A PROJECT REPORT ON

BOOKSTORE APPLICATION

Submitted by

Ms. Aishwarya Sanku

Mr. Appa Gange

Mr. Bhairulal Jangid

Ms. Kakshya Mahathi

Ms. Masadi Shravani

Batch No. 7397

Under the Guidance of

Trainer Mrs. Indrakka Mali Maam

INDEX

Introduction:

The project “Book Store Application” is developed using spring boot framework, which mainly focuses on basic operations of a book store. Like the information of inserting, deleting, updating and getting all records of available books .

Admin Module:

The Admin module has the controller to access the bookstore. The admin can insert, update, delete and control the purchase the books in bookstore. The Admin has the access to add the books.

User Module:

The User has access to visit the bookstore. Where the User, will purchase and add the books to the wishlist to buy them when needed. The User module has to connect with the application and can search the books they need. The user visits the site may select the desired book.

Seller Module:

The will act as an mediator between the store and User .This module enables the user to buy the books what they add. The Seller Module provides at the comfort to the User and store to have the proper market space.

This project e-bookstore makes the business to take place in a very certain way to have a fair and friendly access to the store. The store management by the admin has the access to the database and to maintain the updated list according to the availability , the seller and the user interaction according to the availability makes the process simple and efficient.

Requirements:

Software Requirement:

Database: MySQL

API- Spring Data JPA, spring web, spring security

Tools**:** SwaggerUi and Postman, IDE-Spring Tool Suit4

Coding language-Java 1.8

Hardware Requirements:

RAM: 2GB

Processor: 64bit

Memory: 512 MB

Disk Space: 100GB

Spring Tool Suit: STS is an Eclipse-based development environment that is customized for the development of spring applications. It provides a ready-to-use environment to implement, debug, run and deploy your applications.

Postman: Postman is a standalone software testing API (Application Programming Interface) platform to build, test, design, modify, and document APIs. It is a simple Graphic User Interface for sending and viewing HTTP requests and responses.

MySQL:

- MySQL is a relational database management system.
- MySQL is open-source.
- MySQL is free.
- MySQL is ideal for both small and large applications.
- MySQL is very fast, reliable, scalable, and easy to use.

▪ MySQL is cross-platform.

Annotations:

1.@Service:

We mark beans with @Service to indicate that they're holding the business logic. Besides being used in the service layer, there isn't any other special use for this annotation.

2. @Repository:

@Repository's job is to catch persistence-specific exceptions and re-throw them as one of spring's unified unchecked exceptions.

3. @Autowired:

The spring framework enables automatic dependency injection. In other words, by declaring all the bean dependencies in a spring configuration file, Spring container can autowire relationships between collaborating beans. This is called spring bean autowiring.

4. @GetMapping:

The @GetMapping annotation is a specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method = RequestMethod.GET).

5. @PostMapping:

The @PostMapping is specialized version of @RequestMapping annotation that acts as a shortcut for @RequestMapping(method=RequestMethod.POST). The @PostMapping annotated methods in the @Controller annotated classes handle the HTTP POST requests matched with given URI expression.

6. @Configuration:

Spring @Configuration annotation helps in spring annotation based configuration. @Configuration annotation indicates that a class declares one or

more @Bean methods and may be processed by the spring container to generate bean definitions and service requests for those beans at runtime.

7. @OneToMany:

A one-to-many relationship between two entities is defined by using the @OneToMany annotation in Spring Data JPA. It declares the mappedBy element to indicate the entity that owns the bidirectional relationship. Usually, the child entity is one that owns the relationship and the parent entity contains the @OneToMany annotation.

8. @generated Value:

Marking a field with the @GeneratedValue annotation specifies that a value will be automatically generated for that field. This is primarily intended for primary key fields but Object DB also supports this annotation for non-key numeric persistent fields as well.

9. @entity:

The @Entity annotation specifies that the class is an entity and is mapped to a database table. The @Table annotation specifies the name of the database table to be used for mapping.

10. @ Exception Handler:

The @ExceptionHandler is an annotation used to handle the specific exceptions and sending the custom responses to the client.

11. @ControllerAdvice:

@ControllerAdvice is a specialization of the @Component annotation which allows to handle exceptions across the whole application in one global handling component. It can be viewed as an interceptor of exceptions thrown by methods annotated with @RequestMapping and similar.

12. @BeanAnnotation:

Spring @Bean annotation tells that a method produces a bean to be managed by the spring container. It is a method-level annotation. During Java configuration (@Configuration), the method is executed and its return value is registered as a bean within a BeanFactory.

13. @ApiOperation:

The *@ApiOperation* annotation is used to describe a single operation**.** An operation is a unique combination of a path and an HTTP method. Additionally, using *@ApiOperation*, we can describe the result of a successful REST API call. In other words, we can use this annotation to specify the general return type.

14. @ResponseBody

When you use the @ResponseBody annotation on a method, Spring converts the return value and writes it to the HTTP response automatically. Each method in the Controller class must be annotated with @ResponseBody.

15. @CrossOrigin:

cross-origin HTTP request is a request to a specific resource, which is located at a different origin, namely a domain, protocol and port, than the one of the client performing the request**.**

16. @RestController:

@RestController is a specialized version of the controller. It includes the *@Controller* and *@ResponseBody* annotations.The controller is annotated with the *@RestController* annotation; therefore, the *@ResponseBody* isn't required. Every request handling method of the controller class automatically serializes return objects into *HttpResponse*.

17. @Data:

Lombok annotation @Data tell to the IDE or build tool like Maven or Ant, to generate all the boilerplate code during compile time. So, that user need not to create this code every time when created fields, developer can access all these methods as usual within IDE while writing code from the generated .class file.

18. @Component:

@Component is a class-level annotation. It is used to denote a class as a Component. We can use @Component across the application to mark the beans as Spring's managed components. A component is responsible for some operations. Spring framework provides three other specific annotations to be used when marking a class as a Component.

1. @Service
2. @Repository
3. @Controller

19. @Table:

The @Table annotation allows you to specify the details of the table that will be used to persist the entity in the database. The @Table annotation provides four attributes, allowing you to override the name of the table, its catalog, and its schema, and enforce unique constraints on columns in the table.

20. @Id:

The @Id annotation is inherited from *javax.persistence.Id,* indicating the member field below is the primary key of the current entity.

21. @Override:

The @Override annotation denotes that the child class method overrides the base class method.

22. @Service:

The business logic resides within the service layer so we use the @Service Annotation to indicate that a class belongs to that layer.

23. @Transactional:

One of the most essential parts of Spring MVC is @Transactional annotation, which provides broad support for transaction management and allows developers to concentrate on business logic rather than worrying about data integrity in the event of system failures.

24. @Runwith:

@RunWith annotation can still be used in JUnit 5 for the sake of backward compatibility.

25. @InjectMocks:

It marks a field or parameter on which the injection should be performed. It allows shorthand mock and spy injections and minimizes the repetitive mocks and spy injection. In Mockito, the mocks are injected either by setter injection, constructor injection, and property injection. The @InjectMocks annotation is available in the org.mockito package.

26. @BeforeEach:

The @BeforeEach annotation is one of the test lifecycle methods and is the replacement of @Before annotation in JUnit 4.By default, the test methods will be executed in the same thread as @BeforeEach annotated method.

27. @Test:

The Test annotation tells JUnit that the public void method to which it is attached can be run as a test case. To run the method, JUnit first constructs a fresh instance of the class then invokes the annotated method. Any exceptions thrown by the test will be reported by JUnit as a failure. If no exceptions are thrown, the test is assumed to have succeeded.

28. @Mock:

It is used to mock the objects that helps in minimizing the repetitive mock objects. It makes the test code and verification error easier to read as parameter names (field names) are used to identify the mocks. The @Mock annotation is available in the org.mockito package.

29. @Query:

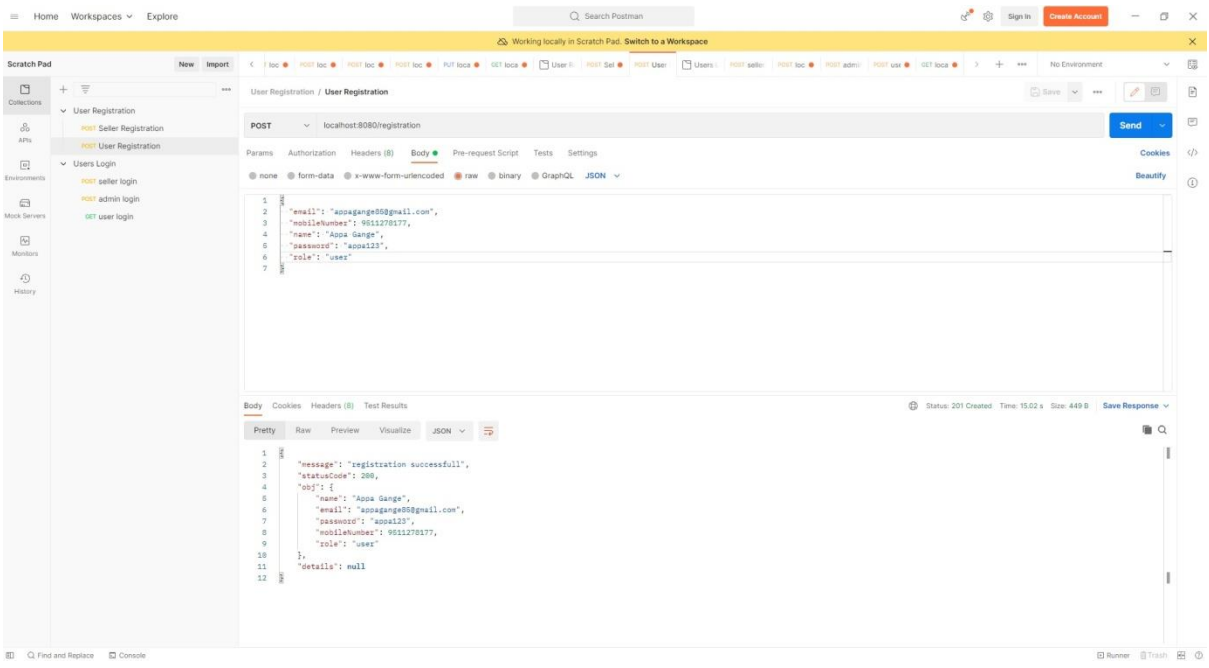
In order to define SQL to execute for a Spring Data repository method, we can annotate the method with the ***@Query*** annotation — its ***value*** attribute contains the JPQL or SQL to execute**.** The *@Query* annotation takes precedence over named queries, which are annotated with *@NamedQuery* or defined in an *orm.xml* file.

30. @Modifying:

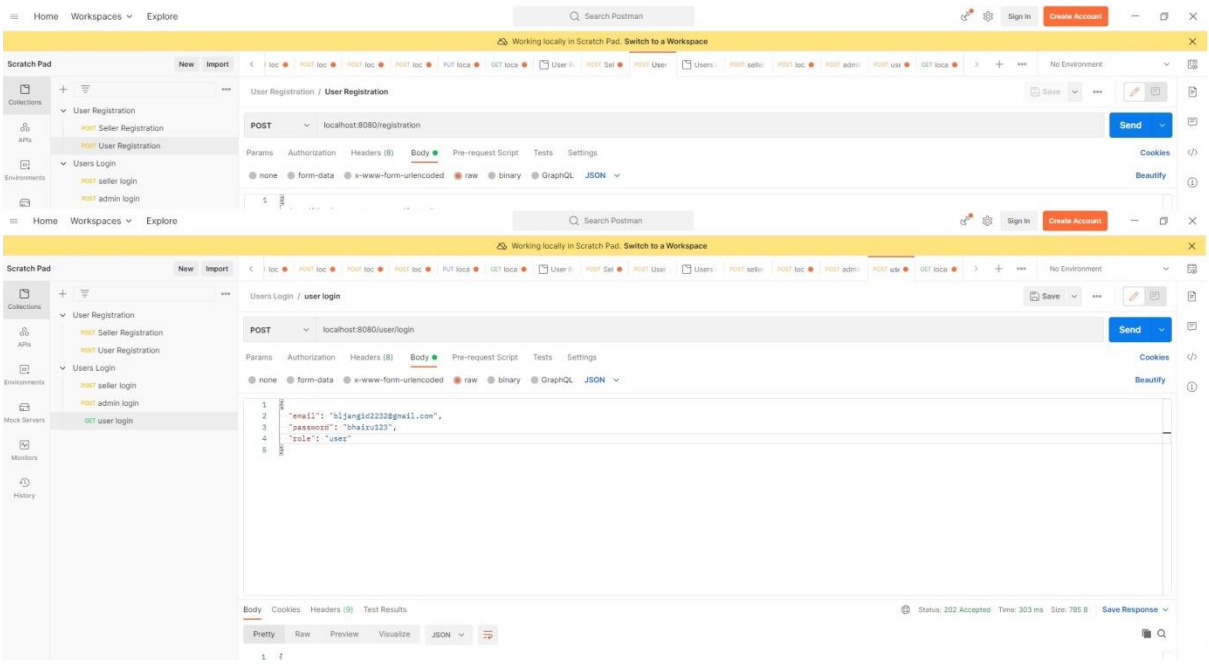
Spring Data JPA provides @Modifying annotation, using this annotation we can write named parameters query using @Query annotation and we can insert, update, or delete an entity.

Screenshots/Outputs:

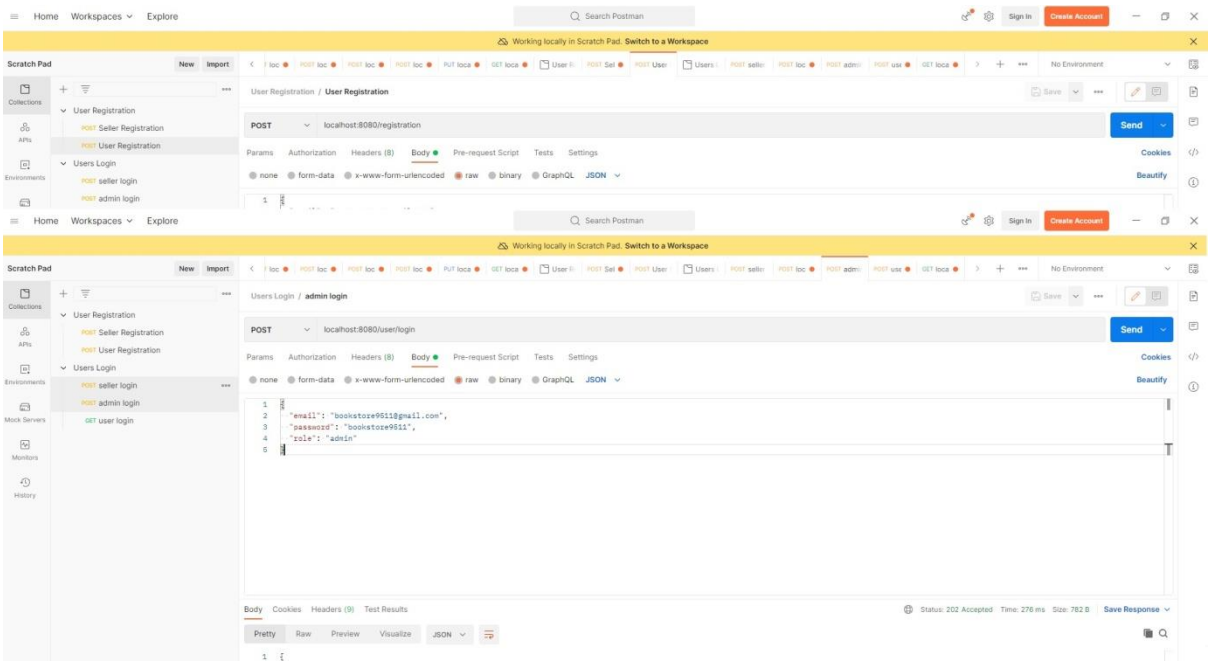
a)Registration of user:



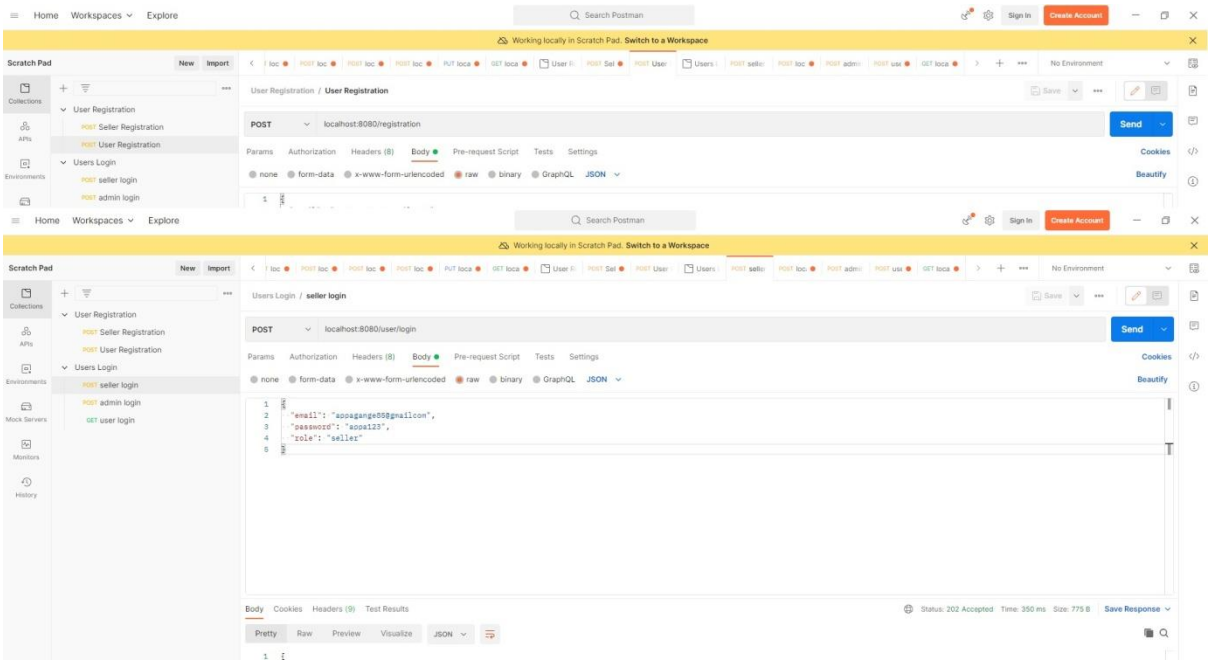
b)Userlogin:



c)adminlogin:



d)Sellerlogin



Conclusion:

This project e-bookstore makes the business to take place in a very certain way to have a fair and friendly access to the store. The store management by the admin has the access to the database and to maintain the updated list according to the availability , the seller and the user interaction according to the availability makes the process simple and efficient.

THANK YOU