Springboot coding challenge

1)

Build a Library Management Application in Spring Boot (Beginner Level)

Project description:

The government is building a centralized library management system that involves a micro-service, allowing people to issue books and deposit them. This micro-service is built using the Spring Boot framework and connects to the database using the Java Persistence API (JPA).

The application exposes the following four APIs:

1. /api/v1/user [POST]: An API to Add a new user.
2. /api/v1/user [GET]: An API to Fetch the details of a user with id.
3. /api/v1/book [POST]: An API toAdd a new user.
4. /api/v1/book/{id} [GET]: An API to Fetch the details of a book.

Your task is to complete the files:

- src/main/java/com/wecp/library/controller/LibraryController.java
- src/main/java/com/wecp/library/repository/BookRepository.java
- src/main/java/com/wecp/library/repository/UserRepository.java

You can find blank JPA repositories here:

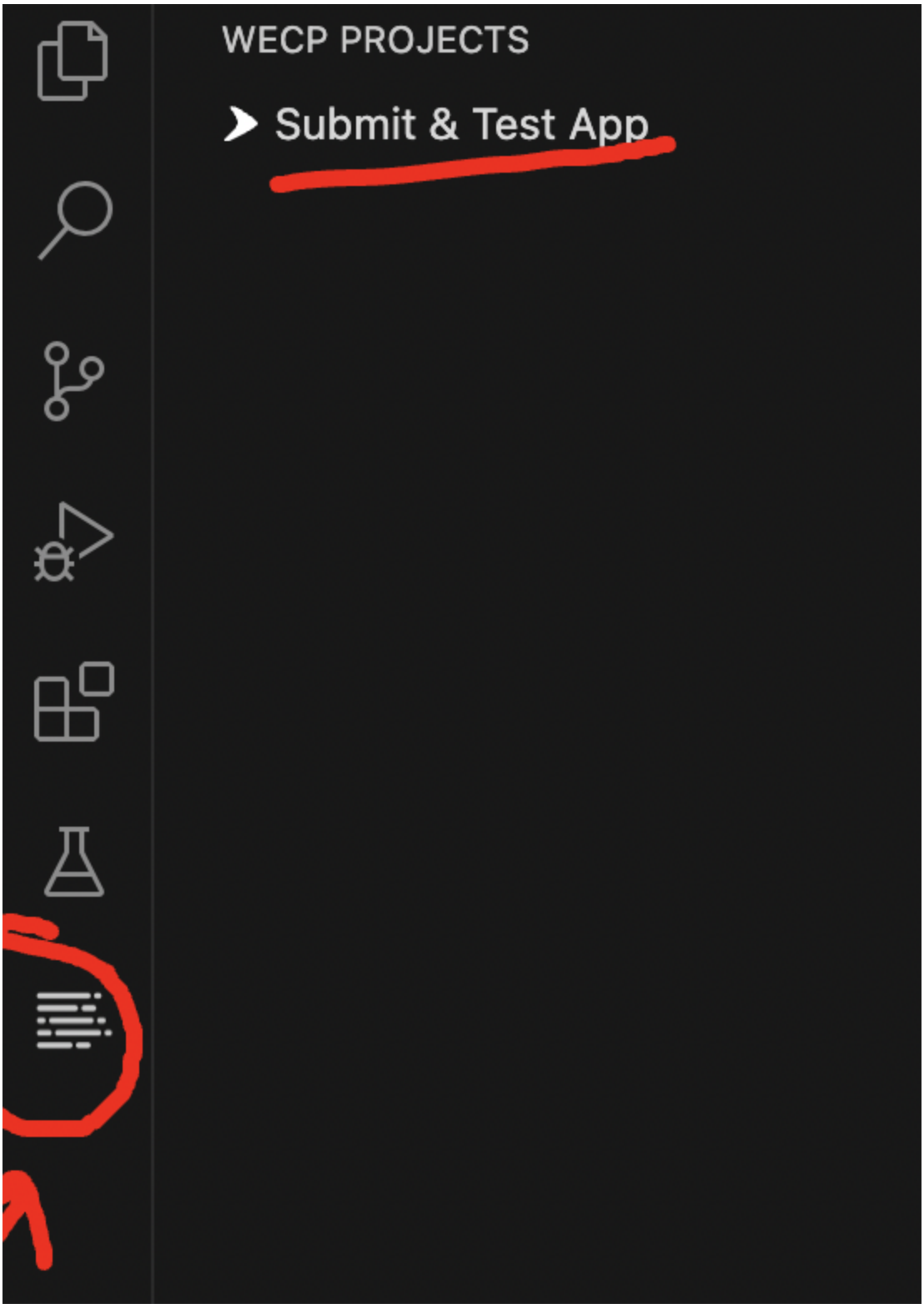
- src/main/java/com/wecp/library/repository/BookRepository.java
- src/main/java/com/wecp/library/repository/UserRepository.java

Notes:

- Implementation related specifics are added directly as javadocs in the workspace.
- Ensure that the structure and datatypes of the APIs are followed as specified in the comments to ensure that the code is evaluated correctly.

Testing & Submitting your code:

Step 1: Click on the **WeCP Projects** Button shown below.



Step 2: Write your code to complete the task(s) according to the question. Click on the **Submit & Test App** button shown above to execute your code and confirm if the application is working as expected.

Step 3: Click on the **Submit & Test App** button to execute your code. You will be given a congratulations message as shown below if your code is working perfectly.

Answer:

```
package com.wecp.library.repository;

import com.wecp.library.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface UserRepository extends JpaRepository<User, Long> {
}
```

```
package com.wecp.library.repository;

import com.wecp.library.domain.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
public interface BookRepository extends JpaRepository<Book,Long> {
}
```

```
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String username;
```

```
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

```
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class Book implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }
```

```
    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
package com.wecp.library.controller;

import com.wecp.library.domain.Book;
import com.wecp.library.domain.User;
import com.wecp.library.repository.BookRepository;
import com.wecp.library.repository.UserRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.EmptyResultDataAccessException;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

/**
 * REST controller for managing library system process
 */
@RestController
@RequestMapping("/api/v1")
public class LibraryController {

@Autowired
    private UserRepository ur;
@Autowired
    private BookRepository br;

    /**
     * {@code GET  /user/:id} : get the "id" User.
     *
     * @param id the id of the user to retrieve.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     * the user, or if does not exist, return with status "noContent".
```

```
     */
    @GetMapping("/user/{id}")
    public ResponseEntity<User> getUser(@PathVariable Long id) {
        User user=null;
        Optional<User> temp=ur.findById(id);
        if(temp.isPresent()){
            user=temp.get();
        }
        return ResponseEntity.ok().body(user);
    }

    /**
     * {@code POST  /user} : Create a new user.
     *
     * @param user the user to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the new user
     */
    @PostMapping("/user")
    public ResponseEntity<User> createUser(@RequestBody User user) {
        ur.save(user);
        return ResponseEntity.ok().body(user);
    }

    /**
     * {@code GET  /book/:id} : get the "id" Book.
     *
     * @param id the id of the book to retrieve.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     * the book, or if does not exist, return with status "noContent".
     */
    @GetMapping("/book/{id}")
    public ResponseEntity<Book> getBook(@PathVariable Long id) {
        Book book=null;
        Optional<Book> temp=br.findById(id);
        if(temp.isPresent()){
            book=temp.get();
        }
        return ResponseEntity.ok().body(book);
    }

    /**
     * {@code POST  /book} : Create a new book.
     *
```

```
     * @param book the book to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the new book
     */
    @PostMapping("/book")
    public ResponseEntity<Book> createBook(@RequestBody Book book) {
        br.save(book);
        return ResponseEntity.ok().body(book);
    }
}
```

2)

Build a Library Management Application in Spring Boot (Intermediate Level)

Project description:

The government is building a centralized library management system which involves a micro-service, allowing people to issue books and return them at the right time.

In this task, you need to conform to the following requirements:

1. Build **"depositBook"** API in a reactive way(should return Mono).
2. Users can issue a book and deposit them. If they issue the book after the given period, fine shoul d apply.
3. You can find the fields and do the math(issueDate, period, returnDate). This micro-service should be built using the Spring Boot REST API framework and connect to the database using the JPA API.
4. "depositBook" API should be built using the Spring Boot WebFlux.

The application exposes 2 more APIs. The details of the APIs are given below:

1. /api/v1/issueBook [POST]: Users issue a book of their choice, if book quantity is zero, it should t hrow the exception **"BookNotAvailableException".**

2. /api/v1/depositBook [POST]: Send issue, return fine if there is one or return 0 (this one should be reactive)

Your task is to complete the file:

- src/main/java/com/wecp/library/controller/LibraryController.java
- src/main/java/com/wecp/library/repository/BookRepository.java
- src/main/java/com/wecp/library/repository/IssueRepository.java

You can find blank jpa repositories here:

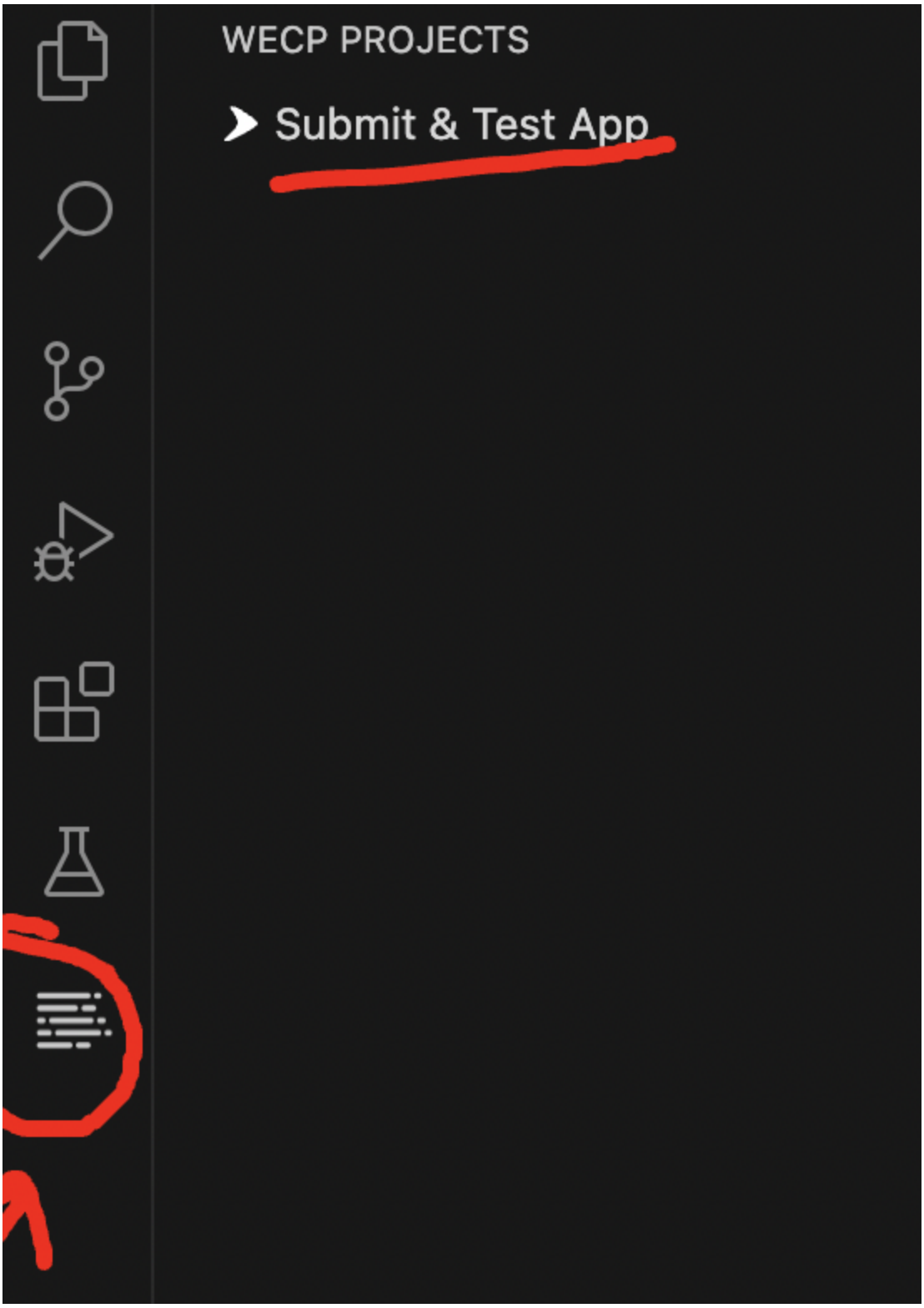
- src/main/java/com/wecp/library/repository/BookRepository.java
- src/main/java/com/wecp/library/repository/IssueRepository.java

Notes:

1. Implementation related specifics are added directly as javadocs in the workspace.
2. Ensure that the structure and
   datatypes of the APIs are followed as specified in the comments to ensure that the code is evalua ted correctly.

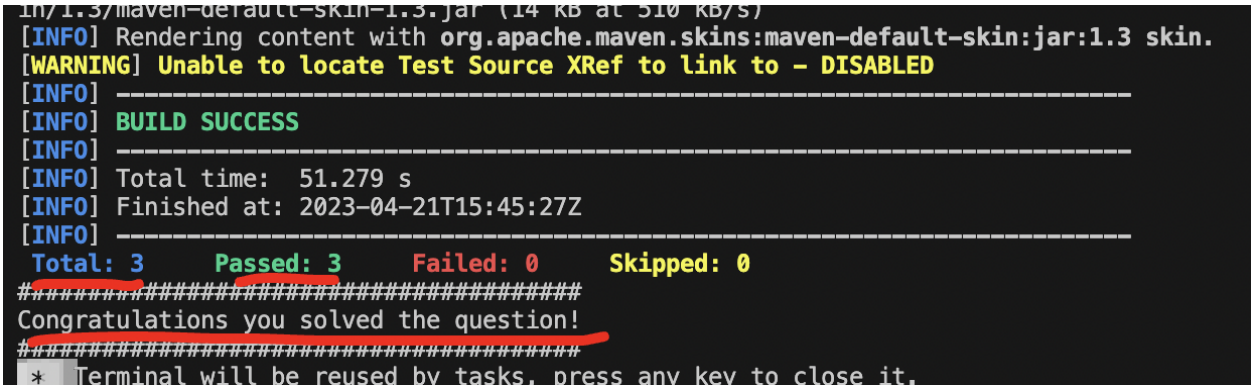
Testing & Submitting your code:

Step 1: Click on the **WeCP Projects** Button shown below.



Step 2: Write your code to complete the task(s) according to the question. Click on the **Submit & Test App** button shown above to execute your code and confirm if the application is working as expected.

Step 3: Click on the **Submit & Test App** button to execute your code. You will be given a congratulations message as shown below if your code is working perfectly.



Answer:

```java
package com.wecp.library.repository;

import com.wecp.library.domain.Issue;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 * Spring Data repository for the Issue entity. Use correct annotation and extend the class with necessary Spring Data class
 */
@Repository
public interface IssueRepository extends JpaRepository<Issue,Long> {
}
```

```java
package com.wecp.library.repository;

import com.wecp.library.domain.Book;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;

/**
 * Spring Data repository for the Book entity. Use correct annotation and extend the class with necessary Spring Data class
 */
@Repository
```

```
public interface BookRepository extends JpaRepository<Book,Long> {
}
```

```
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String username;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

```
package com.wecp.library.domain;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import reactor.core.publisher.Mono;
```

```java
import javax.persistence.*;
import java.io.Serializable;
import java.time.LocalDate;

@Entity
public class Issue implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private LocalDate issueDate;

    private LocalDate returnDate;

    private Integer period;

    private Integer fine;

    @ManyToOne
    @JsonIgnoreProperties(value = "issues", allowSetters = true)
    private Book book;

    private Long userId;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public LocalDate getIssueDate() {
        return issueDate;
    }

    public void setIssueDate(LocalDate issueDate) {
        this.issueDate = issueDate;
    }

    public LocalDate getReturnDate() {
        return returnDate;
```

```
    }

    public void setReturnDate(LocalDate returnDate) {
        this.returnDate = returnDate;
    }

    public Integer getPeriod() {
        return period;
    }

    public void setPeriod(Integer period) {
        this.period = period;
    }

    public Integer getFine() {
        return fine;
    }

    public void setFine(Integer fine) {
        this.fine = fine;
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    public Long getUserId() {
        return userId;
    }

    public void setUserId(Long userId) {
        this.userId = userId;
    }
}
```

```
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
```

```java
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class Book implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    private Integer quantity;

    public Long getId() {
        return id;
    }

    public Integer getQuantity() {
        return quantity;
    }

    public void setQuantity(Integer quantity) {
        this.quantity = quantity;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```java
package com.wecp.library.controller.exception;
```

```java
import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.NOT_FOUND)
public class BookNotAvailableException extends RuntimeException {

    }


package com.wecp.library.controller;

import com.wecp.library.controller.exception.BookNotAvailableException;
import com.wecp.library.domain.Book;
import com.wecp.library.domain.Issue;
import com.wecp.library.repository.BookRepository;
import com.wecp.library.repository.IssueRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import reactor.core.publisher.Mono;

import java.time.temporal.ChronoUnit;
import java.util.Optional;

/**
 * REST controller for managing library system process
 */
@RestController
@RequestMapping("/api/v1")
public class LibraryController {
@Autowired
    private BookRepository bookRepository;
@Autowired
    private IssueRepository issueRepository;


    /**
     * {@code POST  /issueBook} : Create a new issue.
     *
     * @param issue the issue to create.
```

```
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the new issue.
     * throw {@link BookNotAvailableException} if the wanted books quantity is
zero
     */
    @PostMapping("/issueBook")
    public ResponseEntity<Issue> issueBook(@RequestBody Issue issue) {
        Book book=issue.getBook();
        if(book.getQuantity()==0){
            throw new BookNotAvailableException();
        }else{
            issueRepository.save(issue);
        }
        return ResponseEntity.ok().body(issue);
    }

    /**
     * {@code POST  /depositBook} : Inquiry the issue if there is a fine
     *
     * @param issue the issue to inquiry.
     * @return the Mono with fine or just return 0
     */
    @PostMapping("/depositBook")
    public Mono<Integer> depositBook(@RequestBody Issue issue) {
        Book book=issue.getBook();
        Long day=Math.abs(ChronoUnit.DAYS.between(issue.getIssueDate(),
issue.getReturnDate()));
        if(day>issue.getPeriod()){
            return Mono.just(issue.getFine());
        }
        return Mono.just(0);
    }
}
```

```
package com.wecp.library.config;

import com.fasterxml.jackson.datatype.jdk8.Jdk8Module;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class JacksonConfiguration {
```

```
    /**
     * Support for Java date and time API.
     *
     * @return the corresponding Jackson module.
     */
    @Bean
    public JavaTimeModule javaTimeModule() {
        return new JavaTimeModule();
    }

    @Bean
    public Jdk8Module jdk8TimeModule() {
        return new Jdk8Module();
    }

}
```

3) Build a Library Management Application in Spring Boot (Advanced Level)

Previous
Next

Project description:

The government is building a centralized library management system which involves a micro-service, allowing people to issue books and return them at the right time.

Library management requires creating users and renewing user subscriptions via authenticated processes. Secure create-user and renew-user-subscription APIs with Spring Security. The API issue-book should be permitted for everyone. This micro-service should be built using the Spring Boot REST API framework and it should connect to the database using the JPA API and secure methods using Spring Security.

In this task, implement three APIs for which the details are given below:

- `/api/v1/create-user [POST]`: Simple user save method in an authenticated manner

- /api/v1/issue-book [POST]: Send issue, check if users subscribed(see subscribed field in User entity), otherwise throw SubscriptionExpiredException- permitAll
- /api/v1/renew-user-subscription/{id} [GET]: Send userId, set user subscription to true in an authenticated manner.

Your task is to complete the files given below:

1. src/main/java/com/wecp/library/controller/LibraryController.java
2. src/main/java/com/wecp/library/security/WebSecurityConfigurer.java
3. src/main/java/com/wecp/library/repository/UserRepository.java
4. src/main/java/com/wecp/library/repository/IssueRepository.java

You can find blank jpa repositories here:

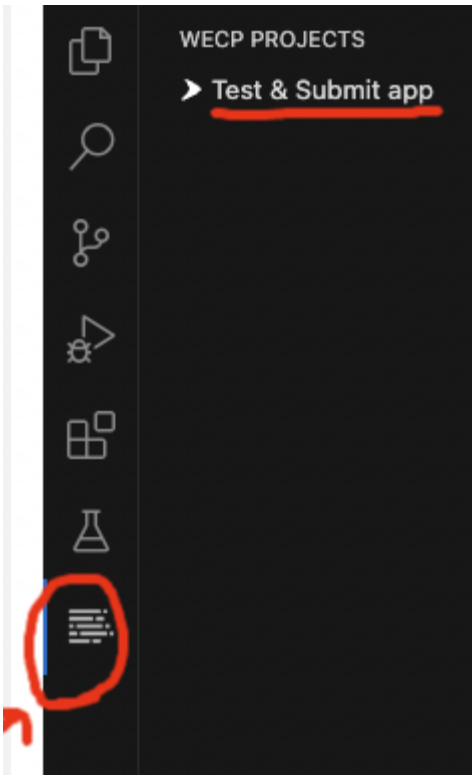
1. src/main/java/com/wecp/library/repository/UserRepository.java
2. src/main/java/com/wecp/library/repository/IssueRepository.java

Notes:

- Implementation related specifics are added directly as javadocs in the workspace.
- Ensure that the structure and datatypes of the APIs are followed as specified in the comments to ensure that the code is evaluated correctly.

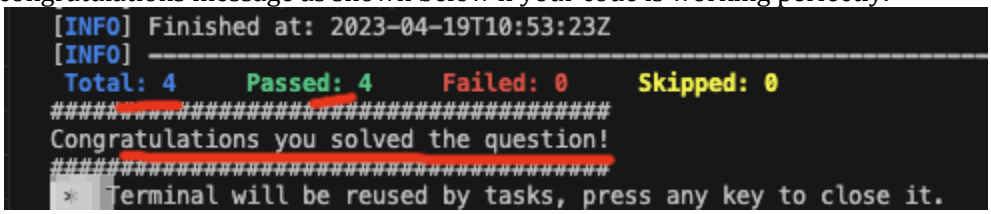
Testing & Submitting your code:

Step 1: Click on the **WeCP Projects** Button shown below.



Step 2: Write your code to complete the task(s) according to the question. Click on the **Test and Submit App** shown above to execute your code and confirm if the application is working as expected.

Step 3: Click on the **Test and Submit App** button to execute your code. You will be given a congratulations message as shown below if your code is working perfectly.



Step 4: You will be given a string url on clicking on **Show testing url button**. Append your API endpoints to the end of this string url to test your endpoints on thunderclient.

Step 5: Use the **Run App** button to start the application before performing api testing.

Answer:

```
package com.wecp.library.repository;

import com.wecp.library.domain.Issue;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
```

```java
@Repository
public interface IssueRepository extends JpaRepository<Issue,Long>{
}
```

```java
package com.wecp.library.repository;

import com.wecp.library.domain.User;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.stereotype.Repository;
@Repository
 public interface UserRepository extends JpaRepository<User,Long>{

 }
```

```java
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class User implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String username;

    private boolean subscribed = false;

    public boolean getSubscribed() {
        return subscribed;
    }

    public void setSubscribed(boolean subscribed) {
        this.subscribed = subscribed;
    }
```

```
    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getUsername() {
        return username;
    }

    public void setUsername(String username) {
        this.username = username;
    }
}
```

```
package com.wecp.library.domain;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;
import reactor.core.publisher.Mono;

import javax.persistence.*;
import java.io.Serializable;
import java.time.LocalDate;

@Entity
public class Issue implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private LocalDate issueDate;

    private LocalDate returnDate;

    private Integer period;

    private Integer fine;

    @ManyToOne
```

```java
    @JsonIgnoreProperties(value = "issues", allowSetters = true)
    private Book book;

    @ManyToOne
    @JsonIgnoreProperties(value = "issues", allowSetters = true)
    private User user;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public LocalDate getIssueDate() {
        return issueDate;
    }

    public void setIssueDate(LocalDate issueDate) {
        this.issueDate = issueDate;
    }

    public LocalDate getReturnDate() {
        return returnDate;
    }

    public void setReturnDate(LocalDate returnDate) {
        this.returnDate = returnDate;
    }

    public Integer getPeriod() {
        return period;
    }

    public void setPeriod(Integer period) {
        this.period = period;
    }

    public Integer getFine() {
        return fine;
    }

    public void setFine(Integer fine) {
        this.fine = fine;
```

```
    }

    public Book getBook() {
        return book;
    }

    public void setBook(Book book) {
        this.book = book;
    }

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

```
package com.wecp.library.domain;

import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.GenerationType;
import javax.persistence.Id;
import java.io.Serializable;

@Entity
public class Book implements Serializable {
    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;

    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
```

```
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
}
```

```
package com.wecp.library.controller.exception;

import org.springframework.http.HttpStatus;
import org.springframework.web.bind.annotation.ResponseStatus;

@ResponseStatus(HttpStatus.FORBIDDEN)
public class UserNotSubscribedException extends RuntimeException {
public UserNotSubscribedException(String message){
    super(message);
}
}
```

```
package com.wecp.library.controller;

import com.wecp.library.controller.exception.UserNotSubscribedException;
import com.wecp.library.domain.Issue;
import com.wecp.library.domain.User;
import com.wecp.library.repository.IssueRepository;
import com.wecp.library.repository.UserRepository;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.*;

import java.util.Optional;

/**
 * REST controller for managing library system process
 */
@RestController
@RequestMapping("/api/v1")
```

```java
public class LibraryController {
    @Autowired
private UserRepository ur;
@Autowired
private IssueRepository ir;
    /**
     * {@code POST  /issueBook} : Create a new issue.
     *
     * @param issue the issue to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     * the user, or if does not exist, return with status "noContent".
     * If user is not subscribed, throw {@link UserNotSubscribedException}
     */
    @PostMapping("/issue-book")
    public ResponseEntity<Issue> issueBook(@RequestBody Issue issue) {
        Long id=issue.getUser().getId();
        Optional<User> temp=ur.findById(id);
        if(temp.isEmpty()){
            ResponseEntity.noContent().build();
        }
        User user=temp.get();
        if(!user.getSubscribed()){
            throw new UserNotSubscribedException("oiutyr");
        }
        return ResponseEntity.ok(ir.save(issue));
    }

    /**
     * {@code POST  /user} : Create a new user.
     *
     * @param user the user to create.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body the new user
     */
    @PostMapping("/user")
    public ResponseEntity<User> createUser(@RequestBody User user) {

        return ResponseEntity.ok(ur.save(user));
    }

    /**
     * {@code GET  /renew-user-subscription/:id} :  Send userId, set user
subscription to true
     *
```

```
     * @param id the id of the user to renew subscription.
     * @return the {@link ResponseEntity} with status {@code 200 (OK)} and with
body
     * the user, or if does not exist, return with status "noContent".
     */
    @GetMapping("renew-user-subscription/{id}")
    public ResponseEntity<User> renewUserSubscription(@PathVariable Long id) {
        Optional<User> temp=ur.findById(id);
        if(temp.isEmpty()){
            return ResponseEntity.noContent().build();
        }
        User user=temp.get();
        user.setSubscribed(true);
        return ResponseEntity.ok(ur.save(user));
    }
}
```

```
package com.wecp.library.config;

import com.fasterxml.jackson.datatype.jdk8.Jdk8Module;
import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

@Configuration
public class JacksonConfiguration {

    /**
     * Support for Java date and time API.
     *
     * @return the corresponding Jackson module.
     */
    @Bean
    public JavaTimeModule javaTimeModule() {
        return new JavaTimeModule();
    }

    @Bean
    public Jdk8Module jdk8TimeModule() {
        return new Jdk8Module();
    }

}
```

```
package com.wecp.library.security;

import org.springframework.context.annotation.Configuration;
import org.springframework.http.HttpStatus;
import org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.authentication.builders.Authentica
tionManagerBuilder;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebSecurit
y;
import
org.springframework.security.config.annotation.web.configuration.WebSecurityConfi
gurerAdapter;
import org.springframework.security.web.authentication.HttpStatusEntryPoint;

/**
 * Configure Spring Security class here. Don't forget to extend the class with
the necessary Spring Security class.
 * user and renew-user-subscription APIs must be authenticated and issue-book
must be permitAll.
 */
@Configuration
@EnableWebSecurity
public class WebSecurityConfigurer extends WebSecurityConfigurerAdapter {

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception
{
        // TODO Auto-generated method stub
        auth.authenticationProvider(new DaoAuthenticationProvider());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        // TODO Auto-generated method stub
        http.authorizeRequests().antMatchers("/api/v1/issue-book").
        permitAll().anyRequest().authenticated().and().exceptionHandling().
        authenticationEntryPoint(new
HttpStatusEntryPoint(HttpStatus.UNAUTHORIZED));
```

```
}
}
```