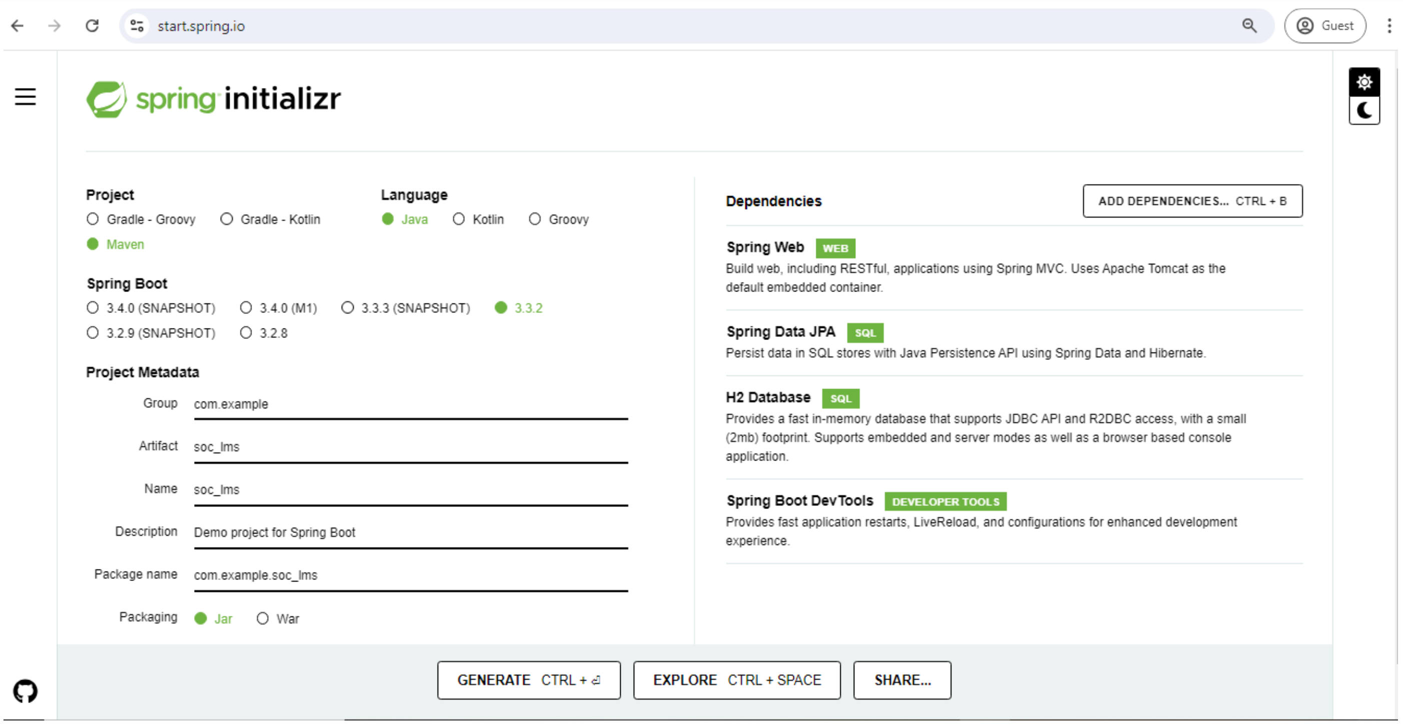
**Name:** Rakesh Chowdhury **BITS ID:** 2022MT70191 **Course Name:** Service Oriented Computing

**Topic Name:** Library management system
**Tool Used:** Eclipse, MySQL, Postman, Spring initializr
**Technology Used:** Java, Springboot, MySQL, JSON

## First we need to initialize our application using Spring Initializr.


start.spring.io

spring initializr

Project: Gradle - Groovy, Gradle - Kotlin, [✓ Maven]

Language: [✓ Java], Kotlin, Groovy

Spring Boot: 3.4.0 (SNAPSHOT), 3.4.0 (M1), 3.3.3 (SNAPSHOT), [✓ 3.3.2], 3.2.9 (SNAPSHOT), 3.2.8

Project Metadata

| | |
|---|---|
| Group | com.example |
| Artifact | soc_lms |
| Name | soc_lms |
| Description | Demo project for Spring Boot |
| Package name | com.example.soc_lms |
| Packaging | [✓ Jar] War |

Dependencies

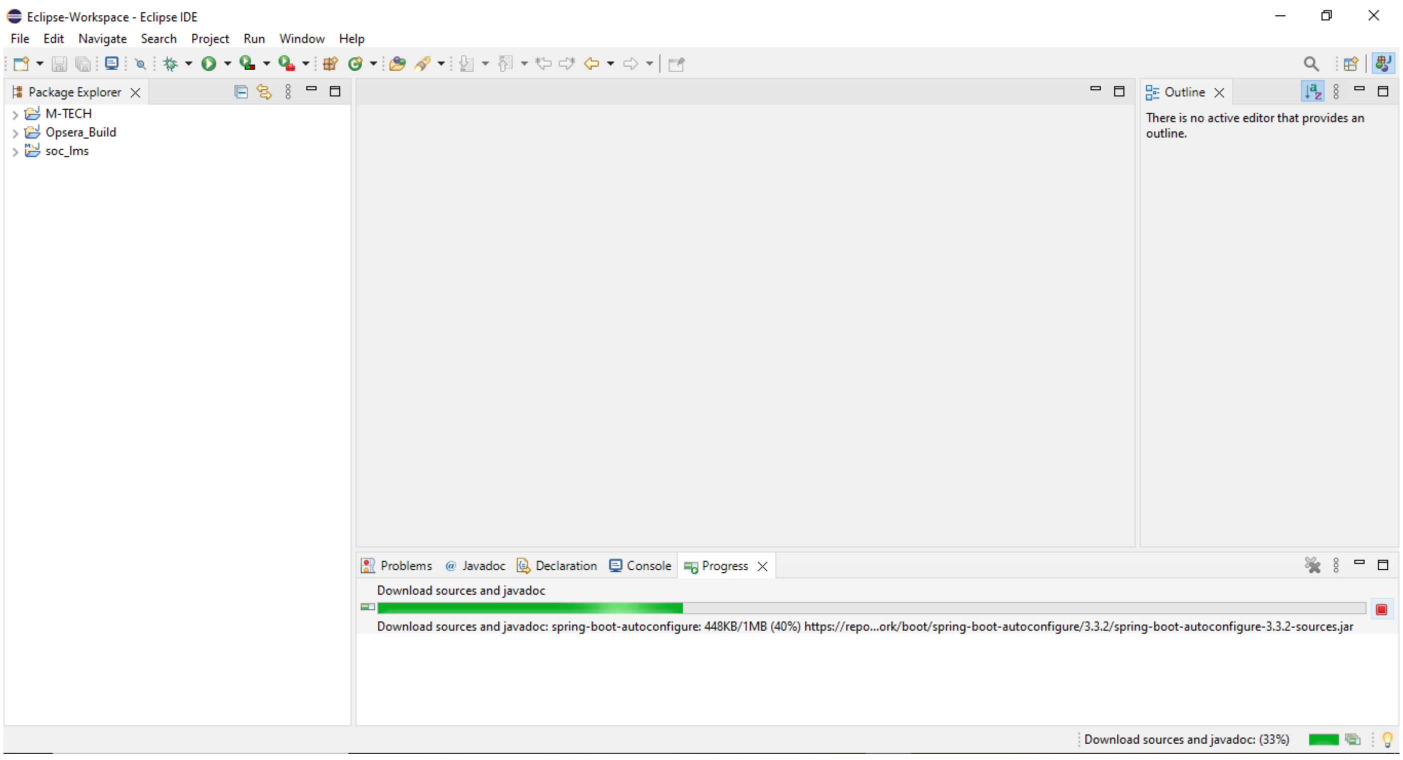
**Spring Web** WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

**Spring Data JPA** SQL
Persist data in SQL stores with Java Persistence API using Spring Data and Hibernate.

**H2 Database** SQL
Provides a fast in-memory database that supports JDBC API and R2DBC access, with a small (2mb) footprint. Supports embedded and server modes as well as a browser based console application.

**Spring Boot DevTools** DEVELOPER TOOLS
Provides fast application restarts, LiveReload, and configurations for enhanced development experience.

## After that we are importing the unzipped file into the Eclipse editor.


Eclipse-Workspace - Eclipse IDE

File Edit Navigate Search Project Run Window Help
Package Explorer
M-TECH
Opsera_Build
soc_lms
Outline
Problems Javadoc Declaration Console Progress

There is no active editor that provides an outline.

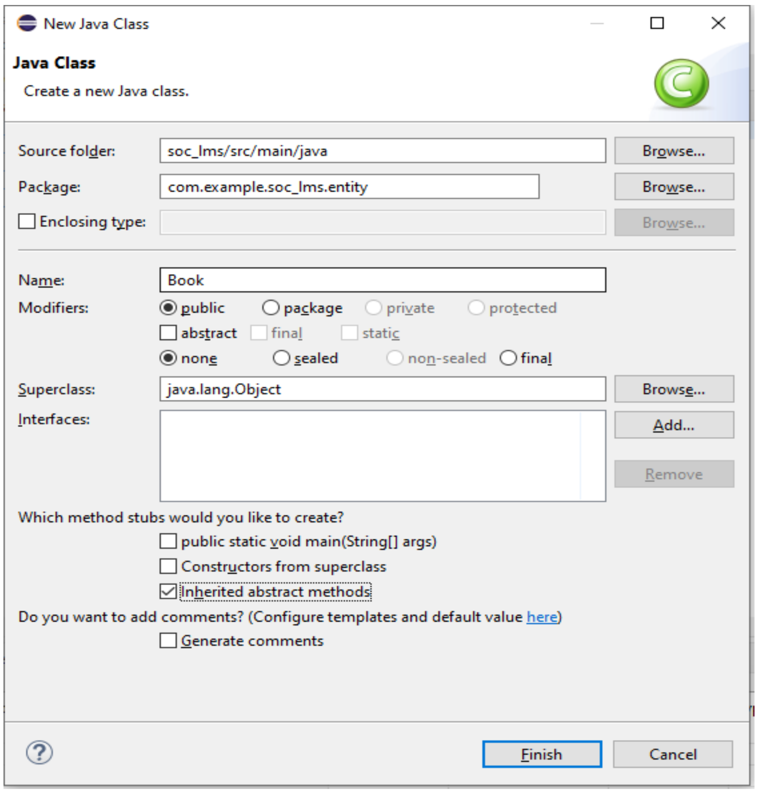
Download sources and javadoc

Download sources and javadoc: spring-boot-autoconfigure: 448KB/1MB (40%) https://repo...ork/boot/spring-boot-autoconfigure/3.3.2/spring-boot-autoconfigure-3.3.2-sources.jar

Download sources and javadoc: (33%)

**Now we are going to create Model files (Entity files):**
**We will create separate packages for each.**

## For Book:



New Java Class

**Java Class**

Create a new Java class.

| | |
|---|---|
| Source folder: | soc_lms/src/main/java |
| Package: | com.example.soc_lms.entity |
| ☐ Enclosing type: | |
| Name: | Book |
| Modifiers: | [✓ public] package private protected |
| | ☐ abstract ☐ final ☐ static |
| | [✓ none] sealed non-sealed final |
| Superclass: | java.lang.Object |
| Interfaces: | |

Which method stubs would you like to create?

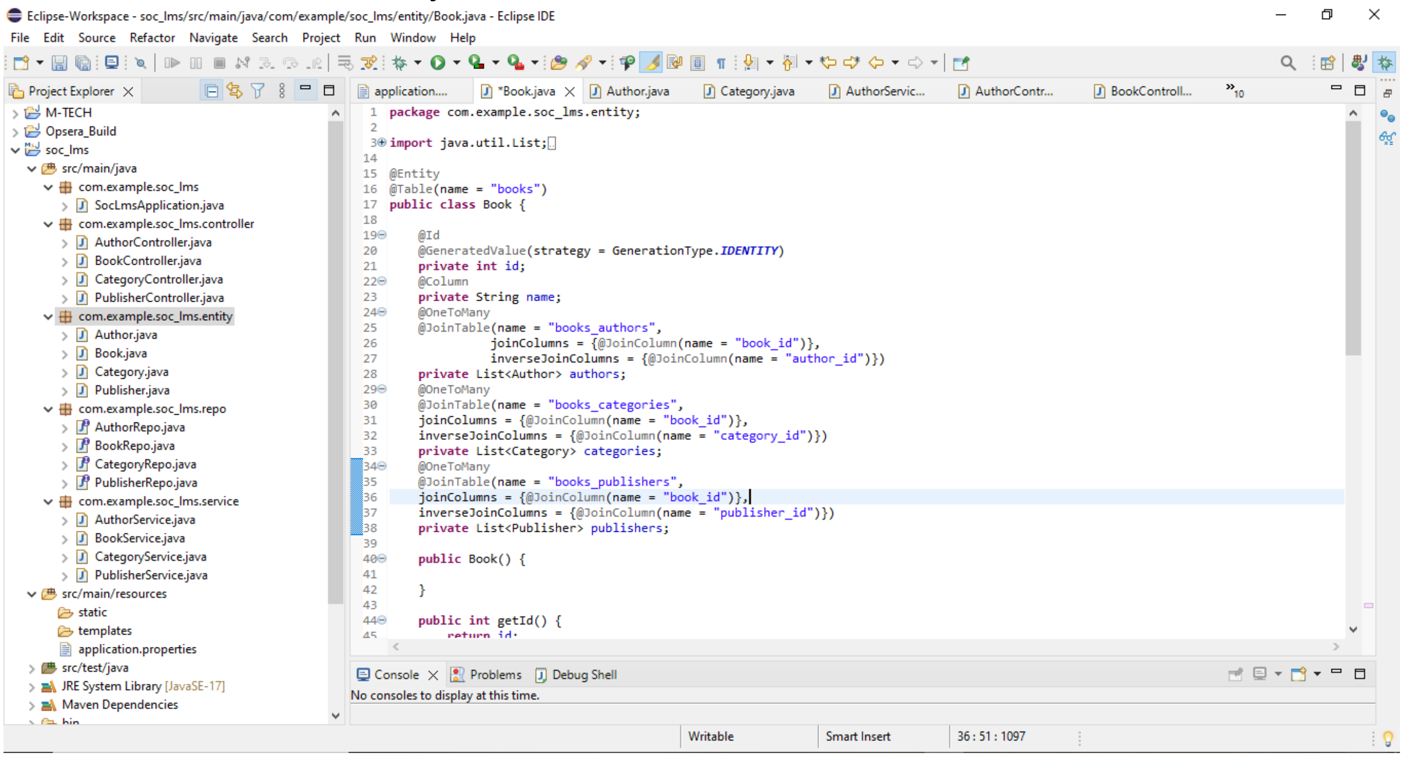
- ☐ public static void main(String[] args)
- ☐ Constructors from superclass
- ☑ Inherited abstract methods

Do you want to add comments? (Configure templates and default value here)

- ☐ Generate comments

Finish  Cancel

## For Book we have created Book.java



Eclipse-Workspace - soc_lms/src/main/java/com/example/soc_lms/entity/Book.java - Eclipse IDE

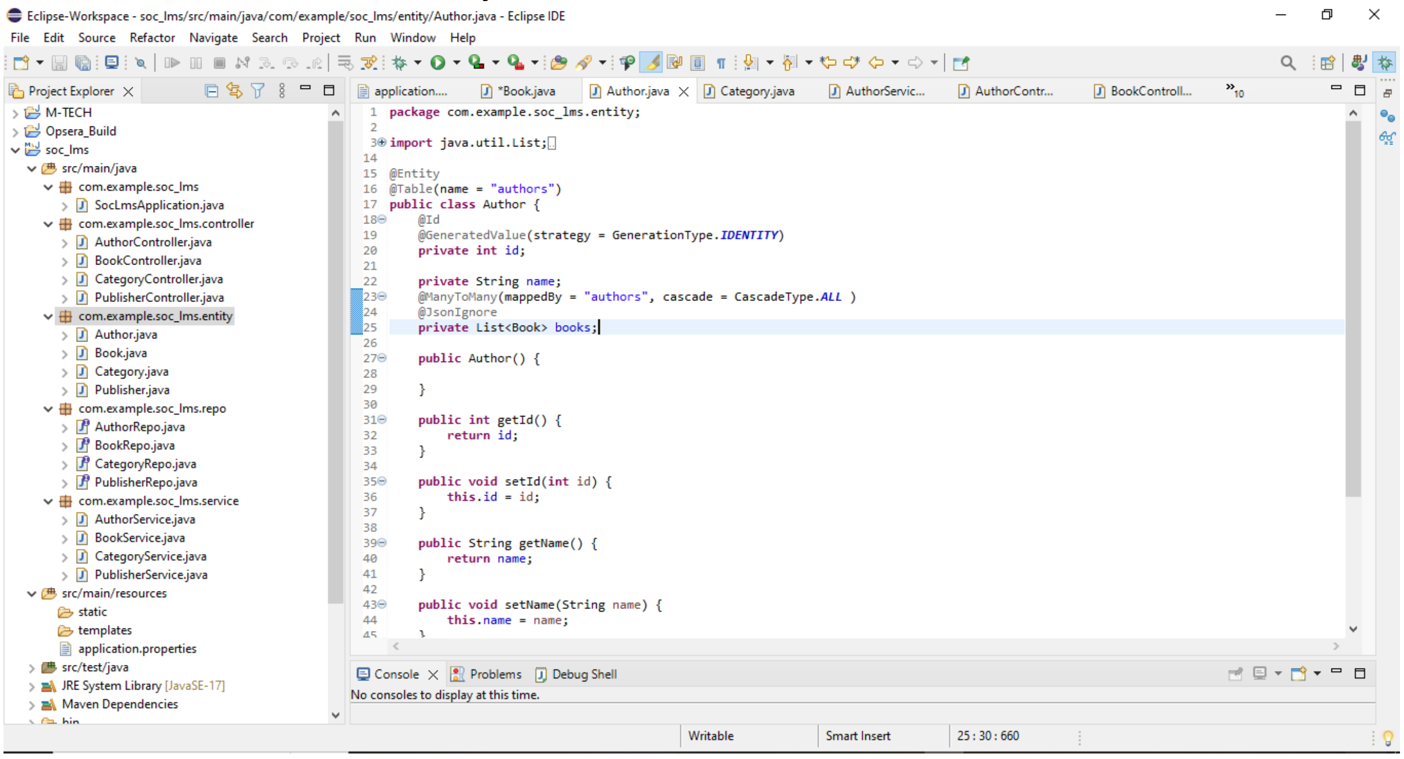
```
M-TECH
Opsera_Build
soc_lms
  src/main/java
    com.example.soc_lms
      SocLmsApplication.java
    com.example.soc_lms.controller
      AuthorController.java
      BookController.java
      CategoryController.java
      PublisherController.java
    com.example.soc_lms.entity
      Author.java
      Book.java
      Category.java
      Publisher.java
    com.example.soc_lms.repo
      AuthorRepo.java
      BookRepo.java
      CategoryRepo.java
      PublisherRepo.java
    com.example.soc_lms.service
      AuthorService.java
      BookService.java
      CategoryService.java
      PublisherService.java
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
```

```java
package com.example.soc_lms.entity;

import java.util.List;

@Entity
@Table(name = "books")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column
    private String name;
    @OneToMany
    @JoinTable(name = "books_authors",
               joinColumns = {@JoinColumn(name = "book_id")},
               inverseJoinColumns = {@JoinColumn(name = "author_id")})
    private List<Author> authors;
    @OneToMany
    @JoinTable(name = "books_categories",
    joinColumns = {@JoinColumn(name = "book_id")},
    inverseJoinColumns = {@JoinColumn(name = "category_id")})
    private List<Category> categories;
    @OneToMany
    @JoinTable(name = "books_publishers",
    joinColumns = {@JoinColumn(name = "book_id")},
    inverseJoinColumns = {@JoinColumn(name = "publisher_id")})
    private List<Publisher> publishers;

    public Book() {

    }

    public int getId() {
        return id;
```

Console: No consoles to display at this time.

Writable | Smart Insert | 36 : 51 : 1097

## For Author we have created Author.java



Eclipse-Workspace - soc_lms/src/main/java/com/example/soc_lms/entity/Author.java - Eclipse IDE

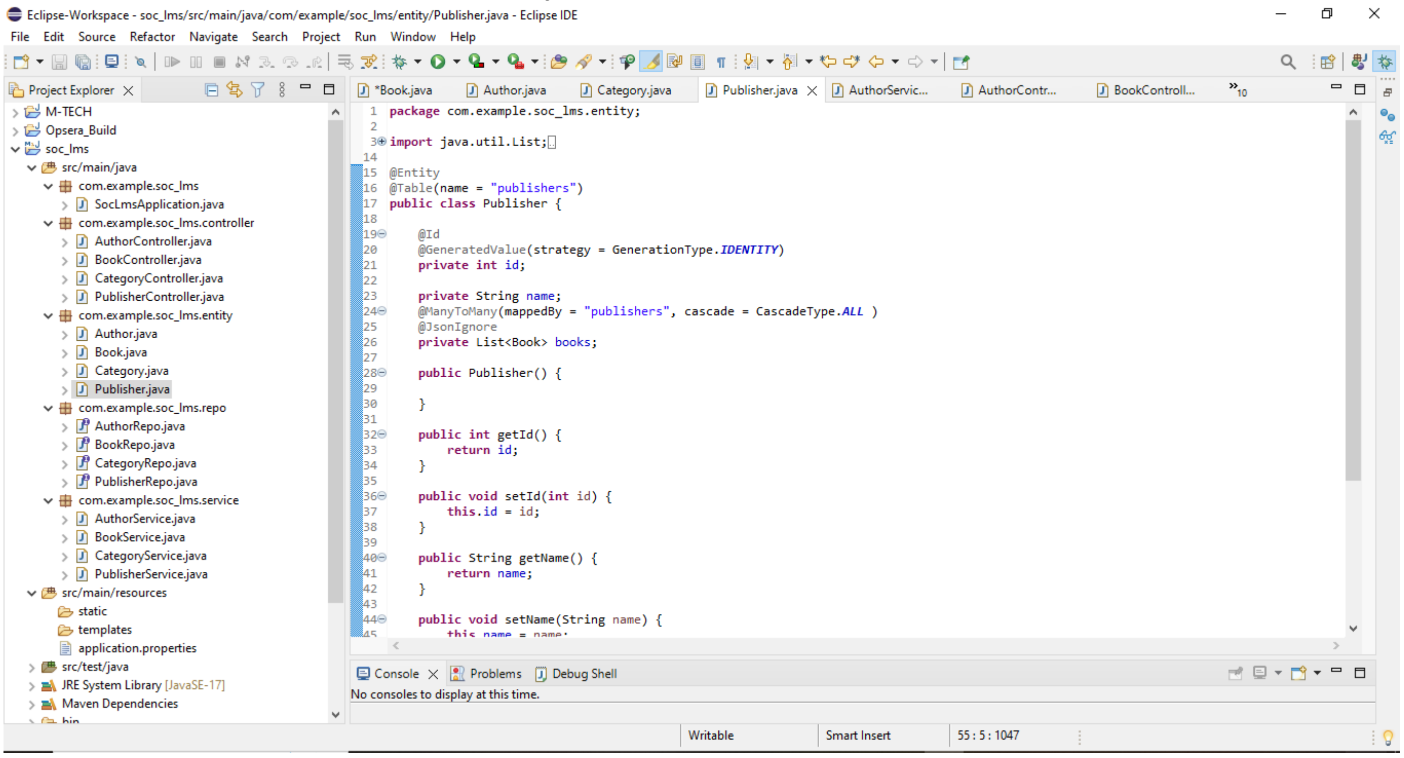
```
soc_lms
  src/main/java
    com.example.soc_lms
      SocLmsApplication.java
    com.example.soc_lms.controller
      AuthorController.java
      BookController.java
      CategoryController.java
      PublisherController.java
    com.example.soc_lms.entity
      Author.java
      Book.java
      Category.java
      Publisher.java
    com.example.soc_lms.repo
      AuthorRepo.java
      BookRepo.java
      CategoryRepo.java
      PublisherRepo.java
    com.example.soc_lms.service
      AuthorService.java
      BookService.java
      CategoryService.java
      PublisherService.java
  src/main/resources
    static
    templates
    application.properties
  src/test/java
  JRE System Library [JavaSE-17]
  Maven Dependencies
```

```java
package com.example.soc_lms.entity;

import java.util.List;

@Entity
@Table(name = "authors")
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    @ManyToMany(mappedBy = "authors", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Author() {

    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

No consoles to display at this time.

Writable | Smart Insert | 25 : 30 : 660

## For Publisher we have created Publisher.java



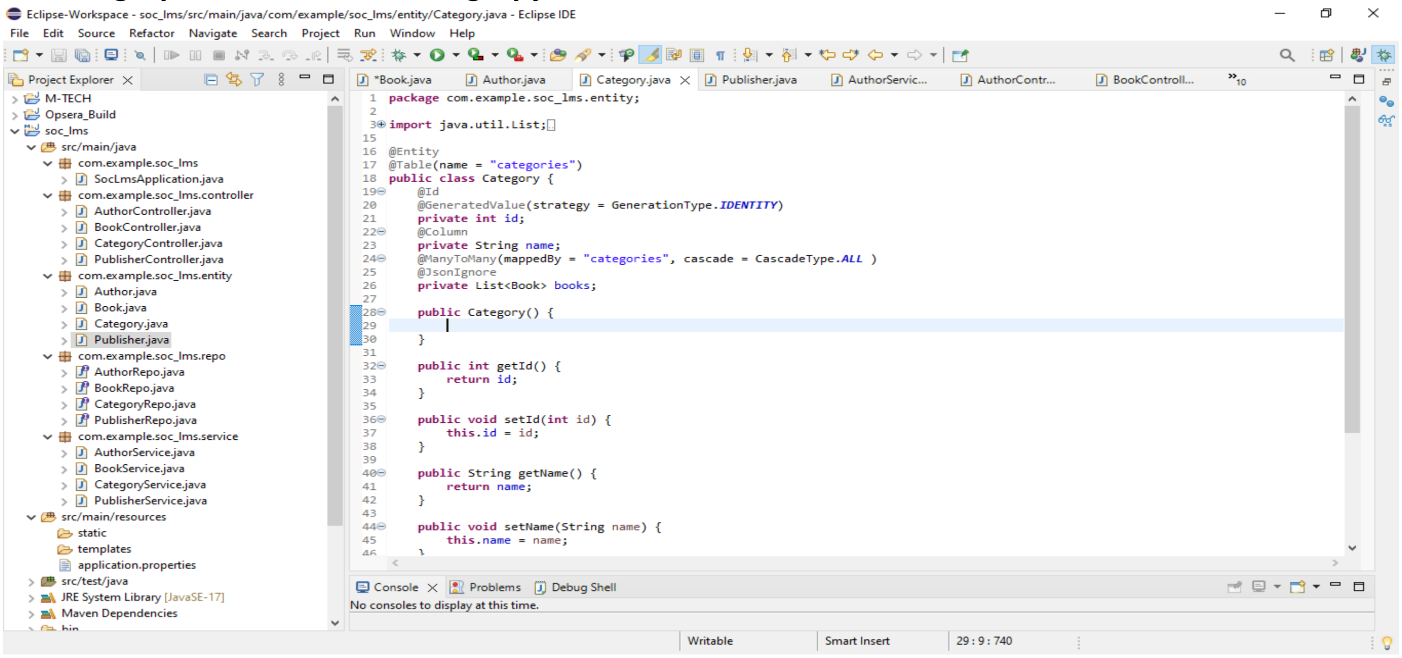
Eclipse-Workspace - soc_lms/src/main/java/com/example/soc_lms/entity/Publisher.java - Eclipse IDE

```java
package com.example.soc_lms.entity;

import java.util.List;

@Entity
@Table(name = "publishers")
public class Publisher {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    @ManyToMany(mappedBy = "publishers", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Publisher() {

    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

No consoles to display at this time.

Writable | Smart Insert | 55 : 5 : 1047

## For Category we have created Category.java

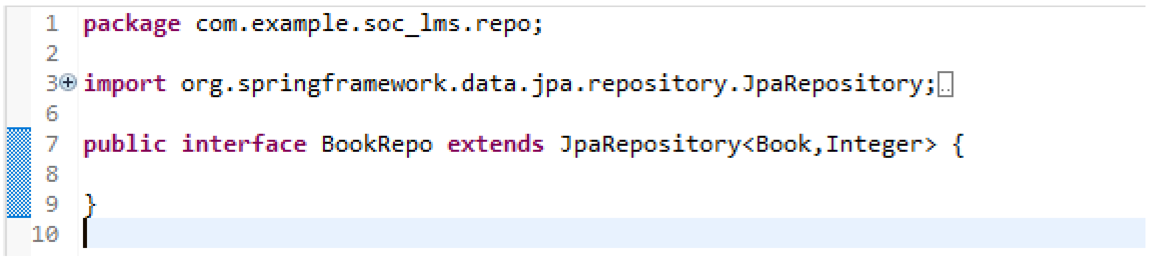


```java
package com.example.soc_lms.entity;

import java.util.List;

@Entity
@Table(name = "categories")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column
    private String name;
    @ManyToMany(mappedBy = "categories", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Category() {
        
    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

```
No consoles to display at this time.
```

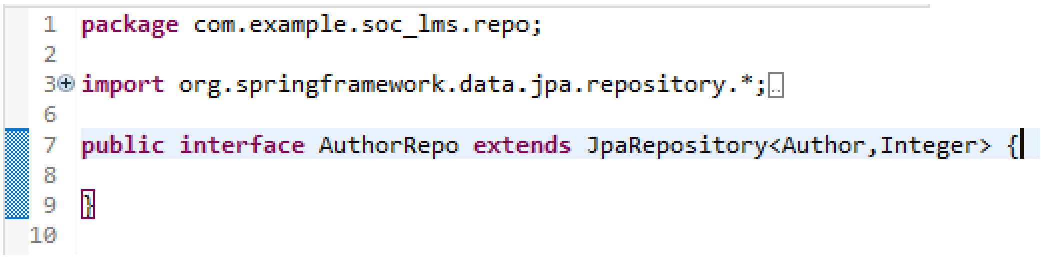
After successfully creating the **Entity** classes we will create the **Repo interfaces**.

For Book:



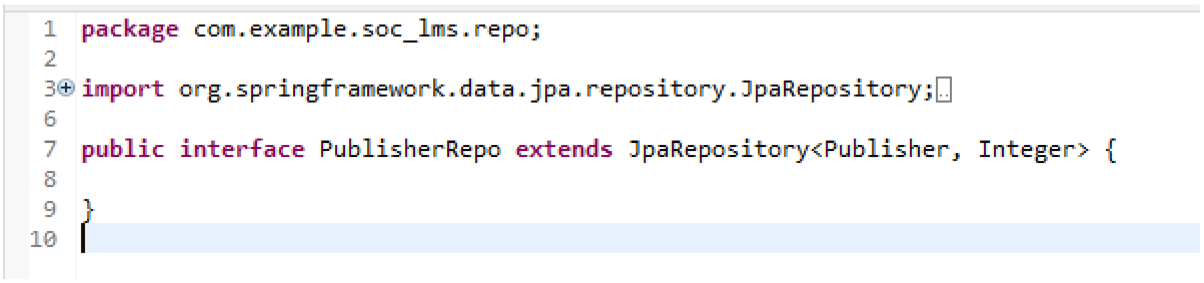
```java
package com.example.soc_lms.repo;

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

public interface BookRepo extends JpaRepository<Book,Integer> {

}

```

For Author:



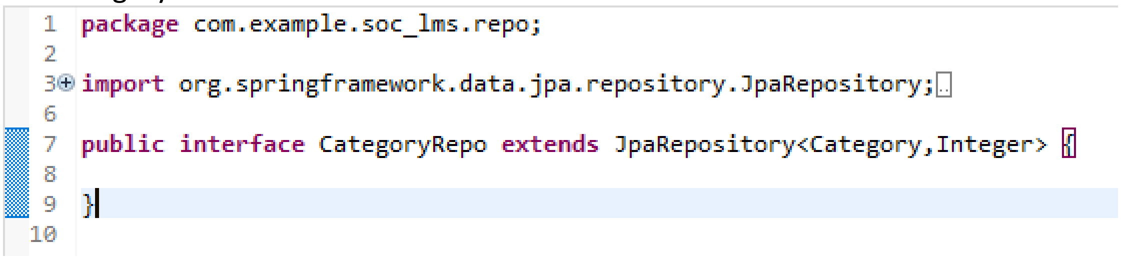
```java
package com.example.soc_lms.repo;

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

public interface AuthorRepo extends JpaRepository<Author,Integer> {

}

```

For Publisher:



```java
package com.example.soc_lms.repo;

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

public interface PublisherRepo extends JpaRepository<Publisher, Integer> {

}

```

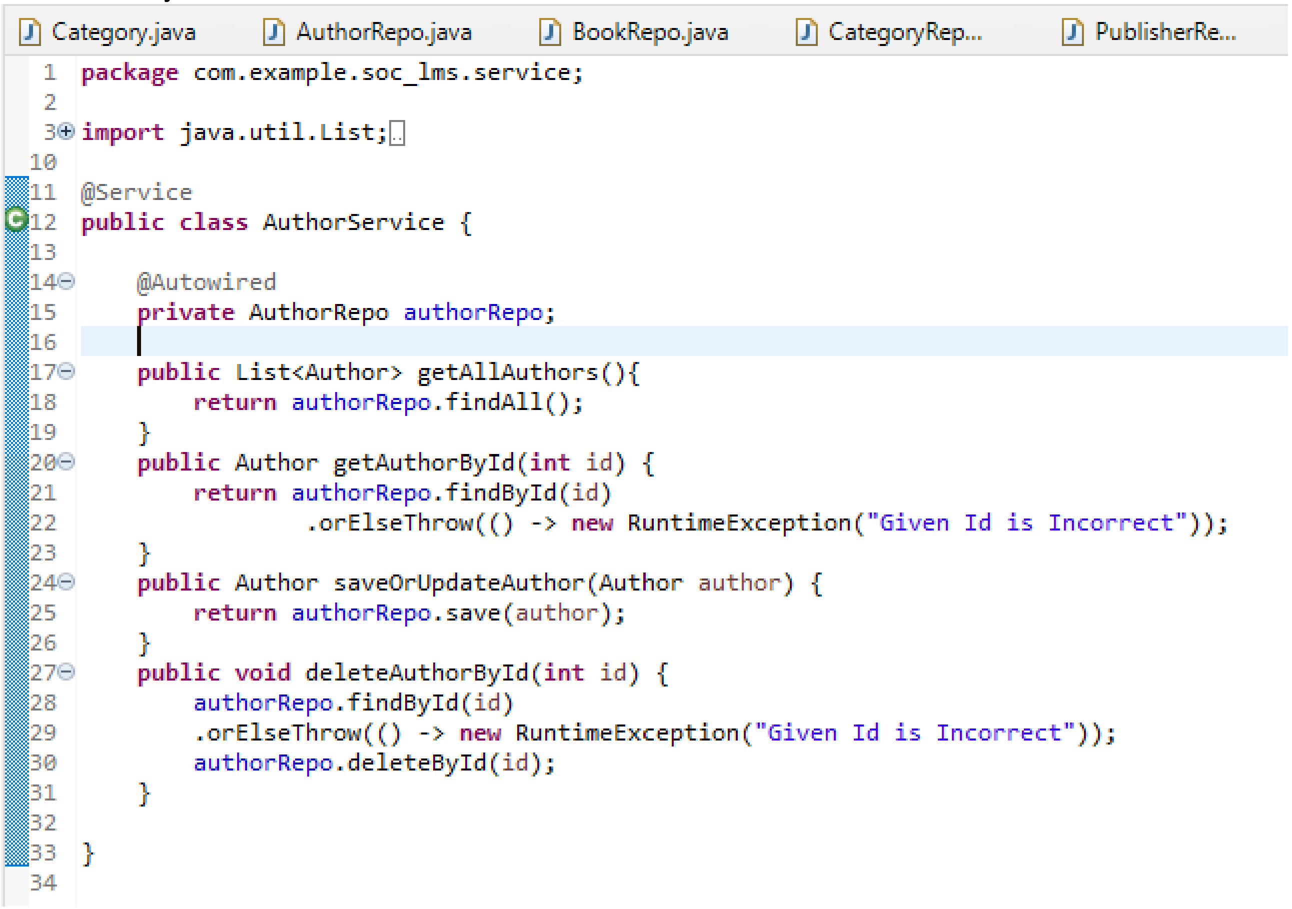
For Category:



```java
package com.example.soc_lms.repo;

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

public interface CategoryRepo extends JpaRepository<Category,Integer> {

}

```

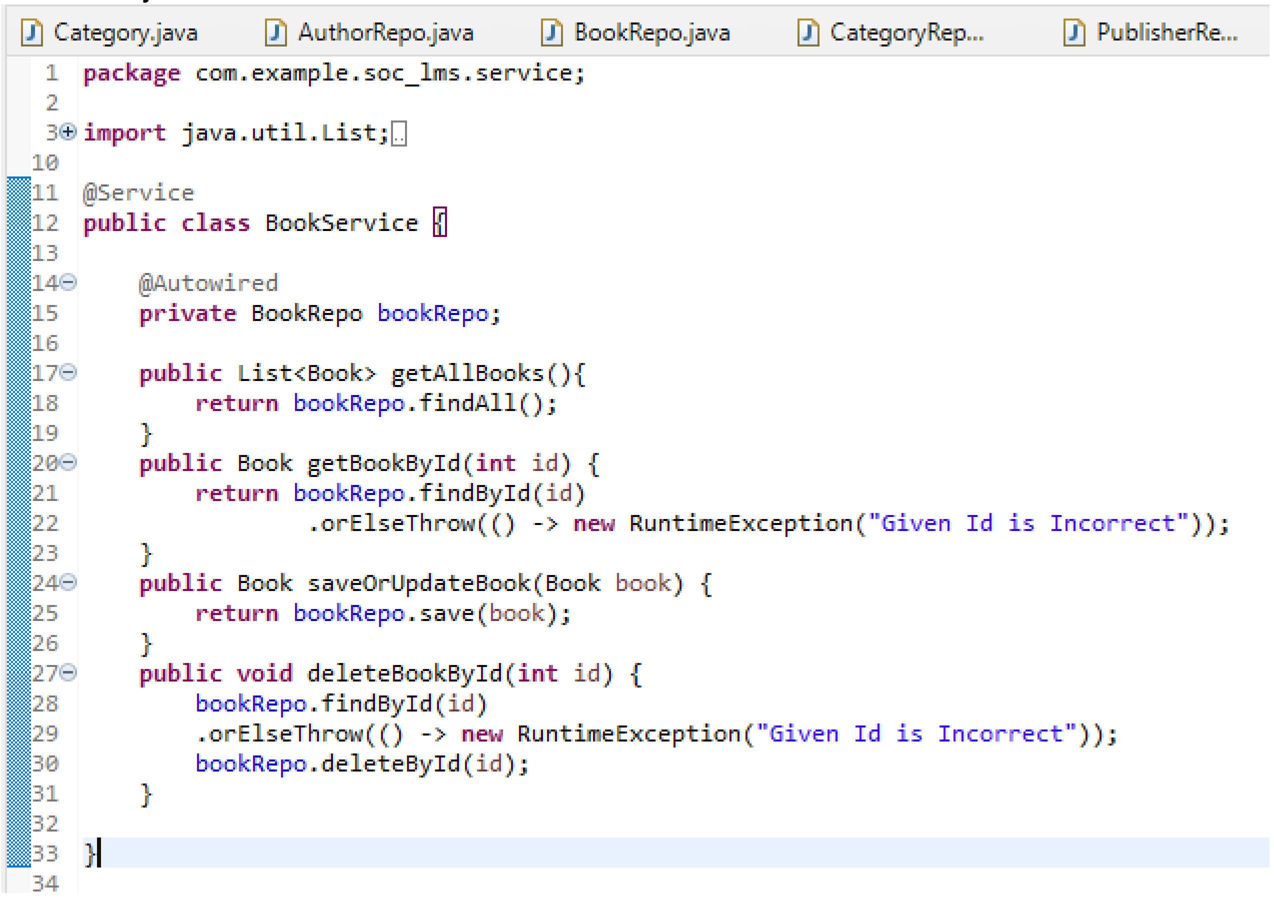
**After that we will create Services for all the entities.**

**AuthorService.java**



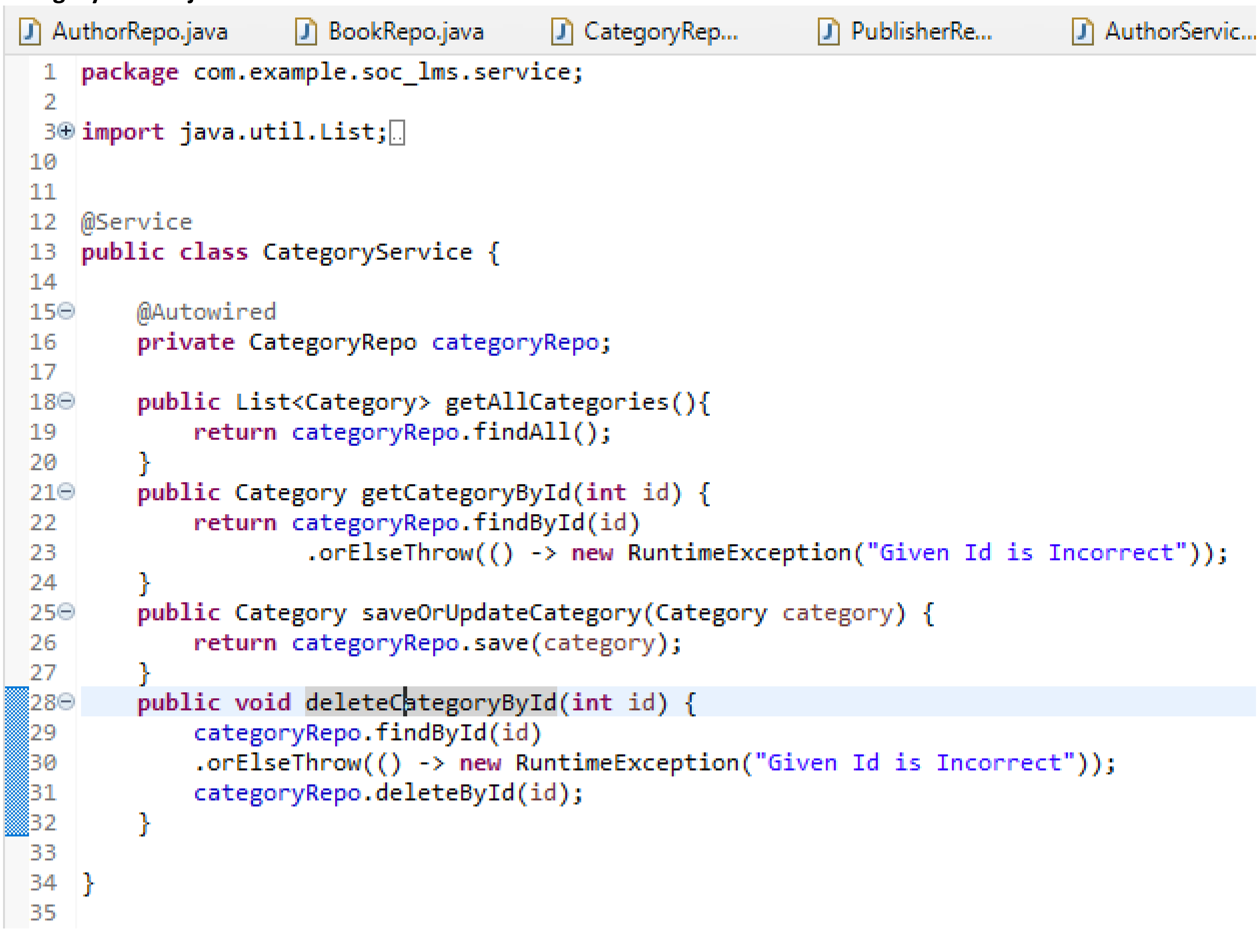
```java
package com.example.soc_lms.service;

import java.util.List;

@Service
public class AuthorService {

    @Autowired
    private AuthorRepo authorRepo;

    public List<Author> getAllAuthors(){
        return authorRepo.findAll();
    }
    public Author getAuthorById(int id) {
        return authorRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Author saveOrUpdateAuthor(Author author) {
        return authorRepo.save(author);
    }
    public void deleteAuthorById(int id) {
        authorRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        authorRepo.deleteById(id);
    }

}
```

**BookService.java**



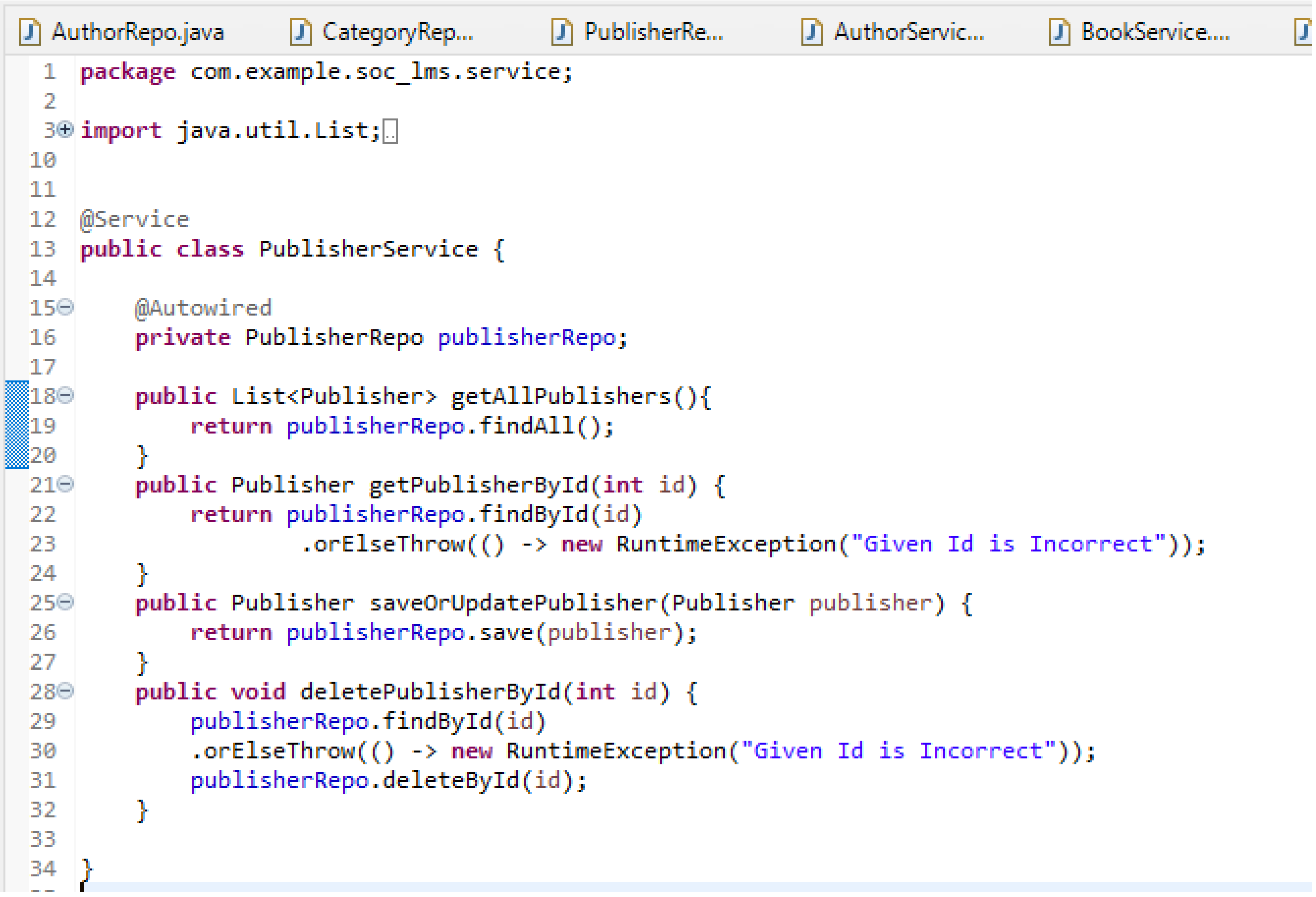
```java
package com.example.soc_lms.service;

import java.util.List;

@Service
public class BookService {

    @Autowired
    private BookRepo bookRepo;

    public List<Book> getAllBooks(){
        return bookRepo.findAll();
    }
    public Book getBookById(int id) {
        return bookRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Book saveOrUpdateBook(Book book) {
        return bookRepo.save(book);
    }
    public void deleteBookById(int id) {
        bookRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        bookRepo.deleteById(id);
    }

}
```

**CategoryService.java**



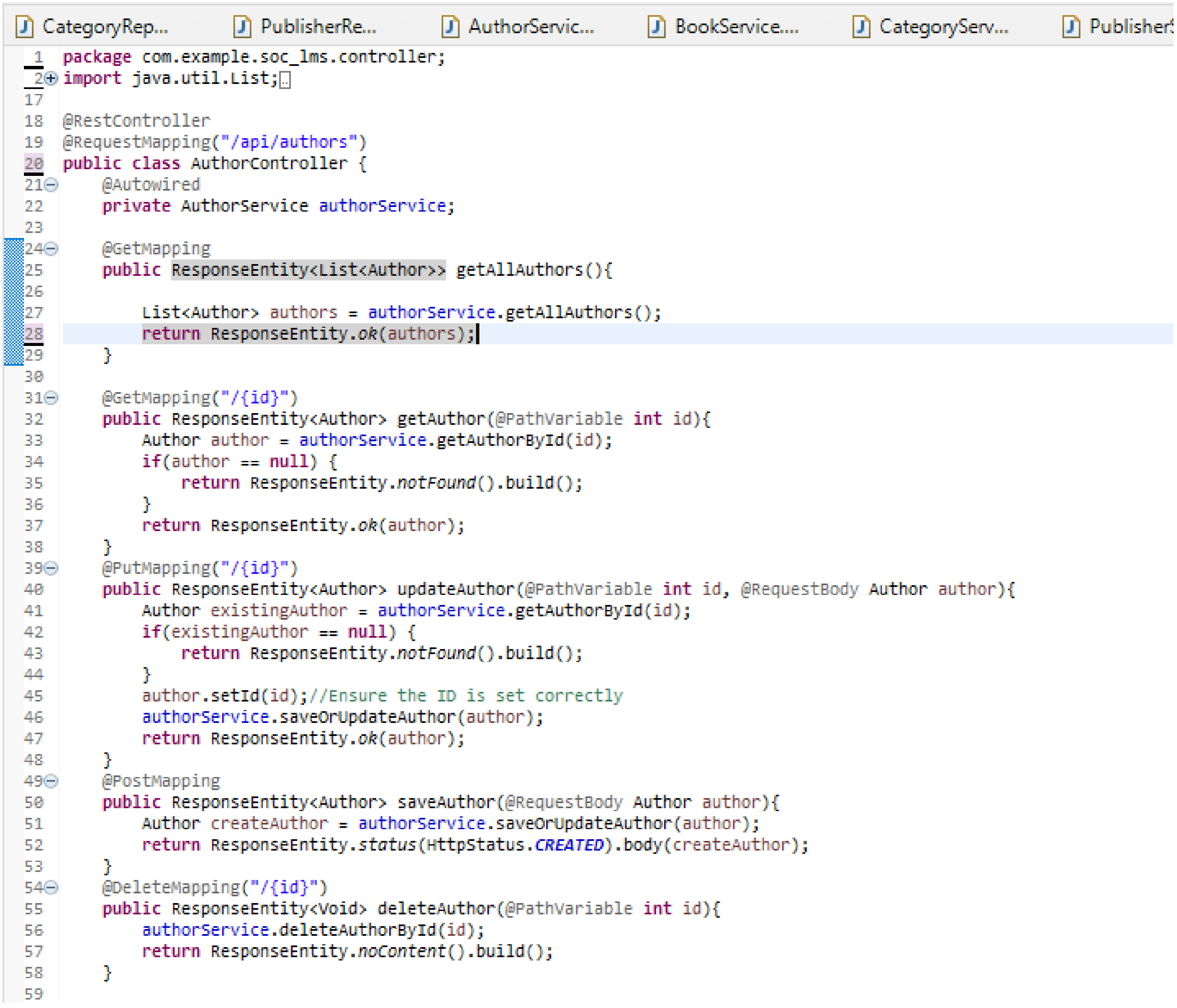
```java
package com.example.soc_lms.service;

import java.util.List;


@Service
public class CategoryService {

    @Autowired
    private CategoryRepo categoryRepo;

    public List<Category> getAllCategories(){
        return categoryRepo.findAll();
    }
    public Category getCategoryById(int id) {
        return categoryRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Category saveOrUpdateCategory(Category category) {
        return categoryRepo.save(category);
    }
    public void deleteCategoryById(int id) {
        categoryRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        categoryRepo.deleteById(id);
    }

}

```

**PublisherService.java**



```java
package com.example.soc_lms.service;

import java.util.List;


@Service
public class PublisherService {

    @Autowired
    private PublisherRepo publisherRepo;

    public List<Publisher> getAllPublishers(){
        return publisherRepo.findAll();
    }
    public Publisher getPublisherById(int id) {
        return publisherRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Publisher saveOrUpdatePublisher(Publisher publisher) {
        return publisherRepo.save(publisher);
    }
    public void deletePublisherById(int id) {
        publisherRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        publisherRepo.deleteById(id);
    }

}
```

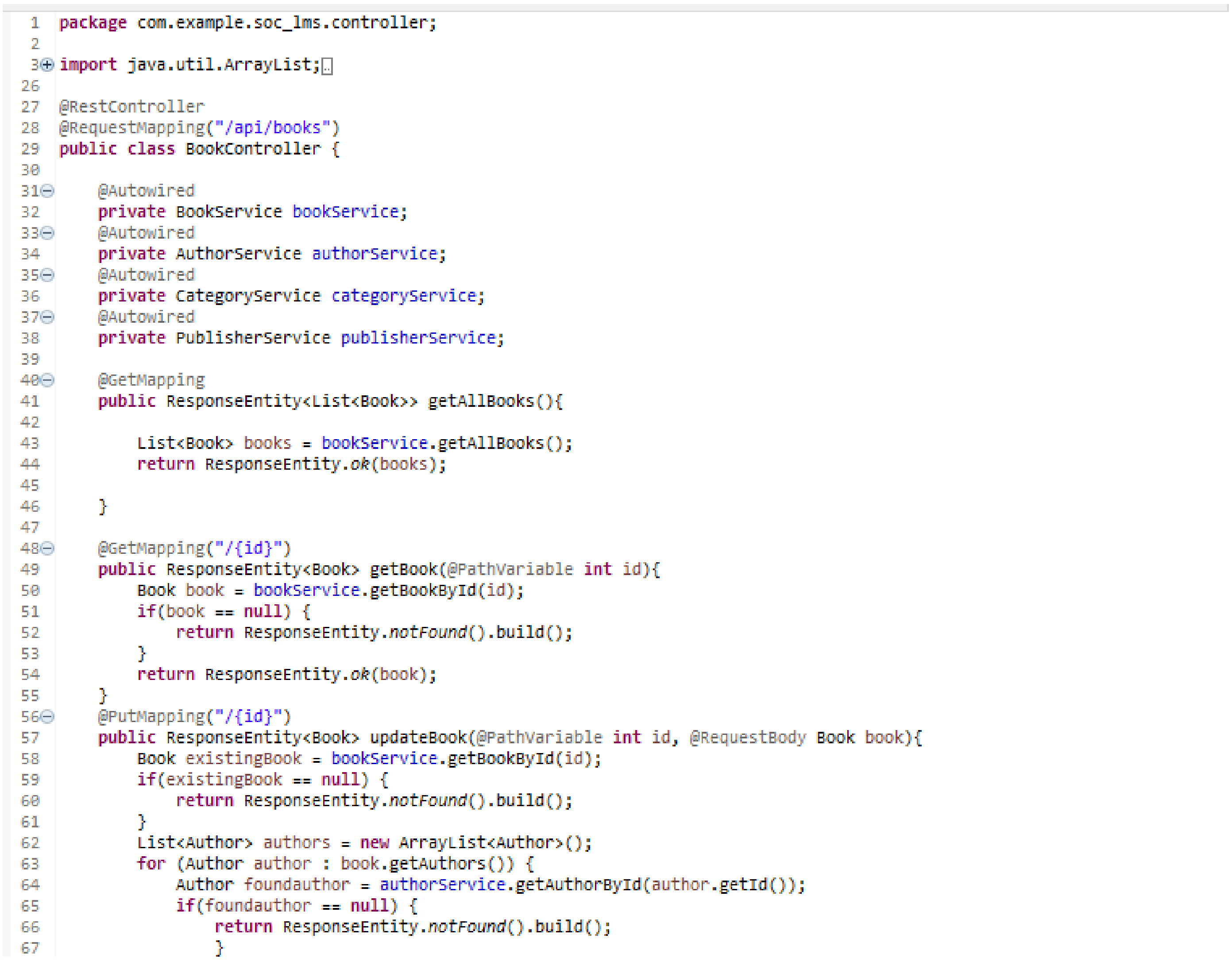
**Now we are going to create the controllers.**



```java
package com.example.soc_lms.controller;
import java.util.List;

@RestController
@RequestMapping("/api/authors")
public class AuthorController {
    @Autowired
    private AuthorService authorService;

    @GetMapping
    public ResponseEntity<List<Author>> getAllAuthors(){

        List<Author> authors = authorService.getAllAuthors();
        return ResponseEntity.ok(authors);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Author> getAuthor(@PathVariable int id){
        Author author = authorService.getAuthorById(id);
        if(author == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(author);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Author> updateAuthor(@PathVariable int id, @RequestBody Author author){
        Author existingAuthor = authorService.getAuthorById(id);
        if(existingAuthor == null) {
            return ResponseEntity.notFound().build();
        }
        author.setId(id);//Ensure the ID is set correctly
        authorService.saveOrUpdateAuthor(author);
        return ResponseEntity.ok(author);
    }
    @PostMapping
    public ResponseEntity<Author> saveAuthor(@RequestBody Author author){
        Author createAuthor = authorService.saveOrUpdateAuthor(author);
        return ResponseEntity.status(HttpStatus.CREATED).body(createAuthor);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteAuthor(@PathVariable int id){
        authorService.deleteAuthorById(id);
        return ResponseEntity.noContent().build();
    }

```

**The above controller is made for the Author. Where we can perform Get,GetById Put, Post, Delete operations over the Author. We can create new Author; we can display the Author using the POST we can Update the Author name using PUT and last but not lease we can delete the Author using DELETE.**

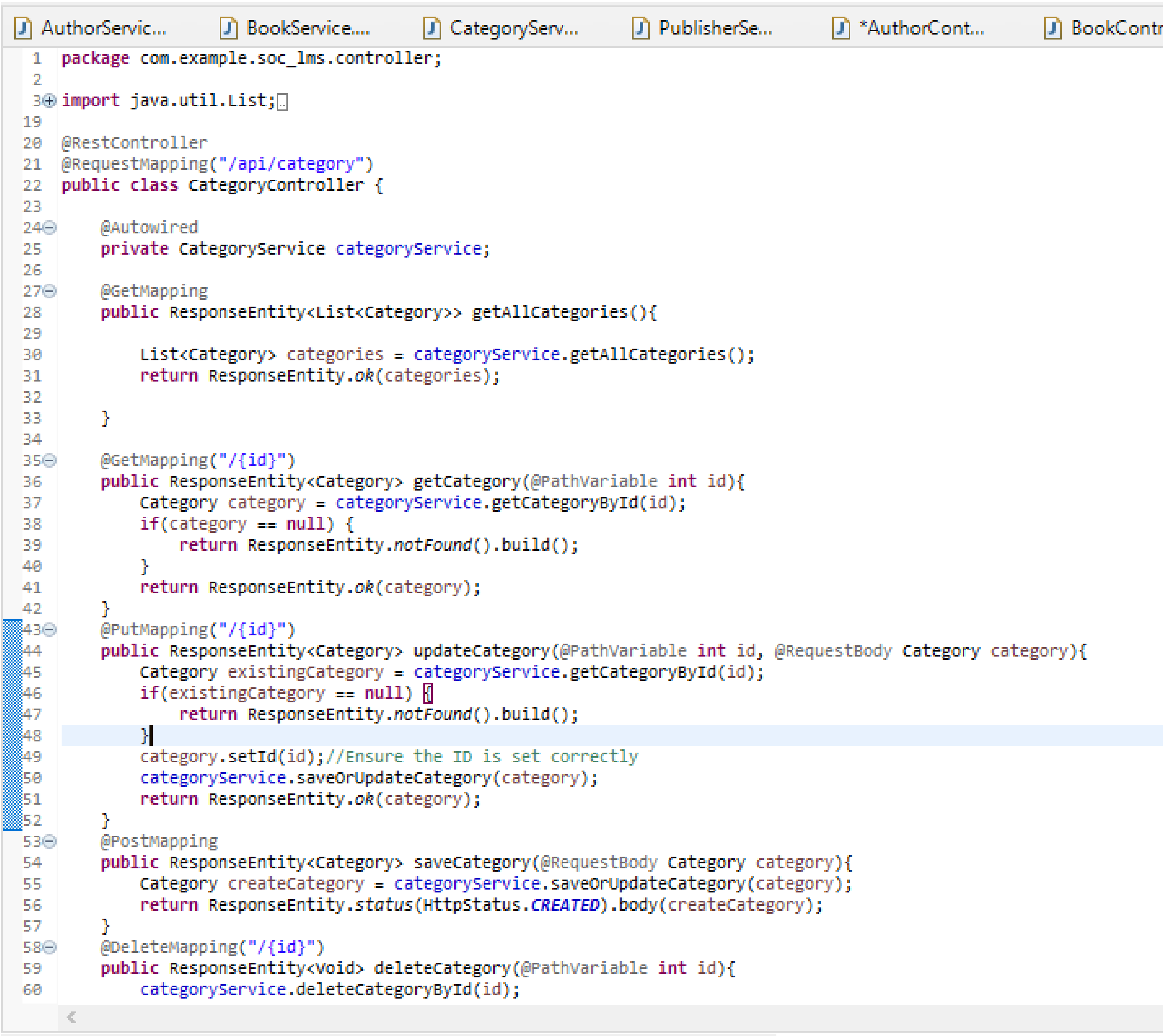
**Note: The above source code is attached with the given folder.**



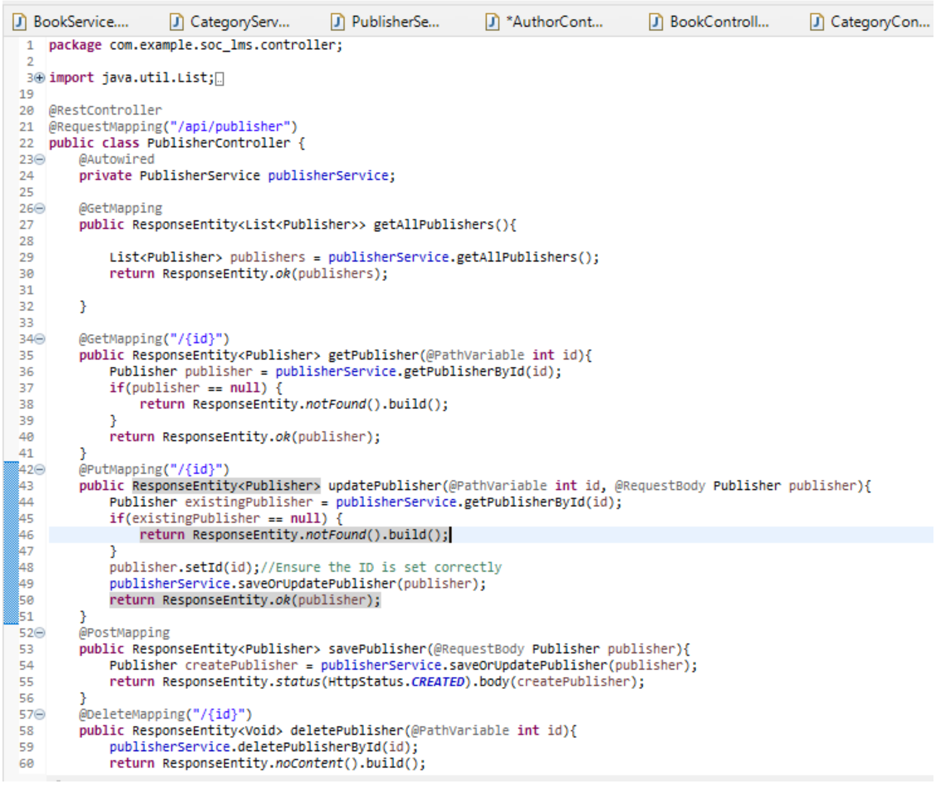
```java
package com.example.soc_lms.controller;

import java.util.ArrayList;

@RestController
@RequestMapping("/api/books")
public class BookController {

    @Autowired
    private BookService bookService;
    @Autowired
    private AuthorService authorService;
    @Autowired
    private CategoryService categoryService;
    @Autowired
    private PublisherService publisherService;

    @GetMapping
    public ResponseEntity<List<Book>> getAllBooks(){

        List<Book> books = bookService.getAllBooks();
        return ResponseEntity.ok(books);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Book> getBook(@PathVariable int id){
        Book book = bookService.getBookById(id);
        if(book == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(book);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable int id, @RequestBody Book book){
        Book existingBook = bookService.getBookById(id);
        if(existingBook == null) {
            return ResponseEntity.notFound().build();
        }
        List<Author> authors = new ArrayList<Author>();
        for (Author author : book.getAuthors()) {
            Author foundauthor = authorService.getAuthorById(author.getId());
            if(foundauthor == null) {
                return ResponseEntity.notFound().build();
            }
```

**The above controller is made for Book. Where we can perform Get,GetById Put, Post, Delete operations over the Book. We can create new Book; we can display the Author using the POST we can Update the Book name using PUT and last but not lease we can delete the Author using DELETE.**

**Note: The above source code is attached with the given folder.**

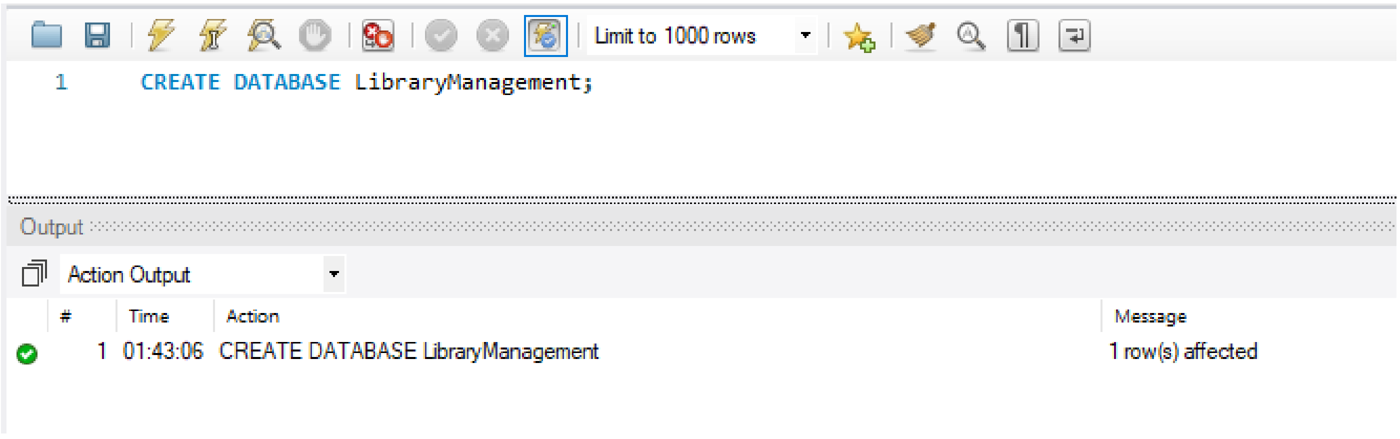


```java
package com.example.soc_lms.controller;

import java.util.List;

@RestController
@RequestMapping("/api/category")
public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    @GetMapping
    public ResponseEntity<List<Category>> getAllCategories(){

        List<Category> categories = categoryService.getAllCategories();
        return ResponseEntity.ok(categories);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Category> getCategory(@PathVariable int id){
        Category category = categoryService.getCategoryById(id);
        if(category == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(category);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Category> updateCategory(@PathVariable int id, @RequestBody Category category){
        Category existingCategory = categoryService.getCategoryById(id);
        if(existingCategory == null) {
            return ResponseEntity.notFound().build();
        }
        category.setId(id);//Ensure the ID is set correctly
        categoryService.saveOrUpdateCategory(category);
        return ResponseEntity.ok(category);
    }
    @PostMapping
    public ResponseEntity<Category> saveCategory(@RequestBody Category category){
        Category createCategory = categoryService.saveOrUpdateCategory(category);
        return ResponseEntity.status(HttpStatus.CREATED).body(createCategory);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCategory(@PathVariable int id){
        categoryService.deleteCategoryById(id);
```



```java
package com.example.soc_lms.controller;

import java.util.List;

@RestController
@RequestMapping("/api/publisher")
public class PublisherController {
    @Autowired
    private PublisherService publisherService;

    @GetMapping
    public ResponseEntity<List<Publisher>> getAllPublishers(){

        List<Publisher> publishers = publisherService.getAllPublishers();
        return ResponseEntity.ok(publishers);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Publisher> getPublisher(@PathVariable int id){
        Publisher publisher = publisherService.getPublisherById(id);
        if(publisher == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(publisher);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Publisher> updatePublisher(@PathVariable int id, @RequestBody Publisher publisher){
        Publisher existingPublisher = publisherService.getPublisherById(id);
        if(existingPublisher == null) {
            return ResponseEntity.notFound().build();
        }
        publisher.setId(id);//Ensure the ID is set correctly
        publisherService.saveOrUpdatePublisher(publisher);
        return ResponseEntity.ok(publisher);
    }
    @PostMapping
    public ResponseEntity<Publisher> savePublisher(@RequestBody Publisher publisher){
        Publisher createPublisher = publisherService.saveOrUpdatePublisher(publisher);
        return ResponseEntity.status(HttpStatus.CREATED).body(createPublisher);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deletePublisher(@PathVariable int id){
        publisherService.deletePublisherById(id);
        return ResponseEntity.noContent().build();
```

**Now we are going to create a Database in MySQL. As we don't have Xampp or h2 database.**

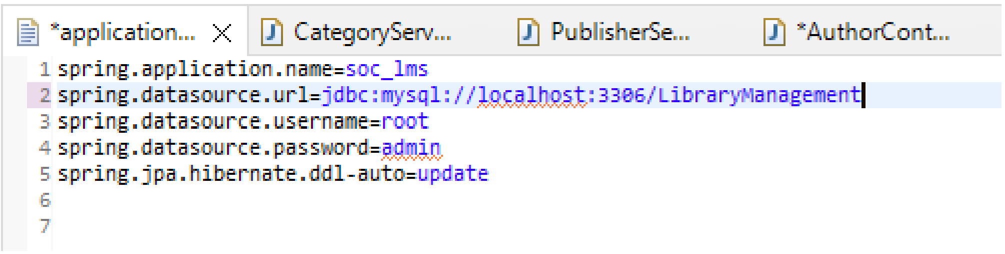


```
CREATE DATABASE LibraryManagement;
```

```
#   Time      Action                               Message
1   01:43:06  CREATE DATABASE LibraryManagement    1 row(s) affected
```

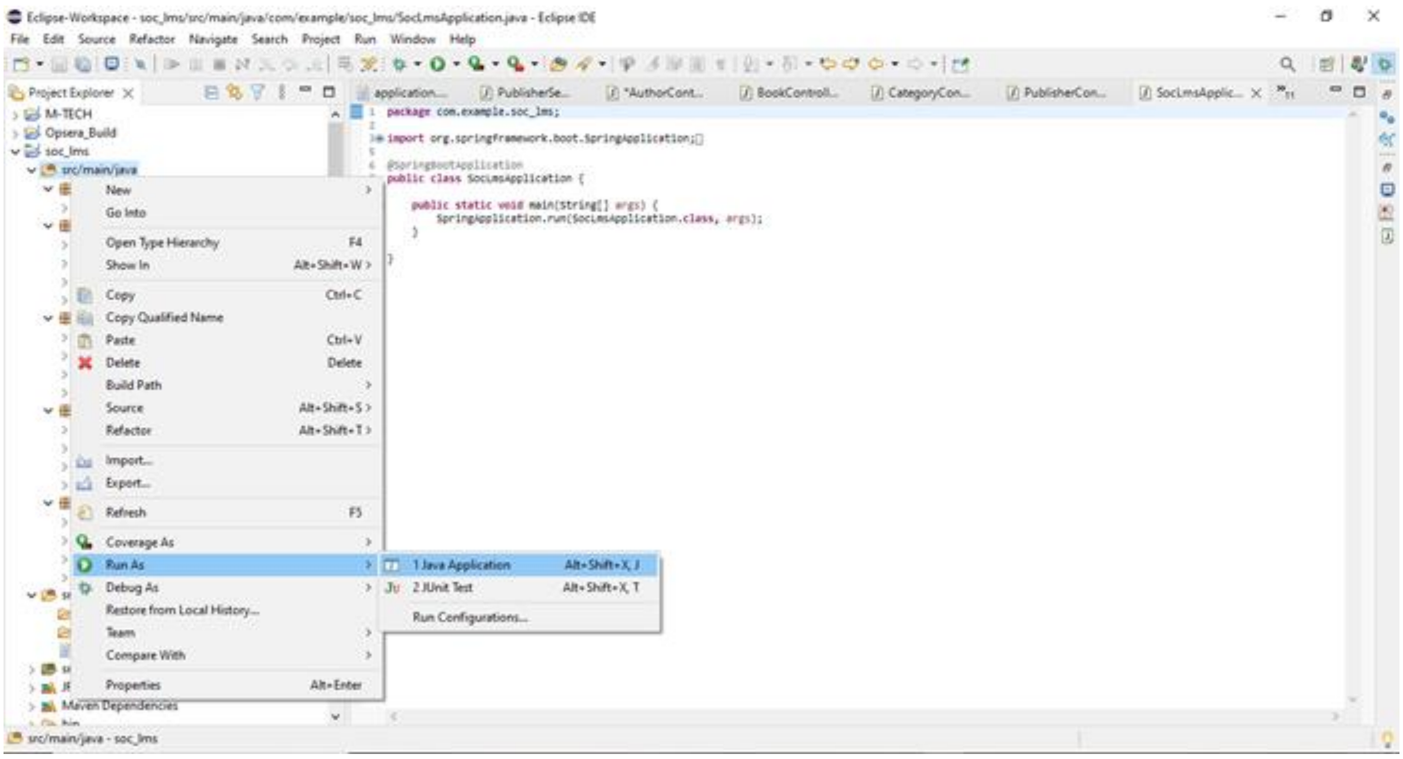
**We have given the Database name as LibraryManagement.**

**#Now we need to set the application.properties file with the database credentials.**
**Path: \soc_lms\src\main\resources\application.properties**



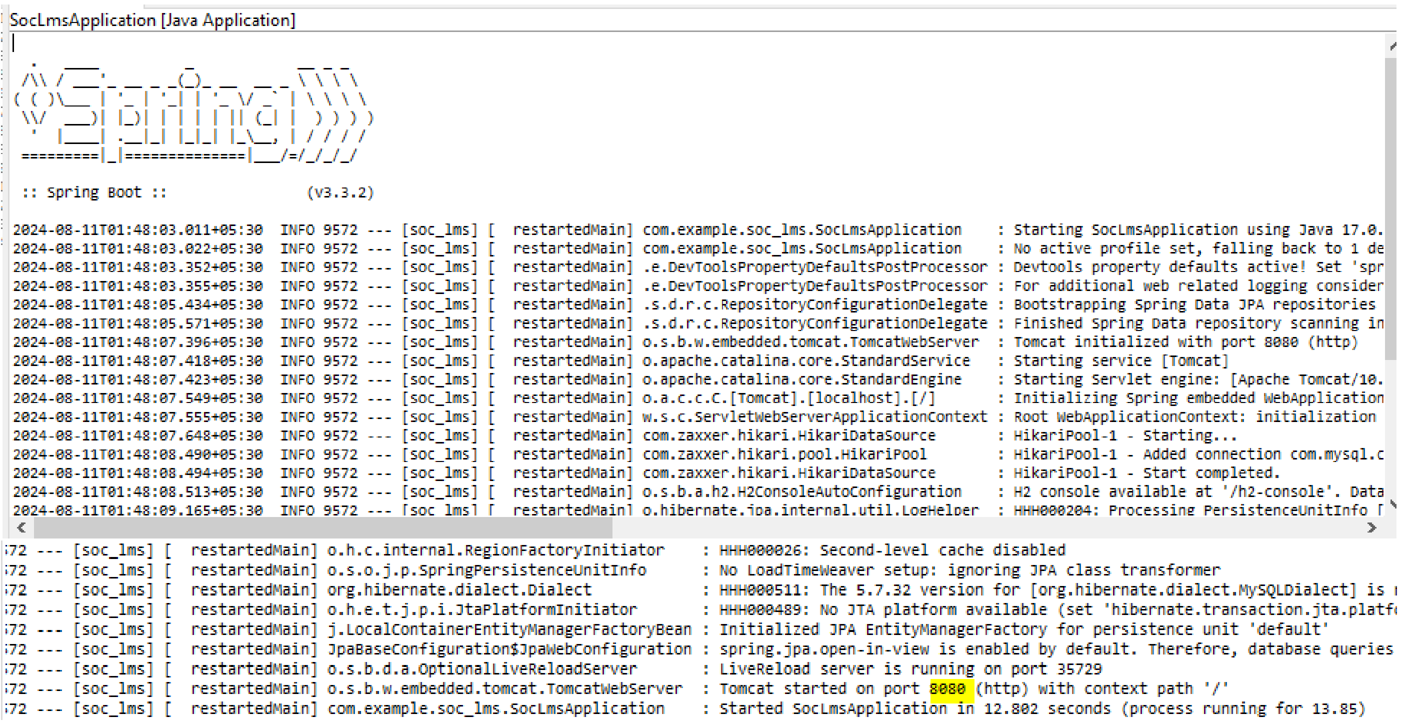
```
spring.application.name=soc_lms
spring.datasource.url=jdbc:mysql://localhost:3306/LibraryManagement
spring.datasource.username=root
spring.datasource.password=admin
spring.jpa.hibernate.ddl-auto=update
```

**All set. Let's try running the code.**



```
package com.example.soc_lms;

import org.springframework.boot.SpringApplication;

@SpringBootApplication
public class SocLmsApplication {

    public static void main(String[] args) {
        SpringApplication.run(SocLmsApplication.class, args);
    }

}
```

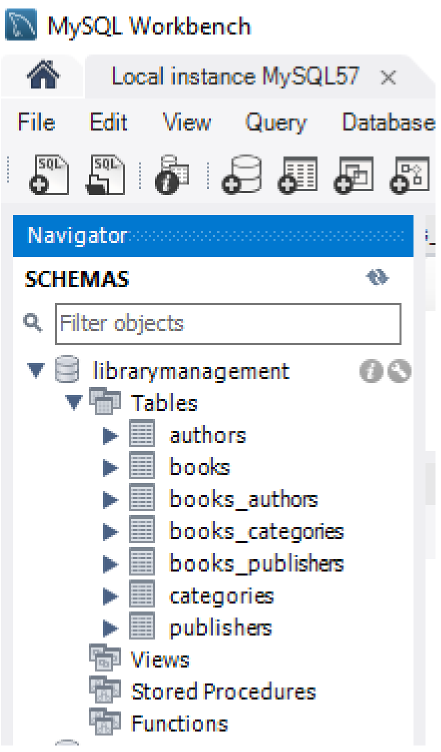
**Here we can see my application got executed successfully and my application got exposed at port 8080**



```
SocLmsApplication [Java Application]

  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/

 :: Spring Boot ::                (v3.3.2)

2024-08-11T01:48:03.011+05:30  INFO 9572 --- [soc_lms] [  restartedMain] com.example.soc_lms.SocLmsApplication    : Starting SocLmsApplication using Java 17.0.
2024-08-11T01:48:03.022+05:30  INFO 9572 --- [soc_lms] [  restartedMain] com.example.soc_lms.SocLmsApplication    : No active profile set, falling back to 1 de
2024-08-11T01:48:03.352+05:30  INFO 9572 --- [soc_lms] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spr
2024-08-11T01:48:03.355+05:30  INFO 9572 --- [soc_lms] [  restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : For additional web related logging consider
2024-08-11T01:48:05.434+05:30  INFO 9572 --- [soc_lms] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories
2024-08-11T01:48:05.571+05:30  INFO 9572 --- [soc_lms] [  restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in
2024-08-11T01:48:07.396+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat initialized with port 8080 (http)
2024-08-11T01:48:07.418+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.apache.catalina.core.StandardService   : Starting service [Tomcat]
2024-08-11T01:48:07.423+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.apache.catalina.core.StandardEngine    : Starting Servlet engine: [Apache Tomcat/10.
2024-08-11T01:48:07.549+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring embedded WebApplication
2024-08-11T01:48:07.555+05:30  INFO 9572 --- [soc_lms] [  restartedMain] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization
2024-08-11T01:48:07.648+05:30  INFO 9572 --- [soc_lms] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Starting...
2024-08-11T01:48:08.490+05:30  INFO 9572 --- [soc_lms] [  restartedMain] com.zaxxer.hikari.pool.HikariPool        : HikariPool-1 - Added connection com.mysql.c
2024-08-11T01:48:08.494+05:30  INFO 9572 --- [soc_lms] [  restartedMain] com.zaxxer.hikari.HikariDataSource       : HikariPool-1 - Start completed.
2024-08-11T01:48:08.513+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Data
2024-08-11T01:48:09.165+05:30  INFO 9572 --- [soc_lms] [  restartedMain] o.hibernate.jpa.internal.util.LogHelper  : HHH000204: Processing PersistenceUnitInfo [
```

```
i72 --- [soc_lms] [  restartedMain] o.h.c.internal.RegionFactoryInitiator    : HHH000026: Second-level cache disabled
i72 --- [soc_lms] [  restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo      : No LoadTimeWeaver setup: ignoring JPA class transformer
i72 --- [soc_lms] [  restartedMain] org.hibernate.dialect.Dialect            : HHH000511: The 5.7.32 version for [org.hibernate.dialect.MySQLDialect] is
i72 --- [soc_lms] [  restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platf
i72 --- [soc_lms] [  restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
i72 --- [soc_lms] [  restartedMain] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries
i72 --- [soc_lms] [  restartedMain] o.s.b.d.a.OptionalLiveReloadServer       : LiveReload server is running on port 35729
i72 --- [soc_lms] [  restartedMain] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
i72 --- [soc_lms] [  restartedMain] com.example.soc_lms.SocLmsApplication    : Started SocLmsApplication in 12.802 seconds (process running for 13.85)
```

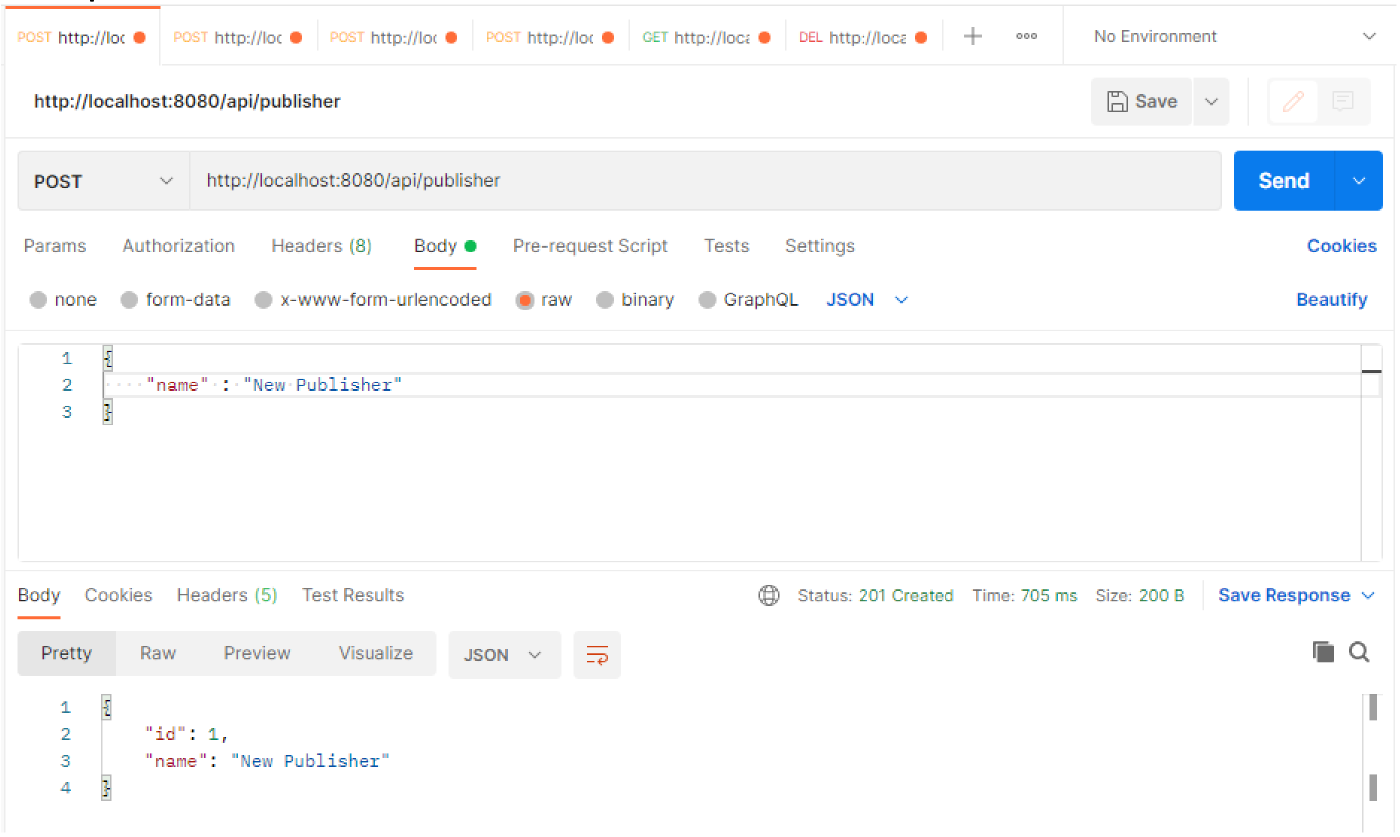
**Let's see whether it is working fine or not.**



**We can see our tables are reflected in MySQL. Now let's try some operation on it.**

**Here we are using Postman for Testing CRUD operations.**

## Post Operation:



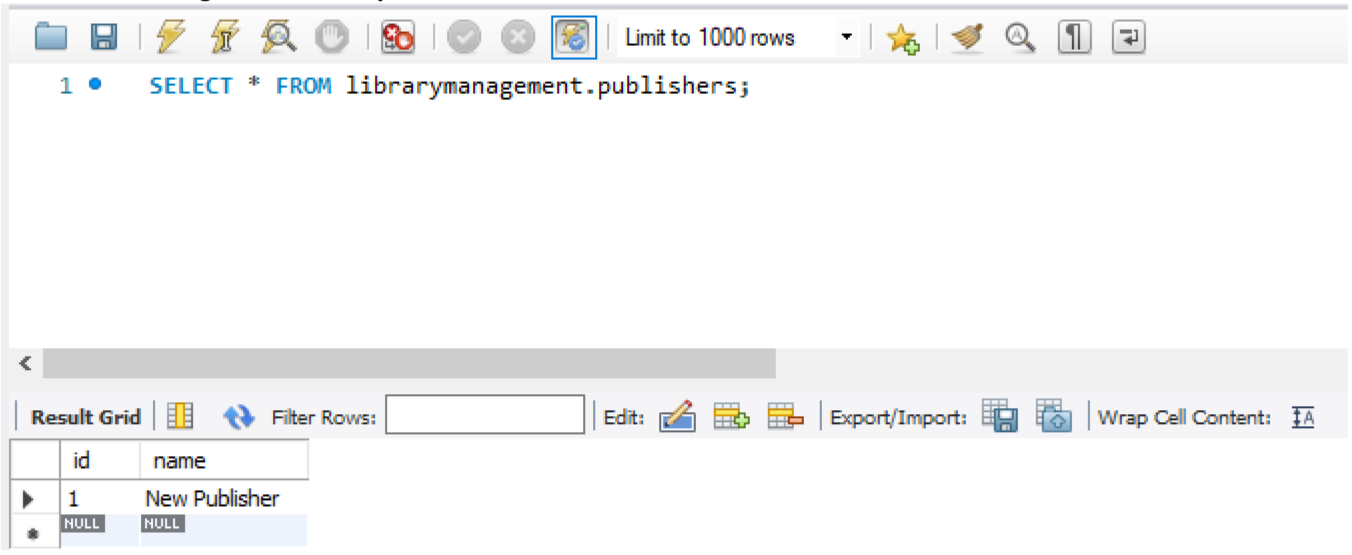
POST http://localhost:8080/api/publisher

```
{
    "name" : "New Publisher"
}
```

Status: 201 Created Time: 705 ms Size: 200 B

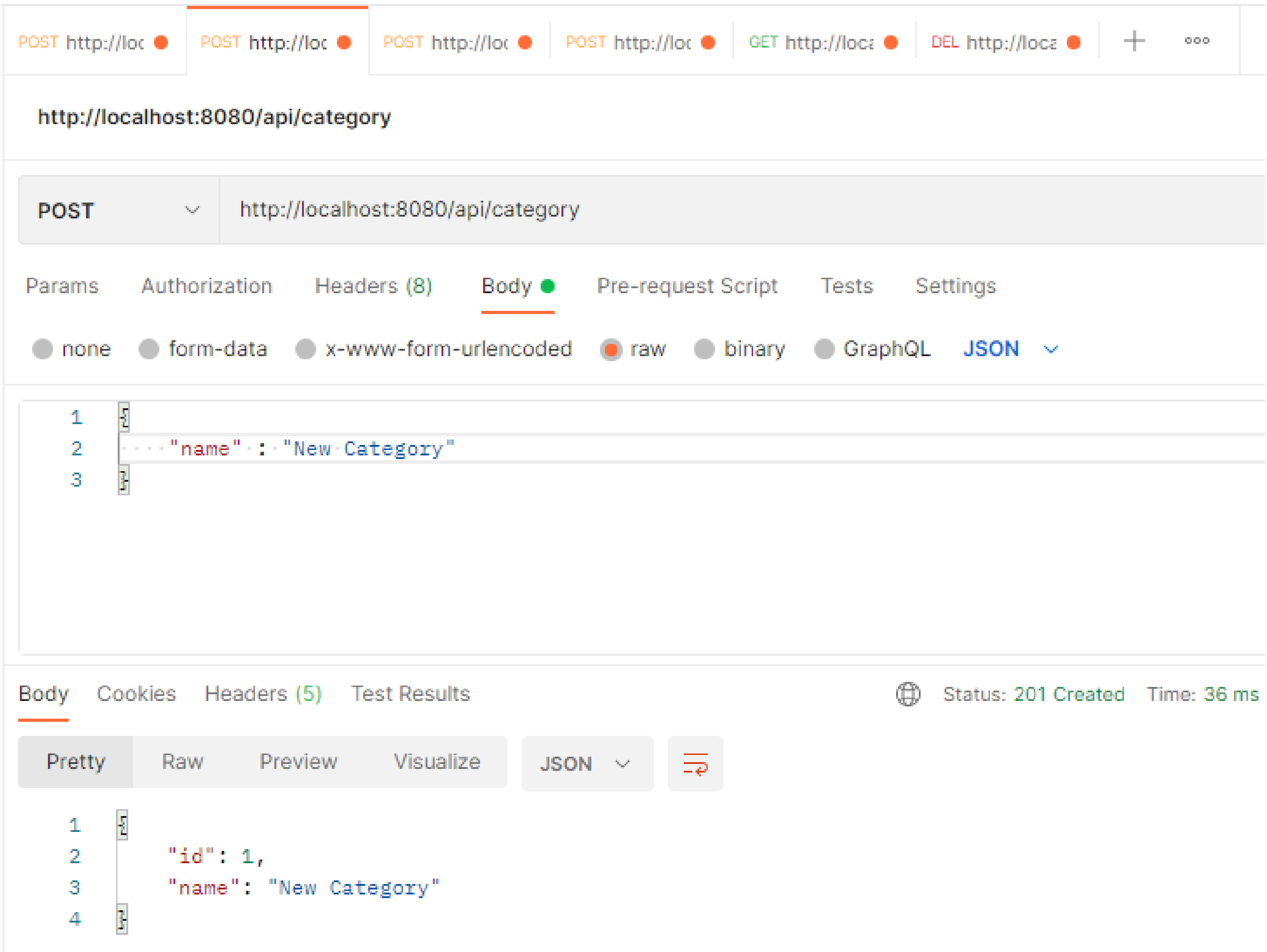
```
{
    "id": 1,
    "name": "New Publisher"
}
```

**Our new record got successfully added to Database.**



```
SELECT * FROM librarymanagement.publishers;
```

| id | name |
|---|---|
| 1 | New Publisher |
| NULL | NULL |



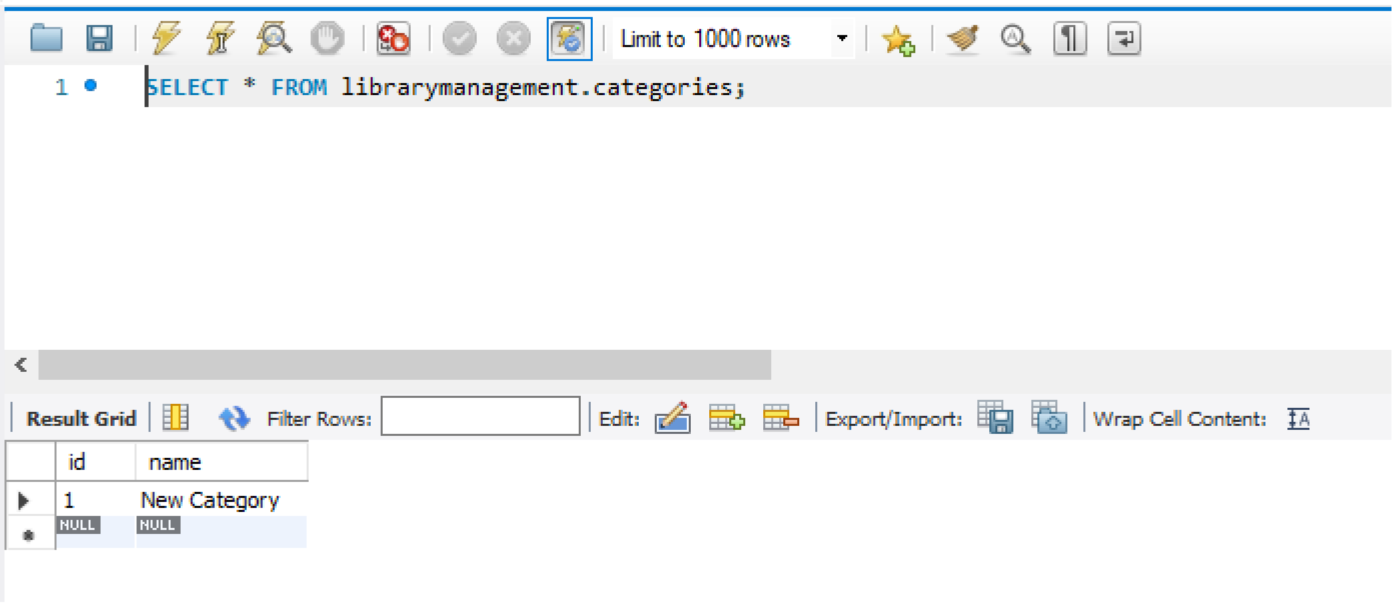
http://localhost:8080/api/category

POST http://localhost:8080/api/category

```
{
    "name" : "New Category"
}
```

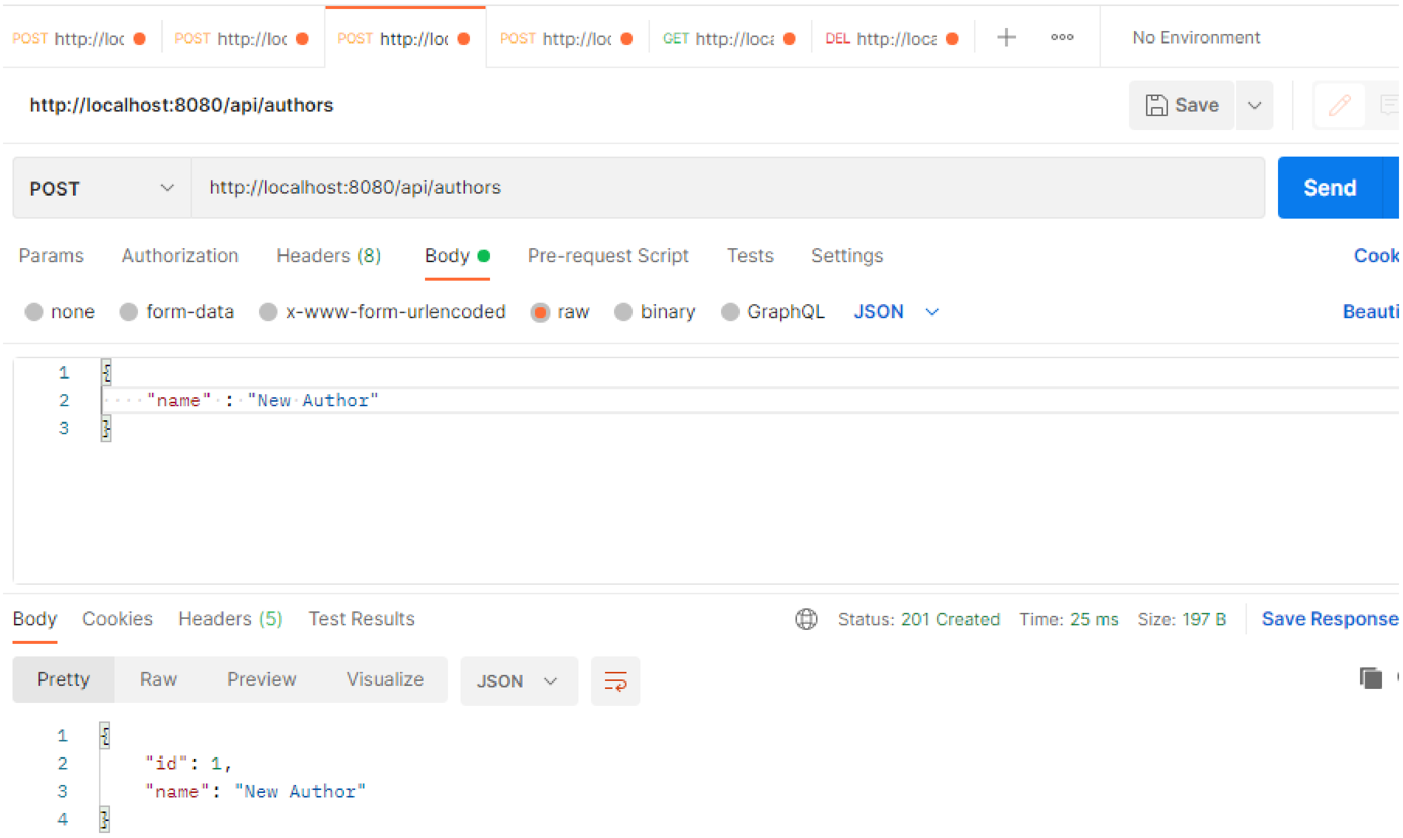
Status: 201 Created Time: 36 ms

```
{
    "id": 1,
    "name": "New Category"
}
```



```
SELECT * FROM librarymanagement.categories;
```

| id | name |
|---|---|
| 1 | New Category |
| NULL | NULL |



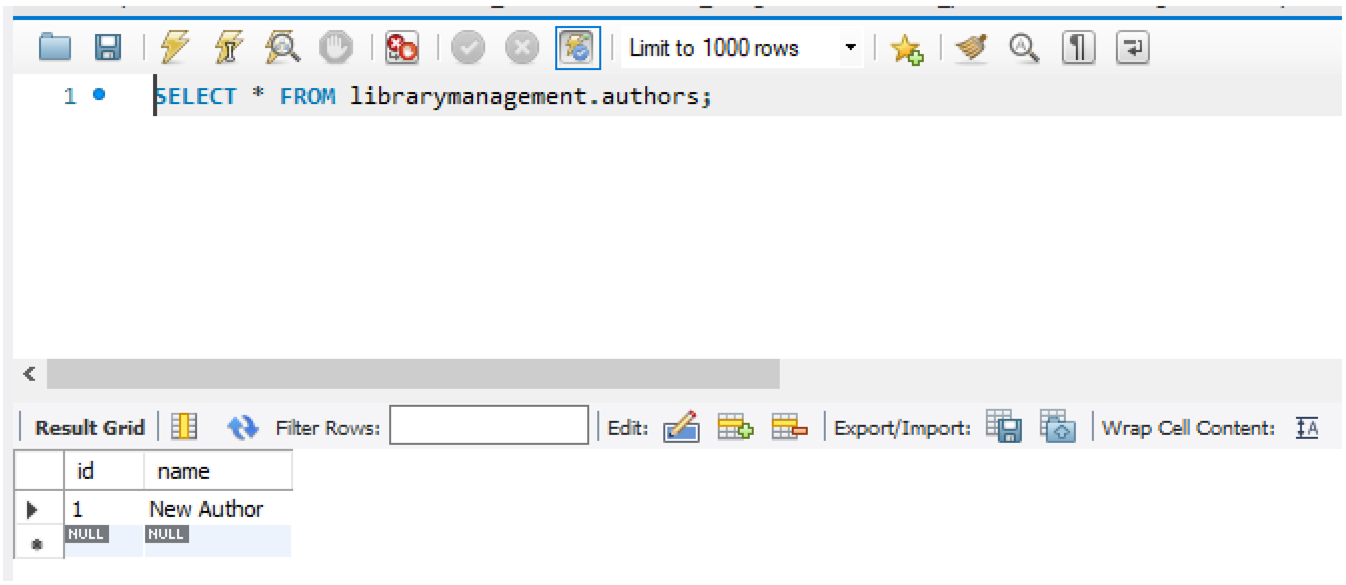
http://localhost:8080/api/authors

POST http://localhost:8080/api/authors

```
{
    "name" : "New Author"
}
```

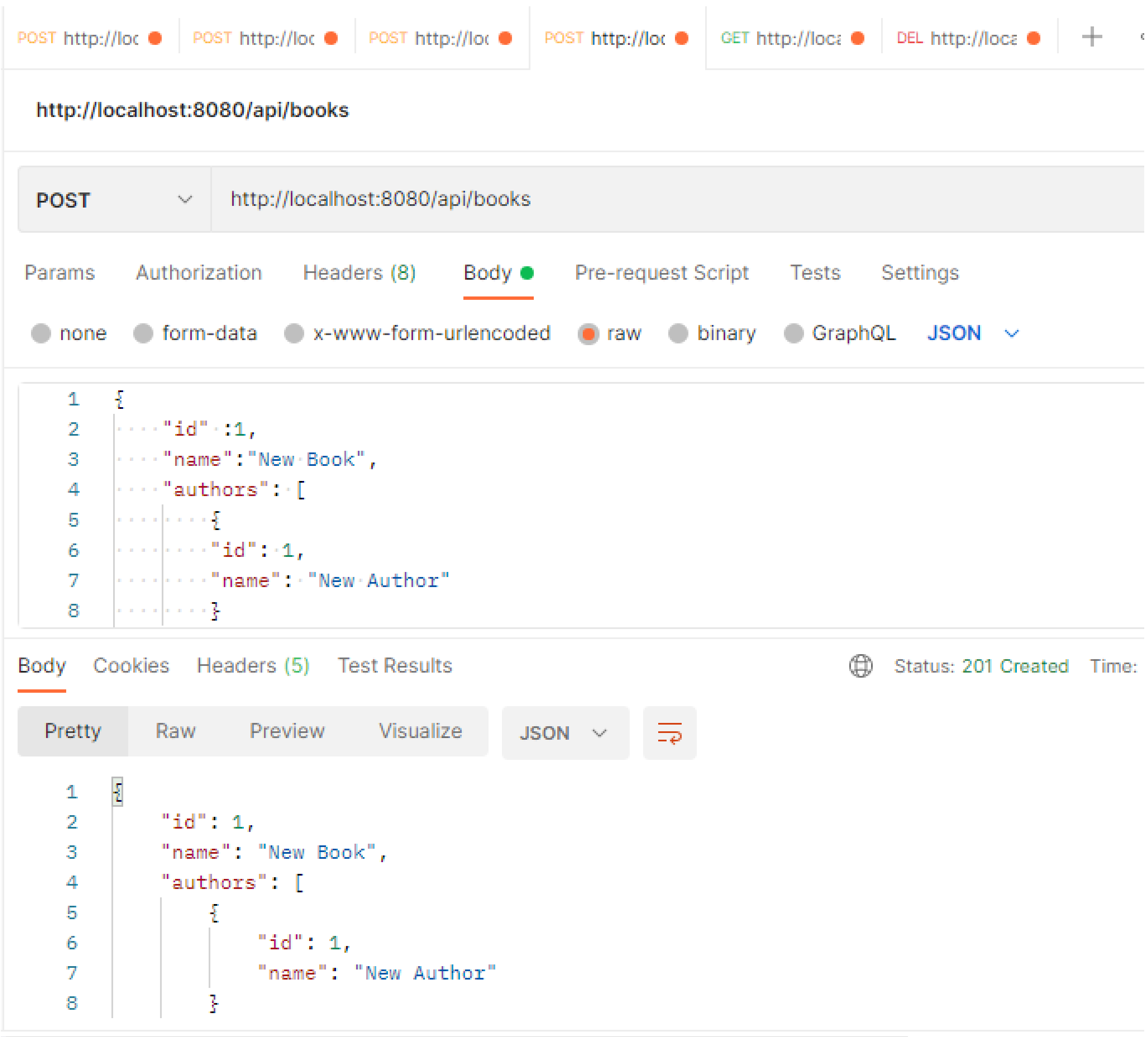
Status: 201 Created Time: 25 ms Size: 197 B

```
{
    "id": 1,
    "name": "New Author"
}
```



```
SELECT * FROM librarymanagement.authors;
```

| id | name |
|---|---|
| 1 | New Author |
| NULL | NULL |



http://localhost:8080/api/books

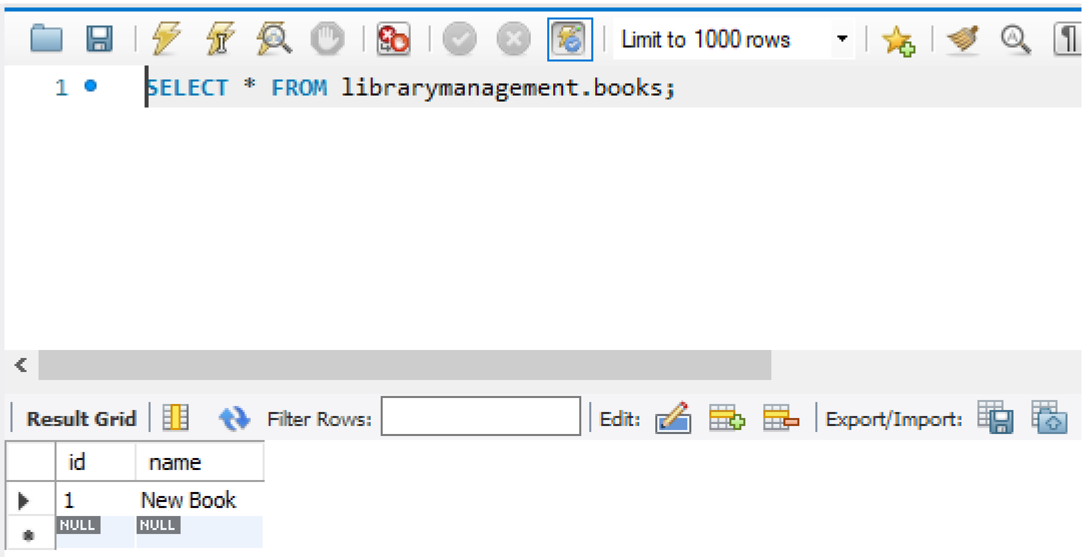
POST http://localhost:8080/api/books

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
    "id" :1,
    "name":"New Book",
    "authors": [
        {
        "id": 1,
        "name": "New Author"
        }
```

Status: 201 Created Time:

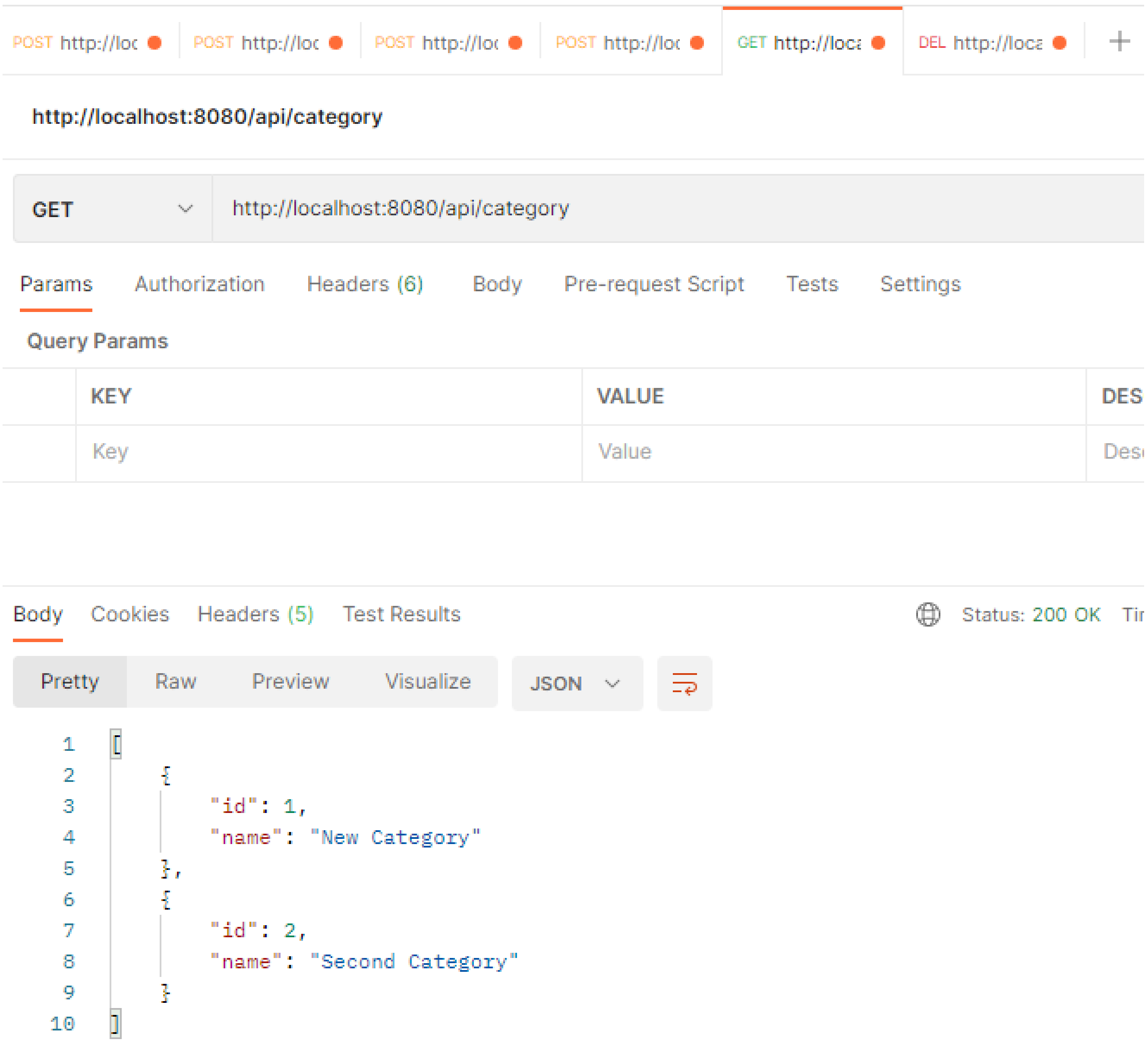
```
{
    "id": 1,
    "name": "New Book",
    "authors": [
        {
            "id": 1,
            "name": "New Author"
        }
```



```
SELECT * FROM librarymanagement.books;
```

| | id | name |
|---|---|---|
| ▶ | 1 | New Book |
| ● | NULL | NULL |

**Get Operation:**



http://localhost:8080/api/category

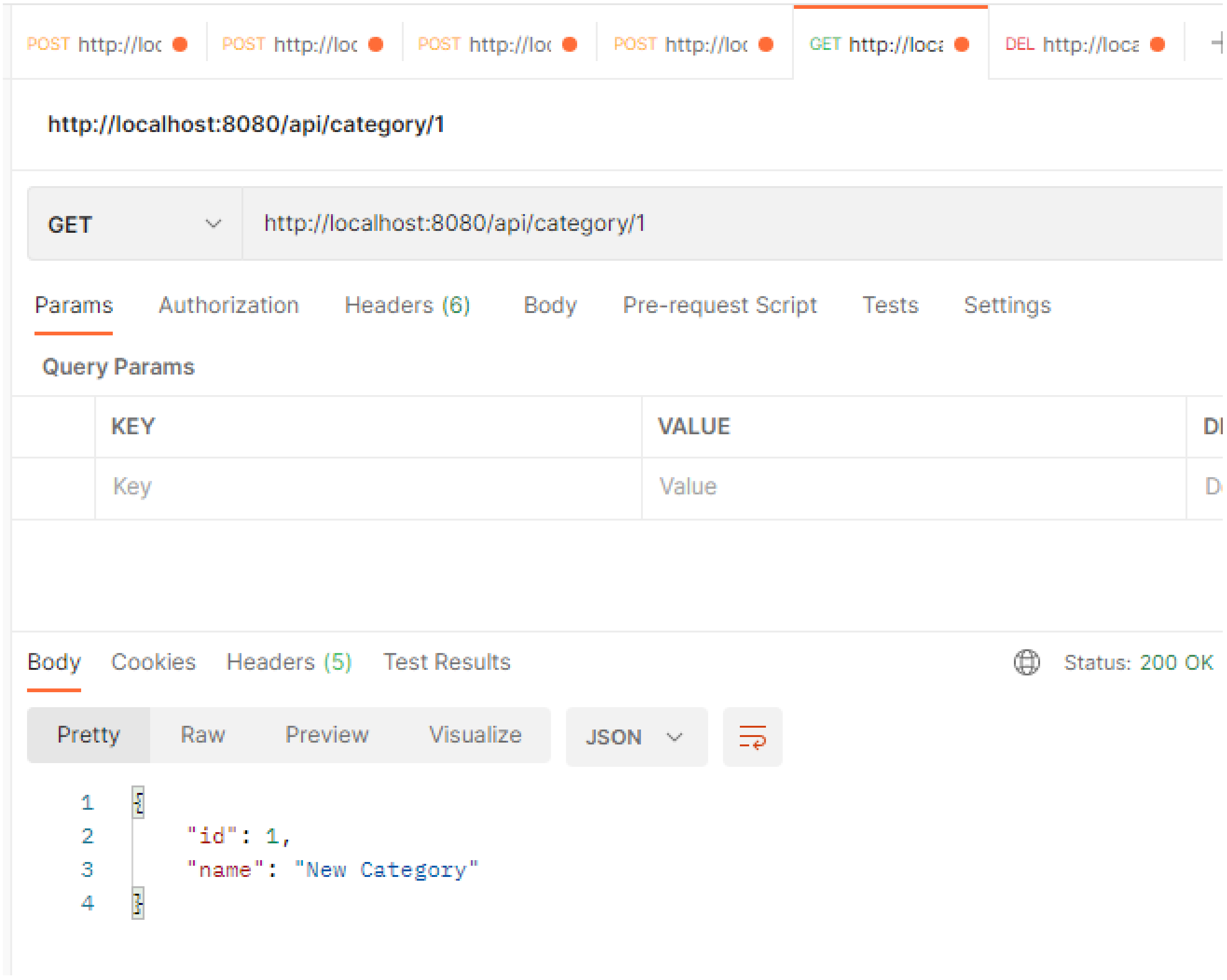
GET http://localhost:8080/api/category

| KEY | VALUE | DES |
|---|---|---|
| Key | Value | Des |

Status: 200 OK

```
[
    {
        "id": 1,
        "name": "New Category"
    },
    {
        "id": 2,
        "name": "Second Category"
    }
]
```

**Get by id:**

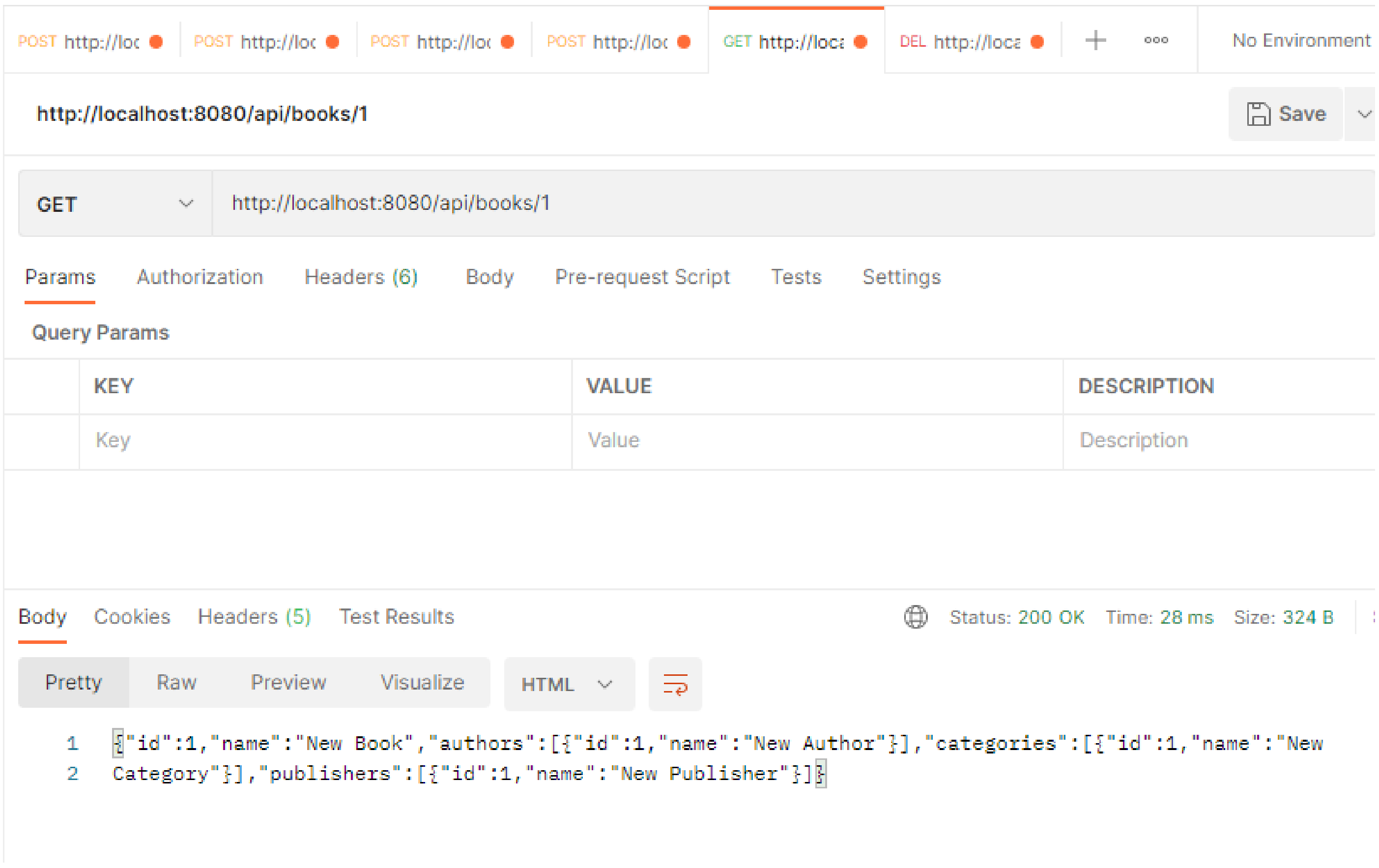


http://localhost:8080/api/category/1

GET http://localhost:8080/api/category/1

Status: 200 OK

```
{
    "id": 1,
    "name": "New Category"
}
```



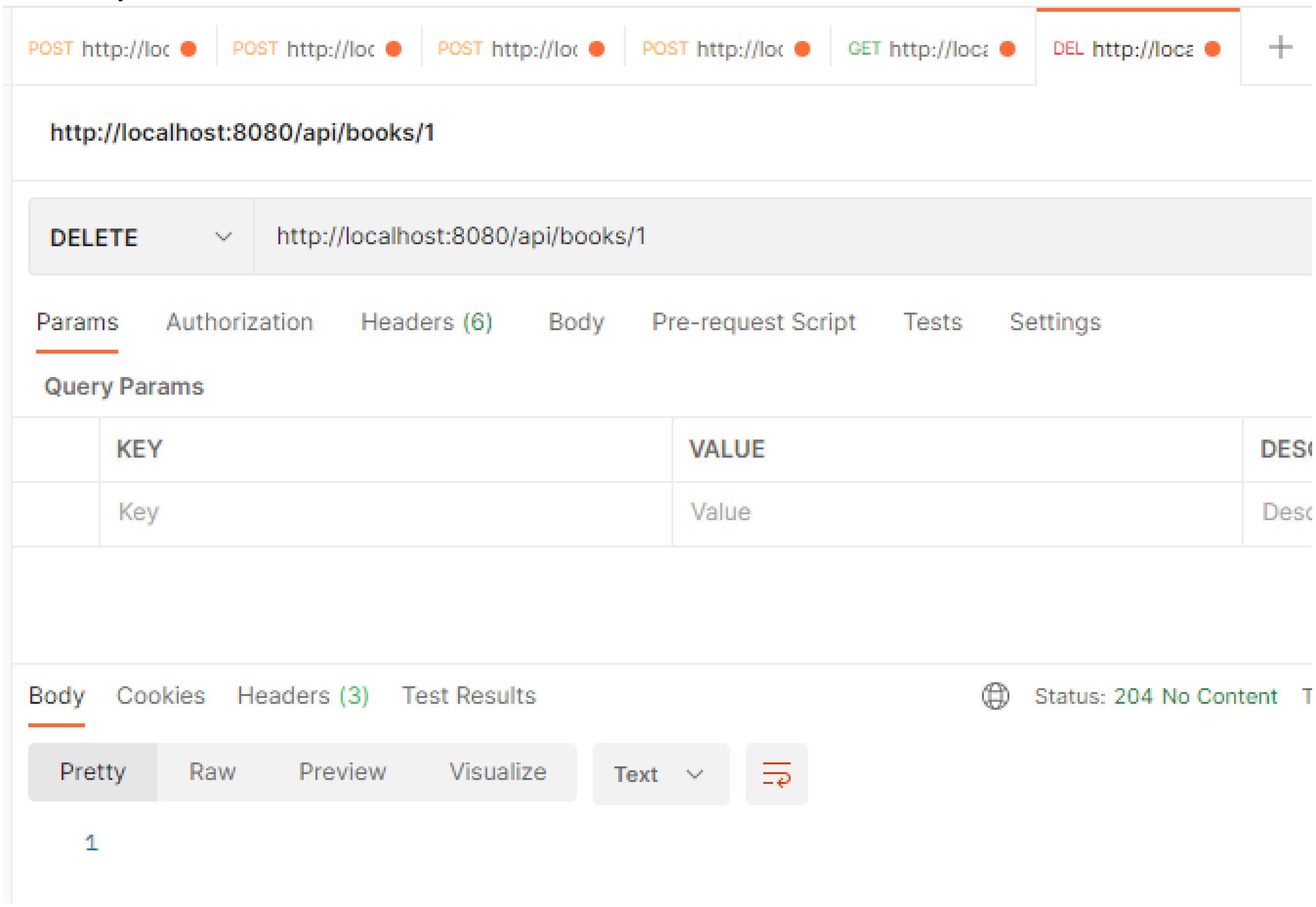
http://localhost:8080/api/books/1

GET http://localhost:8080/api/books/1

Status: 200 OK Time: 28 ms Size: 324 B

```
{"id":1,"name":"New Book","authors":[{"id":1,"name":"New Author"}],"categories":[{"id":1,"name":"New
Category"}],"publishers":[{"id":1,"name":"New Publisher"}]}
```

**Delete Operation:**



http://localhost:8080/api/books/1

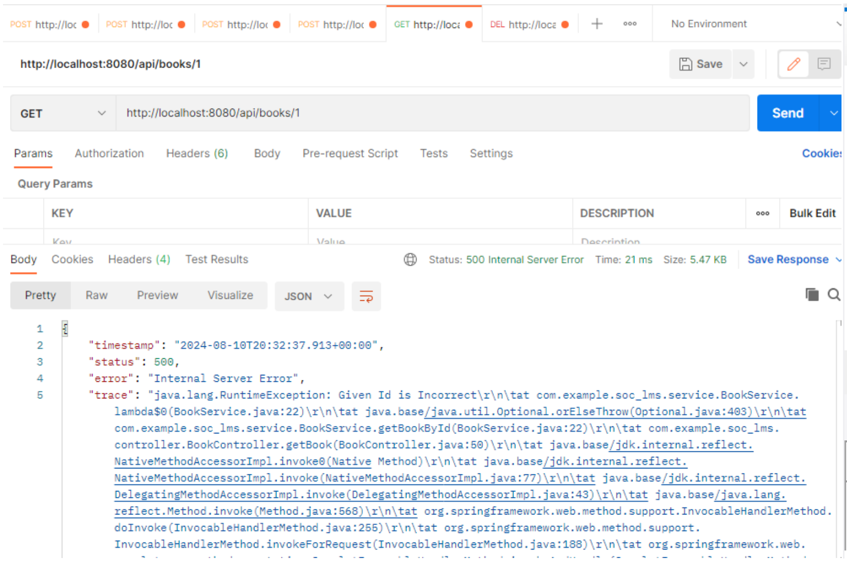
DELETE http://localhost:8080/api/books/1

| KEY | VALUE | DESC |
|---|---|---|
| Key | Value | Desc |

Status: 204 No Content

```
1
```

**Verifying the deletion in Postman and Database:**



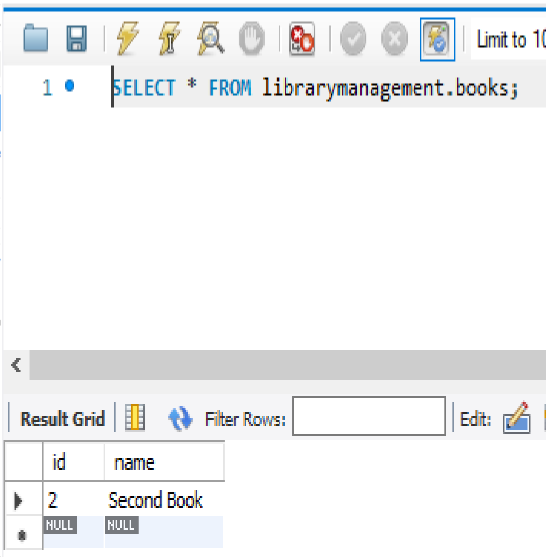
http://localhost:8080/api/books/1

GET http://localhost:8080/api/books/1

| KEY | VALUE | DESCRIPTION |
|---|---|---|
| Key | Value | Description |

Status: 500 Internal Server Error Time: 21 ms Size: 5.47 KB

```
{
    "timestamp": "2024-08-10T20:32:37.913+00:00",
    "status": 500,
    "error": "Internal Server Error",
    "trace": "java.lang.RuntimeException: Given Id is Incorrect\r\n\tat com.example.soc_lms.service.BookService.
        lambda$0(BookService.java:22)\r\n\tat java.base/java.util.Optional.orElseThrow(Optional.java:403)\r\n\tat
        com.example.soc_lms.service.BookService.getBookById(BookService.java:22)\r\n\tat com.example.soc_lms.
        controller.BookController.getBook(BookController.java:50)\r\n\tat java.base/jdk.internal.reflect.
        NativeMethodAccessorImpl.invoke0(Native Method)\r\n\tat java.base/jdk.internal.reflect.
        NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:77)\r\n\tat java.base/jdk.internal.reflect.
        DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)\r\n\tat java.base/java.lang.
        reflect.Method.invoke(Method.java:568)\r\n\tat org.springframework.web.method.support.InvocableHandlerMethod.
        doInvoke(InvocableHandlerMethod.java:255)\r\n\tat org.springframework.web.method.support.
        InvocableHandlerMethod.invokeForRequest(InvocableHandlerMethod.java:188)\r\n\tat org.springframework.web.
```



```
SELECT * FROM librarymanagement.books;
```

| id | name |
|---|---|
| 2 | Second Book |
| NULL | NULL |

**Successfully deleted.**

## Source Codes:

**Entity:**

**Author.java**

```java
package com.example.soc_lms.entity;
import java.util.List;
import com.fasterxml.jackson.annotation.JsonIgnore;
import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "authors")
public class Author {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    @ManyToMany(mappedBy = "authors", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Author() {

    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }
    public List<Book> getBooks() {
        return books;
    }
    public void setBooks(List<Book> books) {
        this.books = books;
    }
}
```

**Book.java**

```java
package com.example.soc_lms.entity;

import java.util.List;

import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.JoinTable;
import jakarta.persistence.OneToMany;
import jakarta.persistence.Table;
import jakarta.persistence.JoinColumn;

@Entity
@Table(name = "books")
public class Book {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column
    private String name;
    @OneToMany
    @JoinTable(name = "books_authors",
              joinColumns = {@JoinColumn(name = "book_id")},
              inverseJoinColumns = {@JoinColumn(name = "author_id")})
    private List<Author> authors;
    @OneToMany
    @JoinTable(name = "books_categories",
    joinColumns = {@JoinColumn(name = "book_id")},
    inverseJoinColumns = {@JoinColumn(name = "category_id")})
    private List<Category> categories;
    @OneToMany
    @JoinTable(name = "books_publishers",
    joinColumns = {@JoinColumn(name = "book_id")},
    inverseJoinColumns = {@JoinColumn(name = "publisher_id")})
    private List<Publisher> publishers;

    public Book() {

    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public List<Author> getAuthors() {
        return authors;
    }

    public void setAuthors(List<Author> authors) {
        this.authors = authors;
    }

    public List<Category> getCategories() {
        return categories;
    }

    public void setCategories(List<Category> categories) {
        this.categories = categories;
    }

    public List<Publisher> getPublishers() {
        return publishers;
    }

    public void setPublishers(List<Publisher> publishers) {
        this.publishers = publishers;
    }
}
```

**Category.java**

```java
package com.example.soc_lms.entity;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "categories")
public class Category {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
    @Column
    private String name;
    @ManyToMany(mappedBy = "categories", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Category() {

    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public List<Book> getBooks() {
        return books;
    }

    public void setBooks(List<Book> books) {
        this.books = books;
    }

}
```

**Publisher.java**

```java
package com.example.soc_lms.entity;

import java.util.List;

import com.fasterxml.jackson.annotation.JsonIgnore;

import jakarta.persistence.CascadeType;
import jakarta.persistence.Entity;
import jakarta.persistence.GeneratedValue;
import jakarta.persistence.GenerationType;
import jakarta.persistence.Id;
import jakarta.persistence.ManyToMany;
import jakarta.persistence.Table;

@Entity
@Table(name = "publishers")
public class Publisher {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;

    private String name;
    @ManyToMany(mappedBy = "publishers", cascade = CascadeType.ALL )
    @JsonIgnore
    private List<Book> books;

    public Publisher() {
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getName() {
        return name;
    }

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

    public List<Book> getBooks() {
        return books;
    }
    public void setBooks(List<Book> books) {
        this.books = books;
    }
}
```

AuthorService.java

```java
package com.example.soc_lms.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.soc_lms.entity.Author;
import com.example.soc_lms.repo.AuthorRepo;

@Service
public class AuthorService {

    @Autowired
    private AuthorRepo authorRepo;

    public List<Author> getAllAuthors(){
        return authorRepo.findAll();
    }
    public Author getAuthorById(int id) {
        return authorRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Author saveOrUpdateAuthor(Author author) {
        return authorRepo.save(author);
    }
    public void deleteAuthorById(int id) {
        authorRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        authorRepo.deleteById(id);
    }

}
```

BookService.java

```java
package com.example.soc_lms.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.soc_lms.entity.Book;
import com.example.soc_lms.repo.BookRepo;

@Service
public class BookService {

    @Autowired
    private BookRepo bookRepo;
    public List<Book> getAllBooks(){
        return bookRepo.findAll();
    }
    public Book getBookById(int id) {
        return bookRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Book saveOrUpdateBook(Book book) {
        return bookRepo.save(book);
    }
    public void deleteBookById(int id) {
        bookRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        bookRepo.deleteById(id);
    }

}
```

CategoryService.java

```
package com.example.soc_lms.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.soc_lms.entity.Category;
import com.example.soc_lms.repo.CategoryRepo;


@Service
public class CategoryService {

    @Autowired
    private CategoryRepo categoryRepo;

    public List<Category> getAllCategories(){
        return categoryRepo.findAll();
    }
    public Category getCategoryById(int id) {
        return categoryRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Category saveOrUpdateCategory(Category category) {
        return categoryRepo.save(category);
    }
    public void deleteCategoryById(int id) {
        categoryRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        categoryRepo.deleteById(id);
    }

}
```

PublisherService.java
```java
package com.example.soc_lms.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import com.example.soc_lms.entity.Publisher;
import com.example.soc_lms.repo.PublisherRepo;


@Service
public class PublisherService {

    @Autowired
    private PublisherRepo publisherRepo;

    public List<Publisher> getAllPublishers(){
        return publisherRepo.findAll();
    }
    public Publisher getPublisherById(int id) {
        return publisherRepo.findById(id)
                .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
    }
    public Publisher saveOrUpdatePublisher(Publisher publisher) {
        return publisherRepo.save(publisher);
    }
    public void deletePublisherById(int id) {
        publisherRepo.findById(id)
        .orElseThrow(() -> new RuntimeException("Given Id is Incorrect"));
        publisherRepo.deleteById(id);
    }

}
```

**application.properties**

```properties
spring.application.name=soc_lms
spring.datasource.url=jdbc:mysql://localhost:3306/LibraryManagement
spring.datasource.username=root
spring.datasource.password=admin
spring.jpa.hibernate.ddl-auto=update
```

AuthorController.java

```java
package com.example.soc_lms.controller;
import java.util.List;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
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.RestController;

import com.example.soc_lms.entity.Author;
import com.example.soc_lms.service.AuthorService;

@RestController
@RequestMapping("/api/authors")
public class AuthorController {
    @Autowired
    private AuthorService authorService;

    @GetMapping
    public ResponseEntity<List<Author>> getAllAuthors(){

        List<Author> authors = authorService.getAllAuthors();
        return ResponseEntity.ok(authors);
    }

    @GetMapping("/{id}")
    public ResponseEntity<Author> getAuthor(@PathVariable int id){
        Author author = authorService.getAuthorById(id);
        if(author == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(author);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Author> updateAuthor(@PathVariable int id, @RequestBody Author
author){
        Author existingAuthor = authorService.getAuthorById(id);
        if(existingAuthor == null) {
            return ResponseEntity.notFound().build();
        }
        author.setId(id);//Ensure the ID is set correctly
        authorService.saveOrUpdateAuthor(author);
        return ResponseEntity.ok(author);
    }
    @PostMapping
    public ResponseEntity<Author> saveAuthor(@RequestBody Author author){
        Author createAuthor = authorService.saveOrUpdateAuthor(author);
        return ResponseEntity.status(HttpStatus.CREATED).body(createAuthor);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteAuthor(@PathVariable int id){
        authorService.deleteAuthorById(id);
        return ResponseEntity.noContent().build();
    }

}
```

BookController.java

```java
package com.example.soc_lms.controller;

import java.util.ArrayList;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
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.RestController;

import com.example.soc_lms.entity.Author;
import com.example.soc_lms.entity.Book;
import com.example.soc_lms.entity.Category;
import com.example.soc_lms.entity.Publisher;
import com.example.soc_lms.service.AuthorService;
import com.example.soc_lms.service.BookService;
import com.example.soc_lms.service.CategoryService;
import com.example.soc_lms.service.PublisherService;

@RestController
@RequestMapping("/api/books")
public class BookController {

    @Autowired
    private BookService bookService;
    @Autowired
    private AuthorService authorService;
    @Autowired
    private CategoryService categoryService;
    @Autowired
    private PublisherService publisherService;

    @GetMapping
    public ResponseEntity<List<Book>> getAllBooks(){

        List<Book> books = bookService.getAllBooks();
        return ResponseEntity.ok(books);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Book> getBook(@PathVariable int id){
        Book book = bookService.getBookById(id);
        if(book == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(book);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Book> updateBook(@PathVariable int id, @RequestBody Book book){
        Book existingBook = bookService.getBookById(id);
        if(existingBook == null) {
            return ResponseEntity.notFound().build();
        }
        List<Author> authors = new ArrayList<Author>();
        for (Author author : book.getAuthors()) {
            Author foundauthor = authorService.getAuthorById(author.getId());
            if(foundauthor == null) {
                return ResponseEntity.notFound().build();
                }
            authors.add(foundauthor);
        }
        book.setAuthors(authors);
        List<Category> categories = new ArrayList<Category>();
        for (Category category : book.getCategories()) {
            Category foundcategory = categoryService.getCategoryById(category.getId());
            if(foundcategory == null) {
                return ResponseEntity.notFound().build();
                }
            categories.add(foundcategory);
        }
        book.setCategories(categories);
        List<Publisher> publishers = new ArrayList<Publisher>();
        for (Publisher publisher : book.getPublishers()) {
            Publisher foundpublisher =
publisherService.getPublisherById(publisher.getId());
            if(foundpublisher == null) {
                return ResponseEntity.notFound().build();
                }
            publishers.add(foundpublisher);
        }
        book.setCategories(categories);
        book.setId(id);//Ensure the ID is set correctly
        bookService.saveOrUpdateBook(book);
        return ResponseEntity.ok(book);
    }
    @PostMapping
    public ResponseEntity<Book> saveBook(@RequestBody Book book){
        List<Author> authors = new ArrayList<Author>();
        for (Author author : book.getAuthors()) {
            Author foundauthor = authorService.getAuthorById(author.getId());
            if(foundauthor == null) {
                return ResponseEntity.notFound().build();
                }
            authors.add(foundauthor);
        }
        book.setAuthors(authors);
        List<Category> categories = new ArrayList<Category>();
        for (Category category : book.getCategories()) {
            Category foundcategory = categoryService.getCategoryById(category.getId());
            if(foundcategory == null) {
                return ResponseEntity.notFound().build();
                }
            categories.add(foundcategory);
        }
        book.setCategories(categories);
        List<Publisher> publishers = new ArrayList<Publisher>();
        for (Publisher publisher : book.getPublishers()) {
            Publisher foundpublisher =
publisherService.getPublisherById(publisher.getId());
            if(foundpublisher == null) {
                return ResponseEntity.notFound().build();
                }
            publishers.add(foundpublisher);
        }
        book.setCategories(categories);
        Book createBook = bookService.saveOrUpdateBook(book);
        return ResponseEntity.status(HttpStatus.CREATED).body(createBook);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteBook(@PathVariable int id){
        bookService.deleteBookById(id);
        return ResponseEntity.noContent().build();
    }
}
```

CategoryController.java

```
package com.example.soc_lms.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
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;
```

```java
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.example.soc_lms.entity.Category;
import com.example.soc_lms.service.CategoryService;

@RestController
@RequestMapping("/api/category")
public class CategoryController {

    @Autowired
    private CategoryService categoryService;

    @GetMapping
    public ResponseEntity<List<Category>> getAllCategories(){

        List<Category> categories = categoryService.getAllCategories();
        return ResponseEntity.ok(categories);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Category> getCategory(@PathVariable int id){
        Category category = categoryService.getCategoryById(id);
        if(category == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(category);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Category> updateCategory(@PathVariable int id, @RequestBody
Category category){
        Category existingCategory = categoryService.getCategoryById(id);
        if(existingCategory == null) {
            return ResponseEntity.notFound().build();
        }
        category.setId(id);//Ensure the ID is set correctly
        categoryService.saveOrUpdateCategory(category);
        return ResponseEntity.ok(category);
    }
    @PostMapping
    public ResponseEntity<Category> saveCategory(@RequestBody Category category){
        Category createCategory = categoryService.saveOrUpdateCategory(category);
        return ResponseEntity.status(HttpStatus.CREATED).body(createCategory);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deleteCategory(@PathVariable int id){
        categoryService.deleteCategoryById(id);
        return ResponseEntity.noContent().build();
    }
}
```

PublisherController.java

```java
package com.example.soc_lms.controller;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
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.RestController;

import com.example.soc_lms.entity.Publisher;
import com.example.soc_lms.service.PublisherService;

@RestController
@RequestMapping("/api/publisher")
public class PublisherController {
    @Autowired
    private PublisherService publisherService;

    @GetMapping
    public ResponseEntity<List<Publisher>> getAllPublishers(){

        List<Publisher> publishers = publisherService.getAllPublishers();
        return ResponseEntity.ok(publishers);

    }

    @GetMapping("/{id}")
    public ResponseEntity<Publisher> getPublisher(@PathVariable int id){
        Publisher publisher = publisherService.getPublisherById(id);
        if(publisher == null) {
            return ResponseEntity.notFound().build();
        }
        return ResponseEntity.ok(publisher);
    }
    @PutMapping("/{id}")
    public ResponseEntity<Publisher> updatePublisher(@PathVariable int id, @RequestBody
Publisher publisher){
        Publisher existingPublisher = publisherService.getPublisherById(id);
        if(existingPublisher == null) {
            return ResponseEntity.notFound().build();
        }
        publisher.setId(id);//Ensure the ID is set correctly
        publisherService.saveOrUpdatePublisher(publisher);
        return ResponseEntity.ok(publisher);
    }
    @PostMapping
    public ResponseEntity<Publisher> savePublisher(@RequestBody Publisher publisher){
        Publisher createPublisher = publisherService.saveOrUpdatePublisher(publisher);
        return ResponseEntity.status(HttpStatus.CREATED).body(createPublisher);
    }
    @DeleteMapping("/{id}")
    public ResponseEntity<Void> deletePublisher(@PathVariable int id){
        publisherService.deletePublisherById(id);
        return ResponseEntity.noContent().build();
    }
}
```

Pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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
https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>3.3.2</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <groupId>com.example</groupId>
    <artifactId>soc_lms</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>soc_lms</name>
    <description>Demo project for Spring Boot</description>
    <url/>
    <licenses>
        <license/>
    </licenses>
    <developers>
        <developer/>
    </developers>
    <scm>
        <connection/>
        <developerConnection/>
        <tag/>
        <url/>
    </scm>
    <properties>
        <java.version>17</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>

        <dependency>
            <groupId>com.mysql</groupId>
            <artifactId>mysql-connector-j</artifactId>
            <scope>runtime</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <scope>runtime</scope>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>com.h2database</groupId>
            <artifactId>h2</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>

</project>
```