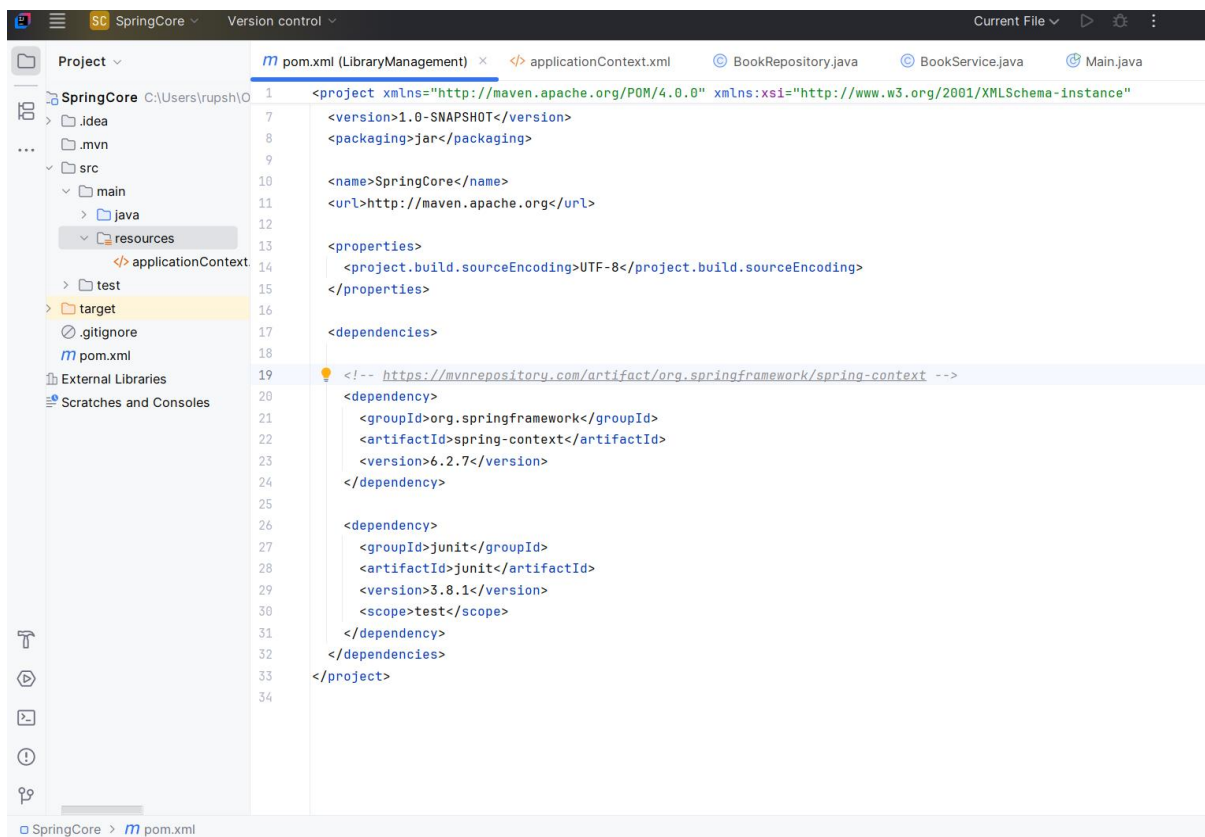
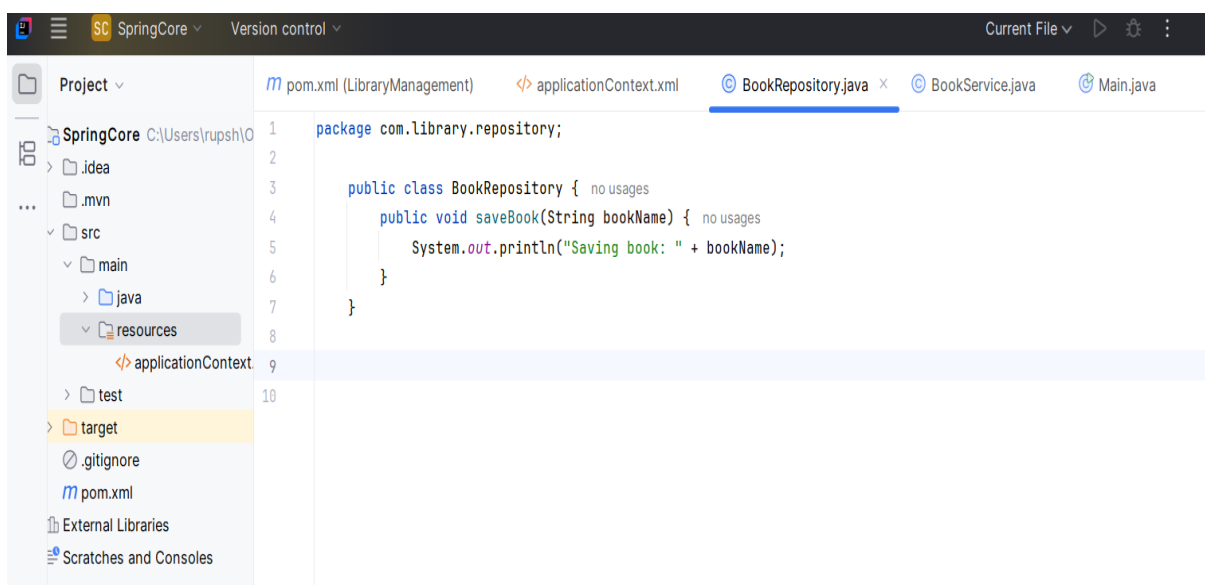


Exercise 1: Configuring a Basic Spring Application

Scenario: Your company is developing a web application for managing a library. You need to use the Spring Framework to handle the backend operations.



```
1 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
2 <version>1.0-SNAPSHOT</version>
3 <packaging>jar</packaging>
4
5 <name>SpringCore</name>
6 <url>http://maven.apache.org</url>
7
8 <properties>
9 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
10 </properties>
11
12 <dependencies>
13
14 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
15 <dependency>
16 <groupId>org.springframework</groupId>
17 <artifactId>spring-context</artifactId>
18 <version>6.2.7</version>
19 </dependency>
20
21 <dependency>
22 <groupId>junit</groupId>
23 <artifactId>junit</artifactId>
24 <version>3.8.1</version>
25 <scope>test</scope>
26 </dependency>
27 </dependencies>
28
29 </project>
```



```
1 package com.library.repository;
2
3 public class BookRepository { no usages
4     public void saveBook(String bookName) { no usages
5         System.out.println("Saving book: " + bookName);
6     }
7 }
8
9
10
```

This screenshot shows the IntelliJ IDEA interface with the SpringCore project. The left sidebar displays the project structure, including the 'resources' directory. The main editor window shows the 'BookService.java' file. The code defines a 'BookService' class that uses a 'BookRepository' to manage books. It includes a constructor, a 'setBookRepository' method, and an 'addBook' method that prints the book name and saves it to the repository.

```
1 package com.library.service;
2
3 import com.library.repository.BookRepository;
4
5 public class BookService { 4 usages
6     private BookRepository bookRepository; 2 usages
7
8     public void setBookRepository(BookRepository bookRepository) { no usages
9         this.bookRepository = bookRepository;
10    }
11
12    public void addBook(String bookName) { 1 usage
13        System.out.println("Adding book to library: " + bookName);
14        bookRepository.saveBook(bookName);
15    }
16 }
17
```

This screenshot shows the 'applicationContext.xml' file in the SpringCore project. The XML configuration defines two beans: 'bookRepository' of type 'com.library.repository.BookRepository' and 'bookService' of type 'com.library.service.BookService'. The 'bookService' bean is configured with a property 'bookRepository' that references the 'bookRepository' bean.

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xsi:schemaLocation="
5           http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">
6
7     <bean id="bookRepository" class="com.library.repository.BookRepository" />
8
9     <bean id="bookService" class="com.library.service.BookService">
10         <property name="bookRepository" ref="bookRepository"/>
11     </bean>
12
13 </beans>
14
```

This screenshot shows the 'Main.java' file in the SpringCore project. The code defines a 'Main' class with a 'main' method that creates an 'ApplicationContext', retrieves the 'bookService' bean, and calls its 'addBook' method. Below the code, the 'Run' tab shows the execution output, indicating that the book was successfully added to the library and the process finished with exit code 0.

```
1 import com.library.service.BookService;
2 import org.springframework.context.ApplicationContext;
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4
5 public class Main {
6     public static void main(String[] args){
7         ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
8
9         BookService bookService = context.getBean("bookService", BookService.class);
10        bookService.addBook("Book added");
11    }
12 }
13
```

Run Main

"C:\Program Files\Java\jdk-21\bin\java.exe" ...

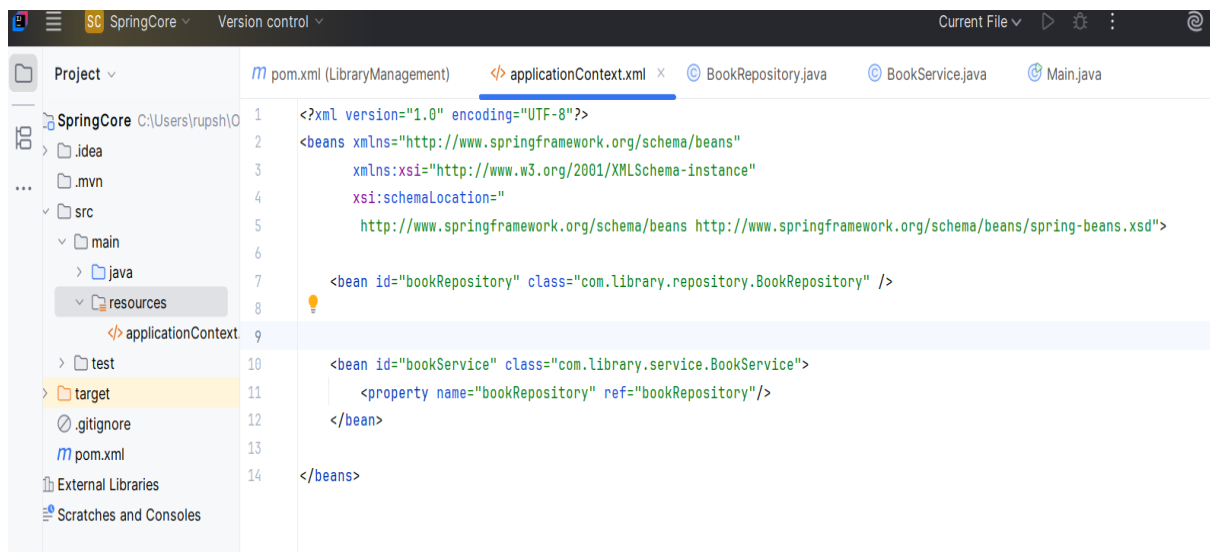
Adding book to library: Book added

Saving book: Book added

Process finished with exit code 0

Exercise 2: Implementing Dependency Injection

Scenario: In the library management application, you need to manage the dependencies between the BookService and BookRepository classes using Spring's IoC and DI.

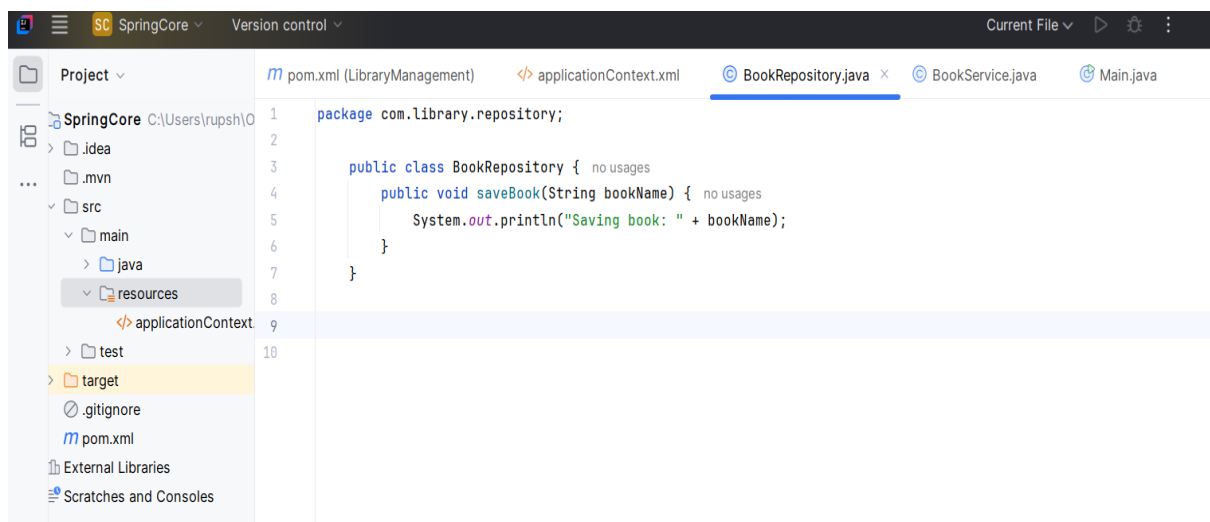


```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="
           http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd">

    <bean id="bookRepository" class="com.library.repository.BookRepository" />

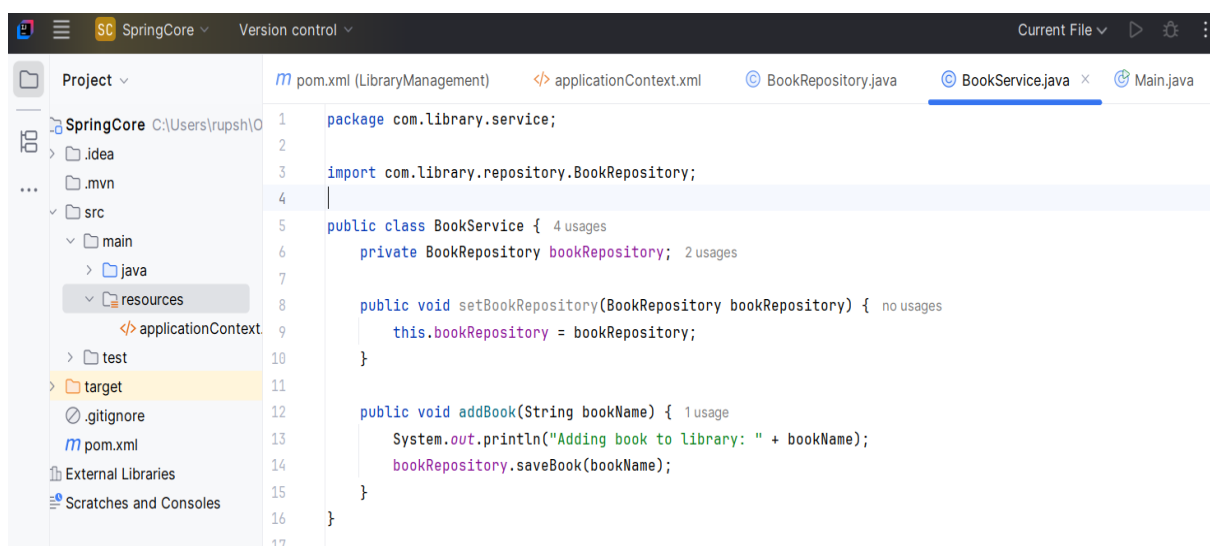
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository"/>
    </bean>

</beans>
```



```
package com.library.repository;

public class BookRepository {
    public void saveBook(String bookName) {
        System.out.println("Saving book: " + bookName);
    }
}
```



```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {
        System.out.println("Adding book to library: " + bookName);
        bookRepository.saveBook(bookName);
    }
}
```

Exercise 4: Creating and Configuring a Maven Project

Scenario: You need to set up a new Maven project for the library management application and add Spring dependencies.

The screenshot shows the 'New Project' dialog box in an IDE. The left sidebar lists 'New Project' options: Java, Kotlin, Groovy, Empty Project, and 'Generators'. Under 'Generators', 'Maven Archetype' is selected. The main area contains the following fields and options:

- Name:** MavenProject
- Location:** ~\OneDrive\Desktop\SpringCore. Below this, it says 'Project will be created in: ~\OneDrive\...SpringCore\MavenProject'. There is an unchecked checkbox for 'Create Git repository'.
- JDK:** 21 Oracle OpenJDK 21.0.2
- Catalog:** Internal. A link 'Manage catalogs...' is visible.
- Archetype:** .maven.archetypes:maven-archetype-quicksta. An 'Add...' button is next to it.
- Version:** 1.1
- Additional Properties:** A section with a '+' and '-' icon and the text 'No properties'.
- Advanced Settings:** A collapsed section containing:
 - GroupId:** com.library
 - ArtifactId:** MavenProject
 - Version:** 1.0-SNAPSHOT

At the bottom, there is a 'More via plugins...' link and a 'Create' button.

