Spring Boot RESTful DataJPA

Spring Boot RESTtful Webservices Assignment

OBJECTIVE

The objective of this assignment is to get hands-on the following topics:

1. Spring Boot
2. Creating RESTful Webservices using Spring Boot
3. Spring Data JPA
4. Testing RESTful Webservices using Postman

PROBLEM STATEMENT

In this assignment, we will create a “loan-service” RESTful Webservice using Spring Boot and Data JPA

EXPECTED SOLUTION

~~Participant has to create the following REST end points to perform various operations on the~~ loan application as mentioned below

END Point	Return Type	Description
api/v1/applyloan	JSON response of the applied loan	This end point should be mapped to applyLoan() method in LoanApplicationController which takes the loanApplication as input and has to call LoanApplicationService saveLoanApplicationToDb() method to store in the database. This method should return the JSON response of the applied loan along with the proper HTTP status codes and should also handle the exceptions
api/v1/approveloan/{loanId}	JSON response of the applied loan	This end point should be mapped to applyLoan() method in LoanApplicationController which takes the loanId as input and has to call LoanApplicationService approveLoanApplication() method to store in the database. This method should return the JSON response of the applied loan along with the proper HTTP status codes and should also handle the exceptions

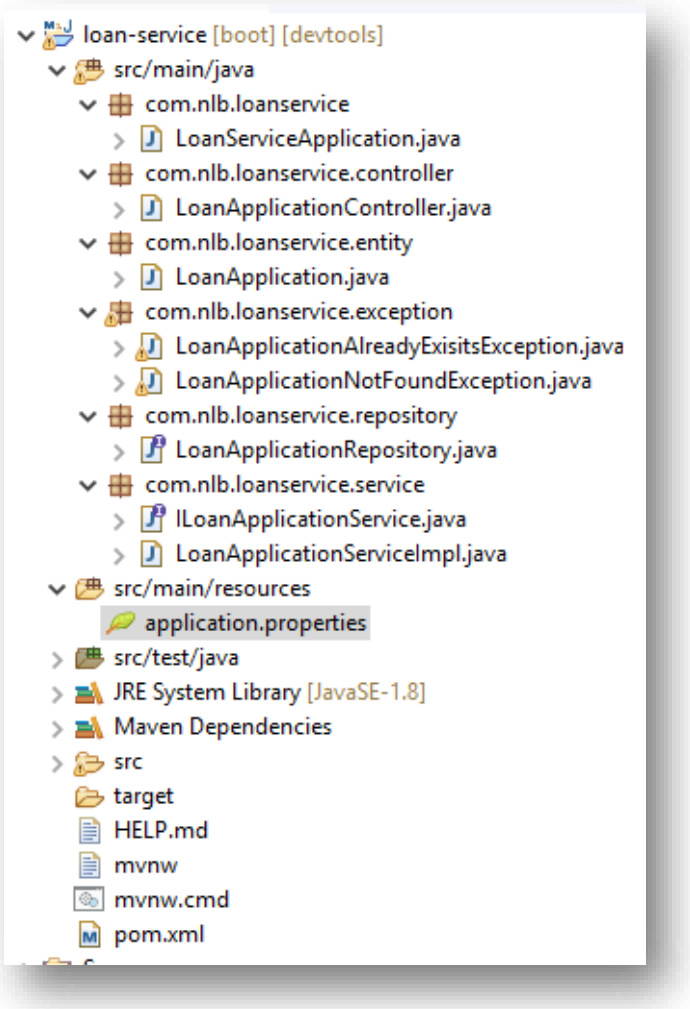
api/v1/rejectloan/{loanId}	JSON response of the applied loan	This end point should be mapped to applyLoan() method in LoanApplicationController which takes the loanId as input and has to call LoanApplicationService rejectLoanApplication() method to store in the database. This method should return the JSON response of the applied loan along with the proper HTTP status codes and should also handle the exceptions

HTTP Status Codes

Code	**Description**
200	OK , If the operations is successful
201	Created , if a new resource is created
400	Bad Request, if the server couldn't able to process the request due to some error in client side
401	Unauthorized, If the authentication is failed
404	Not Found, if the specified resource is not fount
409	Conflict, If the specified resource is already existing
500	Internal Server Error, due to some unexpected condition at server side

STEPS

Step1 : Create a spring boot project “loan-service” as per the given structure



Step2 : Add the following dependencies in pom.xml

1) spring-boot-starter-data-jpa
2) spring-boot-devtools
3) mysql-connector-java

Step3: LoanApplicationController

1) Should be annotated with @RestController
2) Should be annotated with @RequestMapping(“api/v1”)
3) Define the necessary methods as per the above-mentioned end points

Step4: LoanApplication

1) Shuld be annotated with @Entity
2) Define the following properties
    - Id (primary key)
    - bankName
    - minLoanAmount
    - maxLoanAmount
    - minInterestRate
    - minCreditScore
    - termLength
    - processingFee
    - rating
    - Status
3) Implement no-arg constructor
4) Implement getter and setter methods for each property

Step5: LoanApplicationAlreadyExistsException , LoanApplicationNotFoundException

1) Create both these classes extending from Exception class
2) Should return the response with the HttpStatus code and message when these exceptions are thrown from controller

Step6: LoanApplicationRepository interface

This interface should extends from JpaRepository

Step6: ILoanApplicationService interface

This interface should contain the following methods

Return Type	**Method**	**Description**
LoanApplication	saveLoanApplication(LoanApplication) throws LoanApplicationAlreadyExistsException	This method should be an abstract method

LoanApplication	approveLoanApplication(int loanId) throws LoanApplicationNotFoundException	This method should be an abstract method
LoanApplication	rejectLoanApplication(int loanId) throws LoanApplicationNotFoundException	This method should be an abstract method

Step7: LoanApplicationServiceImpl class

This class should implment ILoanApplicationService and override the above-mentioned methods and provide the business logic to perform the expected task.

Step8: application.properties file

Define the following properties

1. spring.application-name
2. server.port
3. spring.datasource.url
4. spring.datasource.username
5. spring.datasource.password
6. spring.jpa.properties.hibernate.dialect
7. spring.jpa.show-sql
8. spring.jpa.hibernate.ddl-auto

Step9 : Test all the above REST end points using POSTMAN