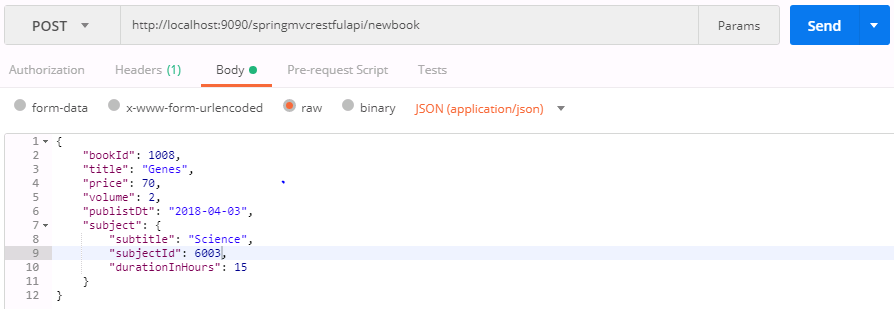
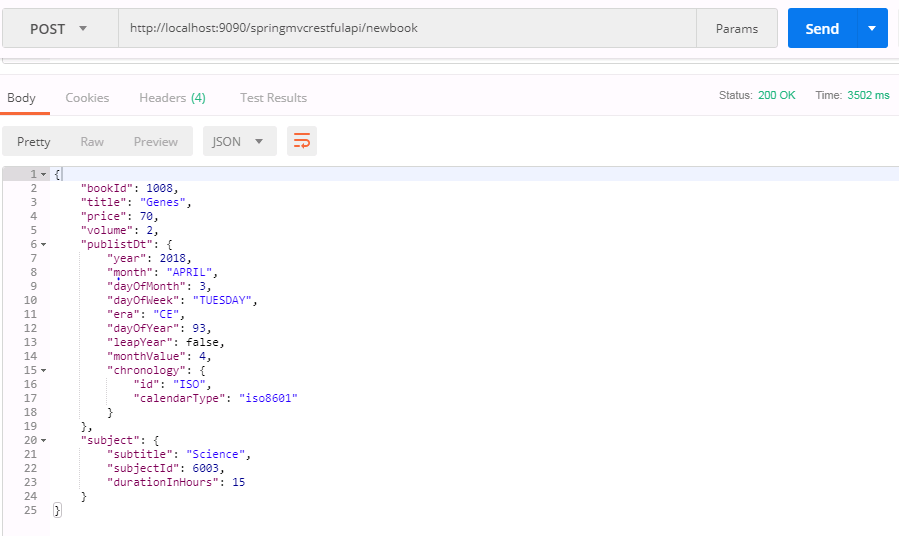
**RESTFULAPI ASSIGNMENT 2**

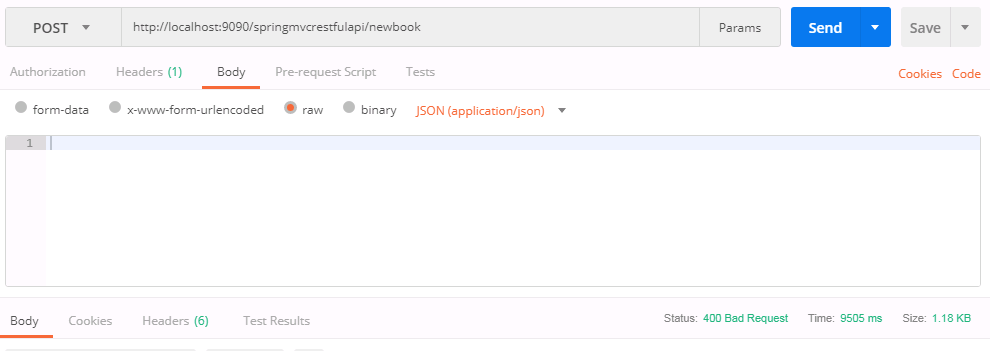
**Results:**

**1.Is a new book added?**

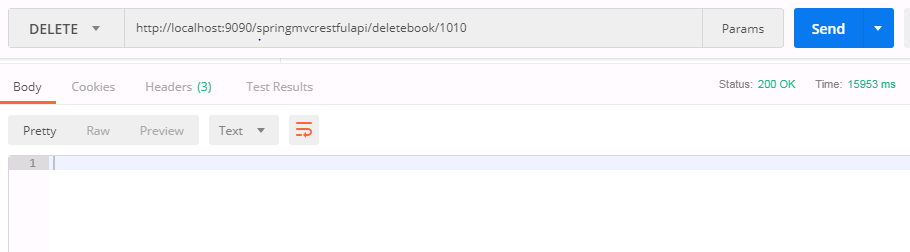
****

****

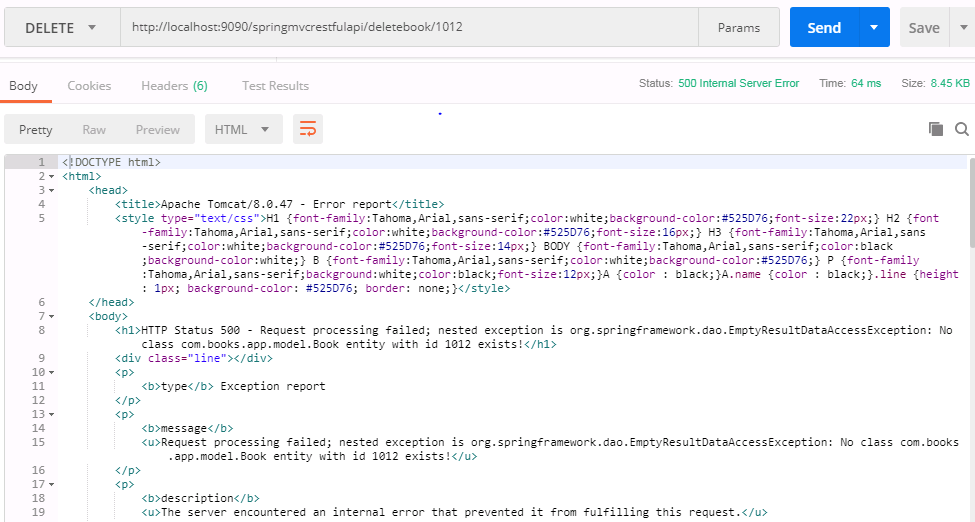
**2.What if the book is null for add book service?**

****

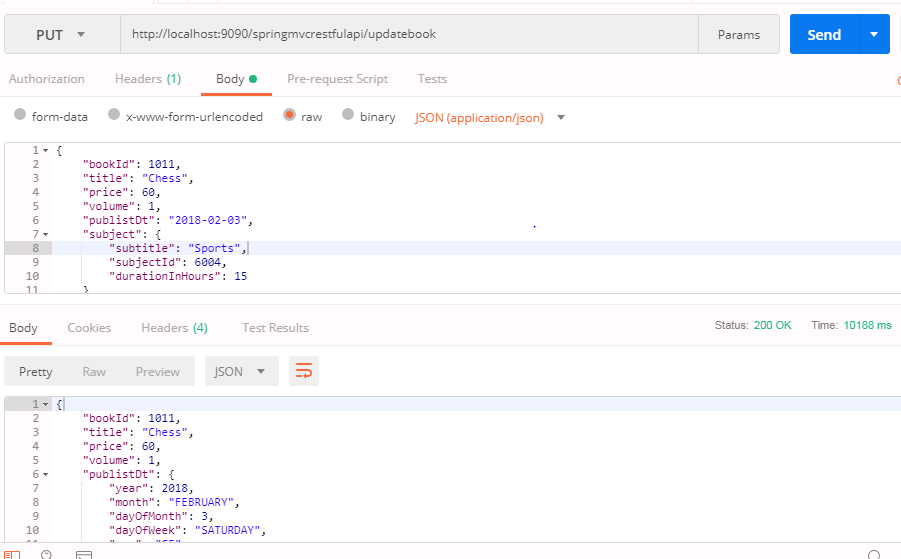
**3.Delete book operation with existing id.**

****

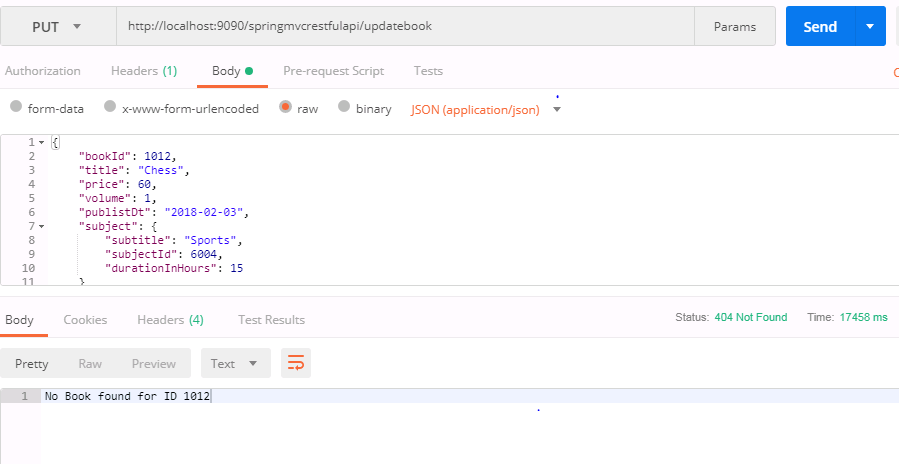
**4.Delete book operation with non-existing id**

****

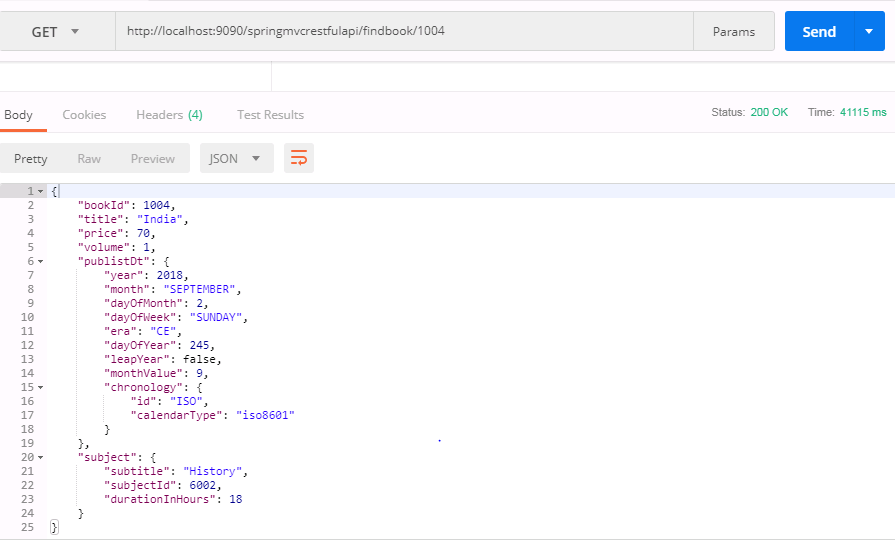
**5.Update book with existing id**

****

**6.Update book operation with non existing id.**

****

**7.Get a book operation with existing id**

****

**8.Get a book operation with non existing id.**

****

**Code:**

**web.xml**

<web-app xmlns=*"http://java.sun.com/xml/ns/javaee"*

version=*"2.5"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd"*>

<display-name>Spring MVC Rest API Application</display-name>

<servlet>

<servlet-name>dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/dispatcher-servlet.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

<listener>

<listener-class>

org.springframework.web.context.ContextLoaderListener

</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring.xml</param-value>

</context-param>

</web-app>

**spring.xml**

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:tx=*"http://www.springframework.org/schema/tx"* xmlns:p=*"http://www.springframework.org/schema/p"*

xmlns:jpa=*"http://www.springframework.org/schema/data/jpa"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context-3.0.xsd*

*http://www.springframework.org/schema/tx*

*http://www.springframework.org/schema/tx/spring-tx.xsd*

*http://www.springframework.org/schema/data/jpa*

*http://www.springframework.org/schema/data/jpa/spring-jpa-1.8.xsd"*>

<context:component-scan base-package=*"com.books.app"* />

<bean id=*"bookAppService"* class=*"com.books.app.service.BookAppServiceImpl"*></bean>

<context:property-placeholder location=*"classpath:database.properties"* />

<bean id=*"basicDataSource"* class=*"org.apache.commons.dbcp2.BasicDataSource"*

destroy-method=*"close"*>

<property name=*"driverClassName"* value=*"${database.driverClassName}"* />

<property name=*"url"* value=*"${database.url}"* />

<property name=*"username"* value=*"${database.username}"* />

<property name=*"password"* value=*"${database.password}"* />

</bean>

<bean name=*"hibernateJpaVendorAdapter"*

class=*"org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter"* />

<bean id=*"entityManagerFactory"*

class=*"org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean"*>

<property name=*"jpaVendorAdapter"* ref=*"hibernateJpaVendorAdapter"* />

<property name=*"dataSource"* ref=*"basicDataSource"* />

<property name=*"persistenceUnitName"* value=*"myJpaPersistenceUnit"* />

<property name=*"packagesToScan"*>

<list>

<value>com.books.app.model</value>

</list>

</property>

<property name=*"jpaProperties"*>

<props>

<prop key=*"hibernate.dialect"*>${hibernate.dialect}</prop>

<prop key=*"hibernate.show\_sql"*>${hibernate.show\_sql}</prop>

<prop key=*"hibernate.format\_sql"*>${hibernate.format\_sql}</prop>

<prop key=*"hibernate.id.new\_generator\_mappings"*>${hibernate.id.new\_generator\_mappings}</prop>

</props>

</property>

</bean>

<bean id=*"transactionManager"* class=*"org.springframework.orm.jpa.JpaTransactionManager"*>

<property name=*"entityManagerFactory"* ref=*"entityManagerFactory"* />

</bean>

<tx:annotation-driven transaction-manager=*"transactionManager"* />

<jpa:repositories base-package=*"com.books.app.dao"*

entity-manager-factory-ref=*"entityManagerFactory"* />

</beans>

**dispatcher-servlet.xml**

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:mvc=*"http://www.springframework.org/schema/mvc"* xmlns:context=*"http://www.springframework.org/schema/context"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*" http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-3.0.xsd http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.0.xsd"*>

<mvc:annotation-driven />

<context:component-scan base-package=*"com.books.app"* />

<mvc:resources location=*"/resources/"* mapping=*"/resources/\*\*"*></mvc:resources>

</beans>

**database.properties**

database.driverClassName=com.mysql.jdbc.Driver

database.url=jdbc:mysql://localhost:3306/bookshop

database.username=root

database.password=root

hibernate.dialect = org.hibernate.dialect.MySQLDialect

hibernate.show\_sql = true

hibernate.format\_sql = true

hibernate.id.new\_generator\_mappings = false

**pom.xml**

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-v4\_0\_0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>books.app</groupId>

<artifactId>springmvcrestfulapi</artifactId>

<packaging>war</packaging>

<version>0.0.1-SNAPSHOT</version>

<name>springmvcrestfulapi Maven Webapp</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<spring.version>5.0.7.RELEASE</spring.version>

<spring.data>2.0.1.RELEASE</spring.data>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-core</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework.data</groupId>

<artifactId>spring-data-jpa</artifactId>

<version>${spring.data}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>jstl</artifactId>

<version>1.2</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.44</version>

</dependency> <!-- Hibernate 5.2.6 Final -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.2.6.Final</version>

</dependency>

<dependency>

<groupId>org.apache.commons</groupId>

<artifactId>commons-dbcp2</artifactId>

<version>2.1.1</version>

</dependency>

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.0</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

<dependency>

<groupId>commons-logging</groupId>

<artifactId>commons-logging</artifactId>

<version>1.2</version>

</dependency>

</dependencies>

<build>

<finalName>springmvcrestfulapi</finalName>

</build>

</project>

**BookController.java**

**package** com.books.app.controller;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.http.HttpStatus;

**import** org.springframework.http.MediaType;

**import** org.springframework.http.ResponseEntity;

**import** org.springframework.web.bind.annotation.DeleteMapping;

**import** org.springframework.web.bind.annotation.GetMapping;

**import** org.springframework.web.bind.annotation.PathVariable;

**import** org.springframework.web.bind.annotation.PostMapping;

**import** org.springframework.web.bind.annotation.PutMapping;

**import** org.springframework.web.bind.annotation.RequestBody;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestMethod;

**import** org.springframework.web.bind.annotation.RestController;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.service.IBookAppService;

@RestController

**public** **class** BookController {

@Autowired

IBookAppService bookAppService;

@RequestMapping(value = "/allbooks", method = RequestMethod.***GET***, produces = MediaType.***APPLICATION\_JSON\_VALUE***)

**public** ResponseEntity<List<Book>> viewAllBooks() **throws** BookAppException {

List<Book> books = bookAppService.view();

**return** **new** ResponseEntity<List<Book>>(books, HttpStatus.***OK***);

}

@GetMapping("/findbook/{bookId}")

**public** ResponseEntity<Book> getBookById(@PathVariable(value = "bookId") **int** bookId) **throws** BookAppException {

Book book = bookAppService.search(bookId);

**if** (book == **null**) {

**return** **new** ResponseEntity("No Book found for ID " + bookId, HttpStatus.***NOT\_FOUND***);

}

**return** **new** ResponseEntity<Book>(book, HttpStatus.***OK***);

}

@PostMapping("/newbook")

**public** ResponseEntity<Book> addNewBook(@RequestBody Book book) **throws** BookAppException {

**if** (book == **null**) {

**return** **new** ResponseEntity("Book data required ", HttpStatus.***BAD\_REQUEST***);

}

**return** **new** ResponseEntity<Book>(bookAppService.add(book, book.getSubject().getSubtitle()), HttpStatus.***OK***);

}

@PutMapping("/updatebook")

**public** ResponseEntity<Book> editBook(@RequestBody Book book) **throws** BookAppException {

Book sbook = bookAppService.edit(book);

**if** (sbook == **null**) {

**return** **new** ResponseEntity("No Book found for ID " + book.getBookId(), HttpStatus.***NOT\_FOUND***);

}

**return** **new** ResponseEntity<Book>(sbook, HttpStatus.***OK***);

}

@DeleteMapping("/deletebook/{bookId}")

**public** ResponseEntity<Void> deleteBook(@PathVariable(value = "bookId") **int** bookId) **throws** BookAppException {

ResponseEntity<Void> response = **null**;

bookAppService.delete(bookId);

response = **new** ResponseEntity<>(HttpStatus.***OK***);

**return** response;

}

}

**IBookAppService.java**

**package** com.books.app.service;

**import** java.util.List;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**public** **interface** IBookAppService {

Book add(Book book, String subtitle) **throws** BookAppException;

Book edit(Book book) **throws** BookAppException;

**void** delete(**int** bookid) **throws** BookAppException;

Book search(**int** bookid) **throws** BookAppException;

List<Book> view() **throws** BookAppException;

}

**BookAppServiceImpl.java**

**package** com.books.app.service;

**import** java.util.List;

**import** java.util.Optional;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.books.app.dao.IBookRepository;

**import** com.books.app.dao.ISubjectRepository;

**import** com.books.app.exception.BookAppException;

**import** com.books.app.model.Book;

**import** com.books.app.model.Subject;

@Service

**public** **class** BookAppServiceImpl **implements** IBookAppService {

@Autowired

**private** ISubjectRepository subjRepo;

@Autowired

**private** IBookRepository bookRepo;

@Transactional

@Override

**public** Book add(Book book, String subtitle) **throws** BookAppException {

Subject subject = **null**;

**if** (book.getSubject() != **null**) {

List<Subject> subjectList = subjRepo.findAll();

**for** (Subject subj : subjectList) {

**if** (subj.getSubtitle() != **null** && subj.getSubtitle().equalsIgnoreCase(subtitle)) {

subject = subj;

}

}

**if** (subject == **null**) {

subject = **new** Subject();

subject.setSubtitle(subtitle);

subjRepo.save(subject);

}

}

book.setSubject(subject);

Book sbook = bookRepo.save(book);

**return** sbook;

}

@Transactional

@Override

**public** Book edit(Book book) **throws** BookAppException {

String subtitle = book.getSubject().getSubtitle();

Subject subject = **null**;

**if** (book.getSubject() != **null**) {

List<Subject> subjectList = subjRepo.findAll();

**for** (Subject subj : subjectList) {

**if** (subj.getSubtitle() != **null** && subj.getSubtitle().equalsIgnoreCase(subtitle)) {

subject = subj;

}

}

**if** (subject == **null**) {

subject = **new** Subject();

subject.setSubtitle(subtitle);

subjRepo.save(subject);

}

}

Book fbook = search(book.getBookId());

**if** (fbook != **null**) {

fbook.setBookId(book.getBookId());

fbook.setTitle(book.getTitle());

fbook.setPrice(book.getPrice());

fbook.setVolume(book.getVolume());

fbook.setPublistDt(book.getPublistDt());

}**else** {

**return** **null**;

}

fbook.setSubject(subject);

**return** bookRepo.save(fbook);

}

@Transactional

@Override

**public** **void** delete(**int** bookid) **throws** BookAppException {

bookRepo.deleteById(bookid);

}

@Override

**public** Book search(**int** bookid) **throws** BookAppException {

Optional<Book> optBook = bookRepo.findById(bookid);

Book book = optBook.isPresent() ? optBook.get() : **null**;

**return** book;

}

@Override

**public** List<Book> view() **throws** BookAppException{

**return** bookRepo.findAll();

}

}

**IBookRepository.java**

**package** com.books.app.dao;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.books.app.model.Book;

**public** **interface** IBookRepository **extends** JpaRepository<Book,Integer>{

}

**ISubjectepository.java**

**package** com.books.app.dao;

**import** org.springframework.data.jpa.repository.JpaRepository;

**import** com.books.app.model.Subject;

**public** **interface** ISubjectRepository **extends** JpaRepository<Subject,String>{

}

**BookAppException.java**

**package** com.books.app.exception;

@SuppressWarnings("serial")

**public** **class** BookAppException **extends** Exception {

**public** BookAppException(String errMsg) {

**super**(errMsg);

}

}

**Subject.java**

**package** com.books.app.model;

**import** java.util.Set;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.OneToMany;

**import** javax.persistence.Table;

**import** com.fasterxml.jackson.annotation.JsonIgnore;

@Entity

@Table(name="subjects")

**public** **class** Subject {

@Id

@Column(name="subtitle")

**private** String subtitle;

@Column(name="subjectid")

**private** **long** subjectId;

@Column(name="durationInHours")

**private** **int** durationInHours;

@JsonIgnore

@OneToMany(mappedBy="subject",orphanRemoval=**true**)

**private** Set<Book> references;

**public** **long** getSubjectId() {

**return** subjectId;

}

**public** **void** setSubjectId(**long** subjectId) {

**this**.subjectId = subjectId;

}

**public** String getSubtitle() {

**return** subtitle;

}

**public** **void** setSubtitle(String subtitle) {

**this**.subtitle = subtitle;

}

**public** **int** getDurationInHours() {

**return** durationInHours;

}

**public** **void** setDurationInHours(**int** durationInHours) {

**this**.durationInHours = durationInHours;

}

**public** Set<Book> getReferences() {

**return** references;

}

**public** **void** setReferences(Set<Book> references) {

**this**.references = references;

}

}

**Book.java**

**package** com.books.app.model;

**import** java.time.LocalDate;

**import** javax.persistence.Column;

**import** javax.persistence.Entity;

**import** javax.persistence.Id;

**import** javax.persistence.JoinColumn;

**import** javax.persistence.ManyToOne;

**import** javax.persistence.Table;

**import** org.springframework.format.annotation.DateTimeFormat;

**import** com.books.app.util.BookHelper;

**import** com.fasterxml.jackson.databind.annotation.JsonDeserialize;

@Entity

@Table(name="books")

**public** **class** Book {

@Id

@Column(name="bookid")

**private** **int** bookId;

@Column(name="title")

**private** String title;

@Column(name="price")

**private** **double** price;

@Column(name="vol")

**private** **int** volume;

@Column(name="pdate")

@DateTimeFormat(pattern ="dd-MM-yyyy")

@JsonDeserialize(using=BookHelper.**class**)

**private** LocalDate publistDt;

@ManyToOne

@JoinColumn(name="stitle" , nullable=**false**)

**private** Subject subject;

**public** Subject getSubject() {

**return** subject;

}

**public** **void** setSubject(Subject subject) {

**this**.subject = subject;

}

**public** **int** getBookId() {

**return** bookId;

}

**public** **void** setBookId(**int** bookId) {

**this**.bookId = bookId;

}

**public** String getTitle() {

**return** title;

}

**public** **void** setTitle(String title) {

**this**.title = title;

}

**public** **double** getPrice() {

**return** price;

}

**public** **void** setPrice(**double** price) {

**this**.price = price;

}

**public** **int** getVolume() {

**return** volume;

}

**public** **void** setVolume(**int** volume) {

**this**.volume = volume;

}

**public** LocalDate getPublistDt() {

**return** publistDt;

}

**public** **void** setPublistDt(LocalDate publistDt) {

**this**.publistDt = publistDt;

}

}

**BookHelper.java**

**package** com.books.app.util;

**import** java.io.IOException;

**import** java.time.LocalDate;

**import** java.time.format.DateTimeFormatter;

**import** com.fasterxml.jackson.core.JsonParser;

**import** com.fasterxml.jackson.core.JsonProcessingException;

**import** com.fasterxml.jackson.databind.DeserializationContext;

**import** com.fasterxml.jackson.databind.JsonDeserializer;

**public** **class** BookHelper **extends** JsonDeserializer<LocalDate>{

@Override

**public** LocalDate deserialize(JsonParser p, DeserializationContext ctxt)

**throws** IOException, JsonProcessingException {

DateTimeFormatter formatter = DateTimeFormatter.*ofPattern*("yyyy-MM-dd");

LocalDate localDate = **null**;

localDate = LocalDate.*parse*(p.getText(), formatter);

**return** localDate;

}

}