Spring MVC CRUD Demo of BookStore

- Frontend: Thymeleaf, HTML, CSS, JS, Bootstrap
- Backend: Spring Boot, Spring MVC, Spring Data JPA, Hibernate
- Database: PostgreSQL

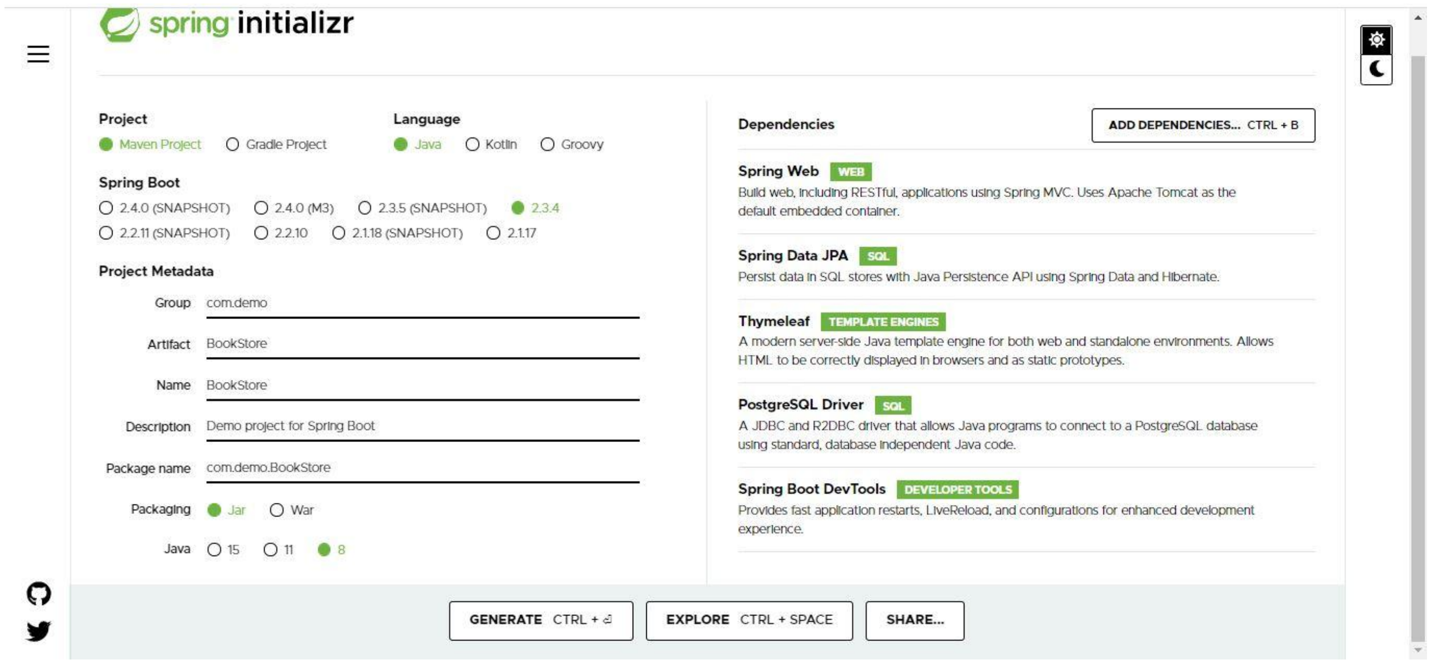
Pre-requisites:

1. Java 8/11/14
2. Eclipse IDE / Spring Tool Suite 4
3. PostgreSQL / MySQL
4. Any Database GUI tool like DBeaver / SQLyog etc.

STEPS

Step 1: Visit https://start.spring.io/ & select options of the left side as shown in the below image. Add 5 dependencies on the right side.

1. Spring Web
2. Spring Data JPA
3. Thymeleaf
4. PostgreSQL / MySQL Driver
5. Spring Boot DevTools (Optional)



Project: [✓ Maven Project] Gradle Project

Language: [✓ Java] Kotlin Groovy

Spring Boot: 2.4.0 (SNAPSHOT) 2.4.0 (M3) 2.3.5 (SNAPSHOT) [✓ 2.3.4] 2.2.11 (SNAPSHOT) 2.2.10 2.1.18 (SNAPSHOT) 2.1.17

Project Metadata

Group	com.demo
Artifact	BookStore
Name	BookStore
Description	Demo project for Spring Boot
Package name	com.demo.BookStore
Packaging	[✓ Jar] War
Java	15 11 [✓ 8]

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.

Thymeleaf TEMPLATE ENGINES
A modern server-side Java template engine for both web and standalone environments. Allows HTML to be correctly displayed in browsers and as static prototypes.

PostgreSQL Driver SQL
A JDBC and R2DBC driver that allows Java programs to connect to a PostgreSQL database using standard, database independent Java code.

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

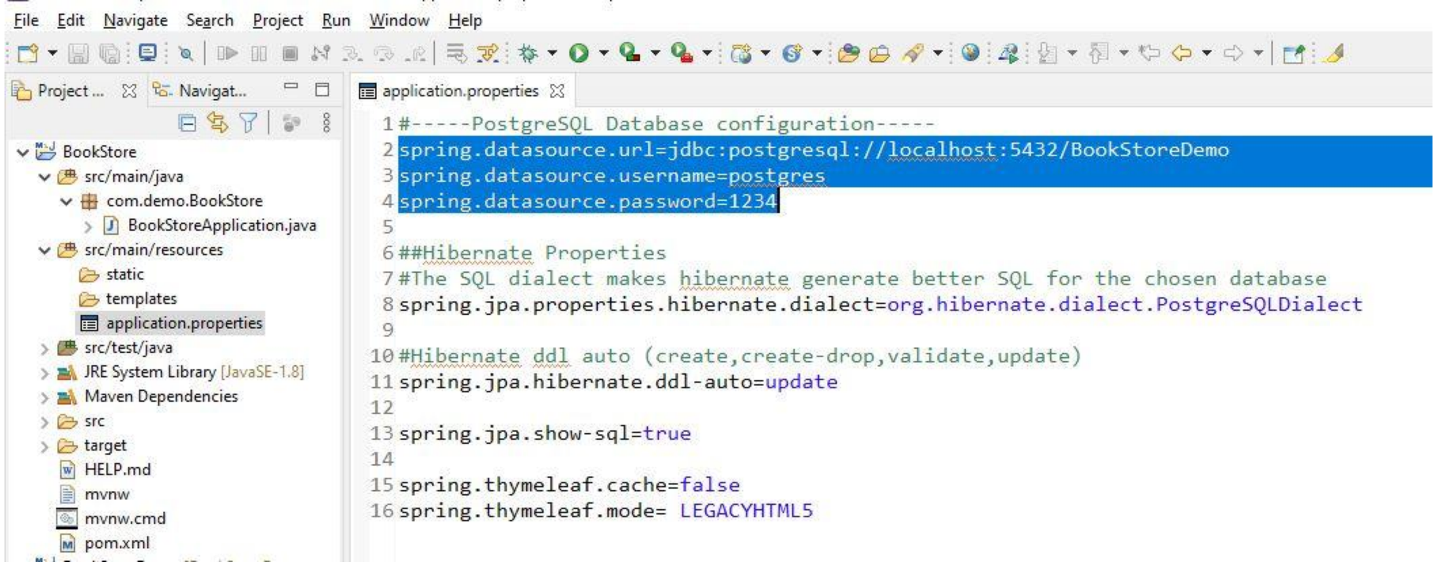
- Press the Generate button and the ZIP file will be downloaded automatically.
- Extract that file.

- Now open **Eclipse** and go to **File->Import->search “Existing Maven Projects” and select it->Next->Browse and select the Extracted folder->Finish.**
- It will take some time to download the dependencies. Make sure your internet connection is stable.

Step 2: Database setup

- Go to **src/main/resources**, open the **application.properties** file and paste the below code.
- For MySQL database, go to this file and find your appropriate code of configuration.
  https://docs.google.com/document/d/1HfFVYrOw4KYmNxKphBbruQhZla4cbnu2rpIbOh3ZGE8

```
#-----PostgreSQL Database configuration-----
spring.datasource.url=jdbc:postgresql://localhost:5432/BookStoreDemo
spring.datasource.username=postgres
spring.datasource.password=1234

##Hibernate Properties
#The SQL dialect makes hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

#Hibernate ddl auto (create,create-drop,validate,update)
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.thymeleaf.cache=false
spring.thymeleaf.mode= LEGACYHTML5
```

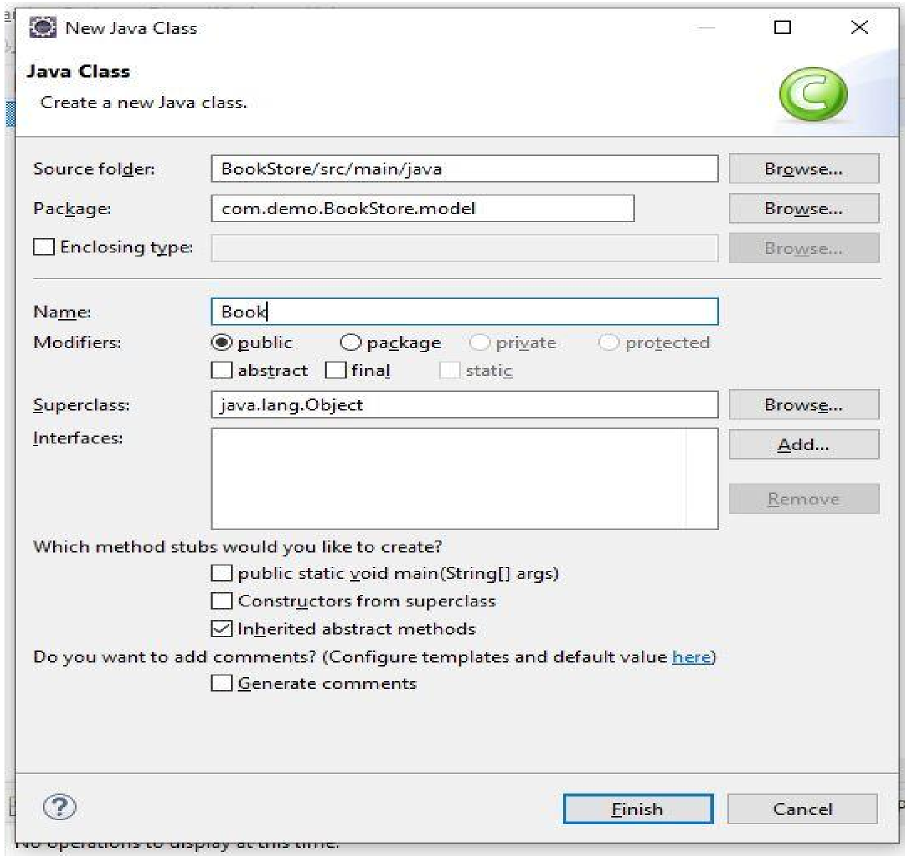


```
#-----PostgreSQL Database configuration-----
spring.datasource.url=jdbc:postgresql://localhost:5432/BookStoreDemo
spring.datasource.username=postgres
spring.datasource.password=1234

##Hibernate Properties
#The SQL dialect makes hibernate generate better SQL for the chosen database
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDialect

#Hibernate ddl auto (create,create-drop,validate,update)
spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.thymeleaf.cache=false
spring.thymeleaf.mode= LEGACYHTML5
```

- Create a database named “BookStoreDemo” & Change the highlighted values according to your database connection.

Step 3: Creating an entity.

- Right-click on **com.demo.BookStore** in src/main/java & select new->class.
- Package: com.demo.BookStore**.model**
  Name: **Book**



Source folder: BookStore/src/main/java
Package: com.demo.BookStore.model
Enclosing type: ☐
Name: Book
Modifiers: [✓ public] package private protected
☐ abstract ☐ final ☐ static
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

- Click on Finish
- Now paste the below code in it.

```
@Entity
public class Book {
        @Id
        private long isbn;
        private String title;
        private double price;
        private int stock;
        public long getIsbn() {
                return isbn;
        }
        public void setIsbn(long isbn) {
                this.isbn = isbn;
        }
        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 getStock() {
                return stock;
        }
        public void setStock(int stock) {
                this.stock = stock;
        }
}
```

Step 4: Creating a Controller.

- Right-click on **com.demo.BookStore** in src/main/java & select new->class.
- Package: com.demo.BookStore**.controller**
  Name: **BookController**
- Click on Finish
- Now paste the below code in it.

```
@Controller
@RequestMapping("/book")
public class BookController {

	@Autowired
	private BookService bookService;

	@GetMapping("/addBook")
	public String addBook(Model model) {
		Book book = new Book();
		model.addAttribute("book", book);
		return "addBook";
	}

	@PostMapping("/addBook")
	public String addBookPost(@ModelAttribute("book") Book book, Model
model,@ModelAttribute("isbn") long isbn) {

		if(bookService.getBook(isbn).orElse(null)!=null) {
			model.addAttribute("isbnExists", true);
			return "addBook";
		}
		bookService.saveBook(book);
		return "redirect:bookList";
	}

	@GetMapping("/bookList")
	public String bookList(Model model)
	{
		List<Book> bookList = bookService.getAllBooks();
		model.addAttribute("bookList", bookList);
		return "bookList";
	}
}
```

- Use ctrl+shift+o to fix the imports.

Step 5: Creating a BookService Interface and a BookServiceImpl class that implements it.

- Right-click on **com.demo.BookStore** in src/main/java & select new->**interface**.
- Package: com.demo.BookStore**.service**
  Name: **BookService**
- Click on Finish
- Now paste the below code in it.

```
    public interface BookService {
    void saveBook(Book b);
    List<Book> getAllBooks();
    Optional<Book> getBook(Long isbn);
}
```

- Use ctrl+shift+o to fix the imports.

- Right-click on **com.demo.BookStore.service** in src/main/java & select new->**class**.
- Package: com.demo.BookStore**.service.impl**
  Name: **BookServiceImpl**
  Interfaces: **Add-> BookService->OK**
- Click on Finish
- Now paste the below code in it.

```
@Service
public class BookServiceImpl implements BookService {

    @Autowired
    private BookRepository bookRepository;

    @Override
    public void saveBook(Book b) {
        bookRepository.save(b);
    }
```

```
        @Override
        public List<Book> getAllBooks() {
                return (List<Book>) bookRepository.findAll();
        }

        @Override
        public Optional<Book> getBook(Long isbn) {
                return bookRepository.findById(isbn);
        }

    }
```

Step 6: Creating a BookRepository interface.

- Right-click on **com.demo.BookStore.repository** in src/main/java & select new->**interface**.
- Package: com.demo.BookStore**.repository**
  Name: **BookRepository**
- Click on Finish & paste the below code.

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

Step 7: Creating an index.html view.

- Go to src/main/resources->templates (right click on it) ->new->File->**index.html**->Finish.
- Paste the below line. It is just for redirecting purpose.

  ```
  <meta http-equiv="refresh" content="0; URL='book/bookList'">
  ```

- Save and close it.

Step 8: Creating 2 more views.

1. **bookList.html**
    - templates (right click on it) ->new->File->**bookList.html**->Finish.
    - Paste the below code.

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
     <!-- Required meta tags -->
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

    <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/font-awesome/4.7.0/css/font-awesome.min.c
ss">
        <title>Bookstore Demo</title>
</head>

<body>

<!-- <div class="container"> -->
<div align="center">

        <hr/><h2>All Books From Database</h2><hr/>
        <div class="container">
        <!-- Add Book Button -->
        <a th:href="@{/book/addBook}" class="btn btn-primary float-left btn-md mb-3">
                Add New Book
        </a>



                <table class="table table-bordered table-responsive-md">
                <thead class="thead-dark">
                        <tr>
                                <th>ISBN</th>
                                <th>Title</th>
```

```
                    <th>Price</th>
                    <th>Stock</th>
                    <th>Edit</th>
                    <th>Delete</th>
                </tr>
            </thead>

            <tbody>
                <tr th:each="tempBook : ${bookList}">

                    <td><a
th:href="@{/book/bookInfo?isbn=}+${tempBook.isbn}"><span
th:text="${tempBook.isbn}"></span></a></td>
                    <td th:text="${tempBook.title}" />
                    <td th:text="${tempBook.price}" />
                    <td th:text="${tempBook.stock}" />



                    <!-- Update button/link -->
                    <td>
                        <a
th:href="@{/book/updateBook?isbn=}+${tempBook.isbn}" class="btn btn-success
btn-sm">
                            <i class="fa fa-pencil-square-o"
aria-hidden="true"></i>
                        </a>
                    </td>

                    <!-- Delete button/link -->
                    <td>
                        <a
th:href="@{/book/deleteBook?isbn=}+${tempBook.isbn}" class="btn btn-danger
btn-sm"
                                onclick="if(!(confirm('Are you sure you want
to delete this book?'))) return false">
                            <i class="fa fa-trash" aria-hidden="true"></i>
                        </a>
                    </td>
                    </tr>
            </tbody>
        </table>
</div>
</div>

</body>
</html>
```

----

2. **addBook.html**
    - templates (right click on it) ->new->File->**addBook.html**->Finish.
    - Paste the below code.

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">

<head>
    <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

   <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

        <title>Add Book</title>
</head>

<body>

        <div class="container">
                <hr/><h1>New Book</h1><hr/>

                <form th:action="@{/book/addBook}" th:object="${book}"
method="POST">

                        <div class="form-group">
```

```
                    <label for="isbn">ISBN:</label>
                    <span style="color: red; font-weight:bold;"
th:if="${isbnExists}">ISBN already exists.</span>
                    <input type="text" th:field="*{isbn}" id="isbn"
class="form-control mb-4 col-4"
                         placeholder="Enter ISBN" required/>
               </div>

               <div class="form-group">
                    <label for="title">Title:</label>
                    <input type="text" th:field="*{title}" id="title" class="form-control
mb-4 col-4"
                              placeholder="Enter Book title" required/>
               </div>

               <div class="form-group">
                    <label for="price">Price:</label>
                    <input type="number" th:field="*{price}" id="price"
class="form-control mb-4 col-4"
                         min="1" step="0.01" placeholder="Enter Price" required/>
               </div>

               <div class="form-group">
                    <label for="stock">Stock:</label>

                    <input type="number" th:field="*{stock}" id="stock"
class="form-control mb-4 col-4"
                         min="1" placeholder="Enter Stock" required/>
               </div>



               <button type="submit" class="btn btn-success col-2">Add
Book</button>

          </form>

          <hr>
          <a th:href="@{/book/bookList}">Back to Books List</a>
```

```
        <hr>
    </div>
</body>

</html>
```

Step 9: Run and check is it working or not.

- Go to src/main/java->com.demo.BookStore->BookStoreApplication(right click)->Run As->Java Application.
- Open a browser and visit http://localhost:8080/

Step 10: Implementing bookInfo feature.

- Paste the below code in **BookController.java**

```
@GetMapping("/bookInfo")
public String bookInfo(@RequestParam("isbn") Long isbn, Model model) {
        model.addAttribute("book", bookService.getBook(isbn).orElse(null));

        return "bookInfo";
}
```

- templates (right click on it) ->new->File->**bookInfo.html**->Finish.
- Paste the below code.

```
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

   <!-- Bootstrap CSS -->
```

```
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">

        <title>Book Information</title>
</head>

<body>
        <div class="container">
                <hr/><h1>Book Details</h1><hr/>

                                <div>
                                        <p><strong>ISBN: </strong><span
th:text="${book.isbn}"></span></p>
                                        <p><strong>Title: </strong><span
th:text="${book.title}"></span></p>
                                        <p><strong>Price: </strong><span
th:text="${book.price}"></span></p>
                                        <p><strong>Stock: </strong><span
th:text="${book.stock}"></span></p>
                                </div>

                <hr>
                <a th:href="@{/book/bookList}">Back to Books List</a>
                <hr>
        </div>
</body>
</html>
```

Step 11: Implementing Update Book feature.

- Paste the below code in **BookController.java**

```
@GetMapping("/updateBook")
public String updateBook(@RequestParam("isbn") Long isbn, Model model) {
        model.addAttribute("book", bookService.getBook(isbn).orElse(null));
        return "updateBook";
}
```

```
        @PostMapping("/updateBook")
        public String updateBookPost(@ModelAttribute("book") Book book) {
                bookService.saveBook(book);
                return "redirect:bookList";
        }
```

- templates (right click on it) ->new->File->**updateBook.html**->Finish.
- Paste the below code.

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <!-- Required meta tags -->
   <meta charset="utf-8">
   <meta name="viewport" content="width=device-width, initial-scale=1,
shrink-to-fit=no">

   <!-- Bootstrap CSS -->
        <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.0/css/bootstrap.min.css">
        <title>Update Books</title>
</head>

<body>

        <div class="container">
                <hr/><h1>Update Book</h1><hr/>

                <form th:action="@{/book/updateBook}" th:object="${book}"
method="POST">

                        <!-- Add hidden form field to handle update -->
                        <input type="hidden" th:field="*{isbn}"/>

                        <div class="form-group">
                                <label for="isbn">ISBN:</label>
```

```
                    <input type="text" th:field="*{isbn}" id="isbn"
class="form-control mb-4 col-4"
                                placeholder="Update Book ISBN" disabled/>
            </div>

            <div class="form-group">
                    <label for="title">Title:</label>
                    <input type="text" th:field="*{title}" id="title" class="form-control
mb-4 col-4"
                                placeholder="Update Book title" required/>
            </div>

            <div class="form-group">
                    <label for="price">Price:</label>
                    <input type="number" th:field="*{price}" id="price"
class="form-control mb-4 col-4"
                            min="1" step="0.01" placeholder="Update Price"
required/>
            </div>

            <div class="form-group">
                    <label for="stock">Stock:</label>

                    <input type="number" th:field="*{stock}" id="stock"
class="form-control mb-4 col-4"
                             min="1" placeholder="Update Stock" required/>
            </div>



            <button type="submit" class="btn btn-success col-2">Update
Book</button>

        </form>

        <hr>
        <a th:href="@{/book/bookList}">Back to Books List</a>

        <hr>
        </div>
</body>
</html>
```

Step 12: Implementing Delete Book feature.

- Paste the below code in **BookController.java** class.

  ```
  @GetMapping("/deleteBook")
  public String deleteBook(@RequestParam("isbn") Long isbn) {
          bookService.deleteBook(isbn);
          return "redirect:bookList";
  }
  ```

- Paste the below line in the **BookService.java** interface.
  ```
          void deleteBook(Long isbn);
  ```

- Paste the below code in **BookServiceImpl.java** class.

  ```
  @Override
  public void deleteBook(Long isbn) {
          bookRepository.deleteById(isbn);
  }
  ```
