and valid values for other member variables.