# Week 3 :Spring Core and Maven

# Exercise 1: Configuring a Basic Spring Application

**CODE :**

**applicationContext.xml**

<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>

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void save() {  
 System.out.println("Saving book to database...");  
 }  
}

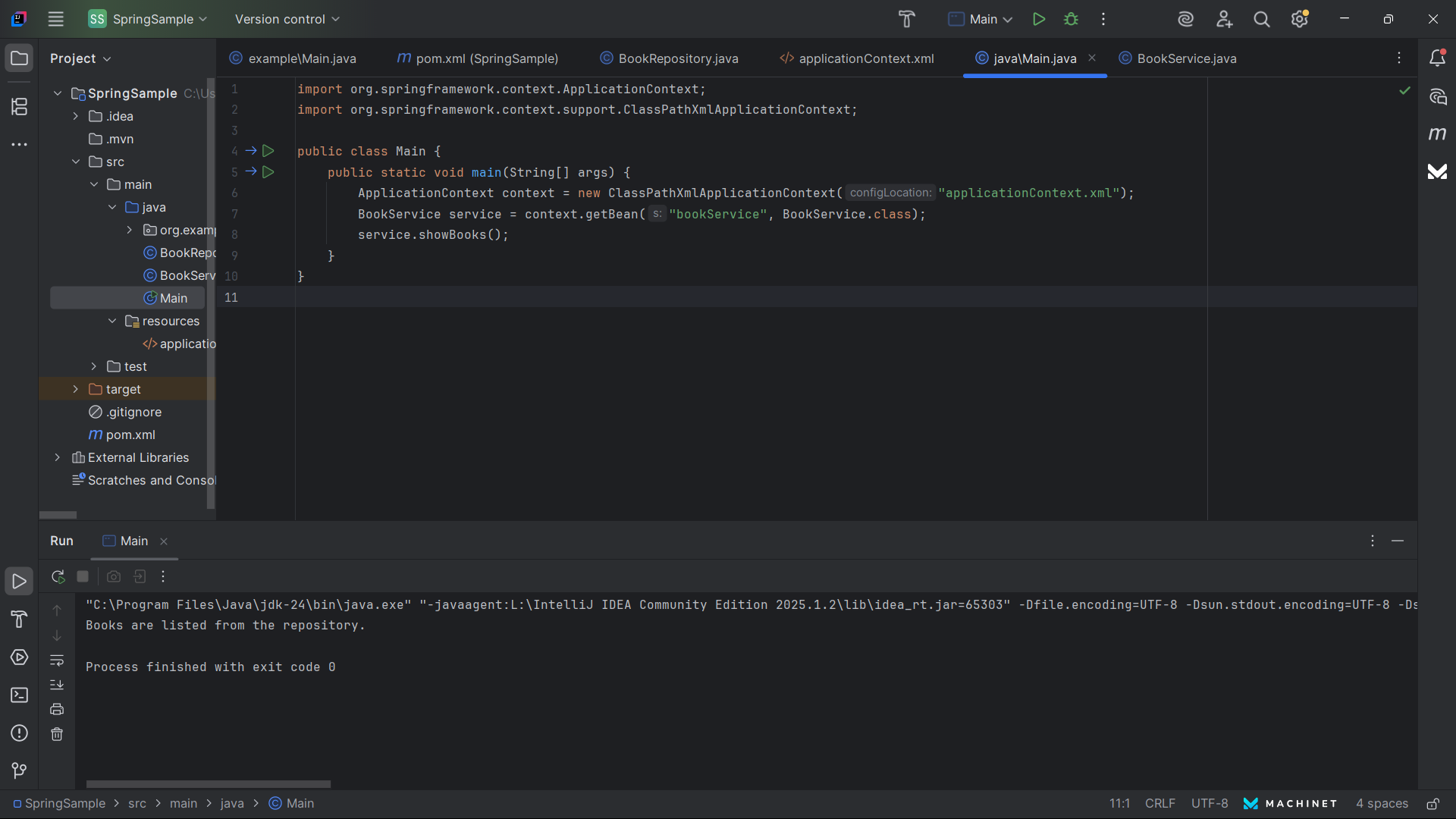
**BookService.java**

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 processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**LibraryManagementApplication.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.processBook();  
 }  
}

**Output :**



# Exercise 2: Implementing Dependency Injection

**CODE :**

**applicationContext.xml**

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

**BookRepository.java**

package com.library.repository;  
  
public class BookRepository {  
 public void save() {  
 System.out.println("Saving book to database...");  
 }  
}

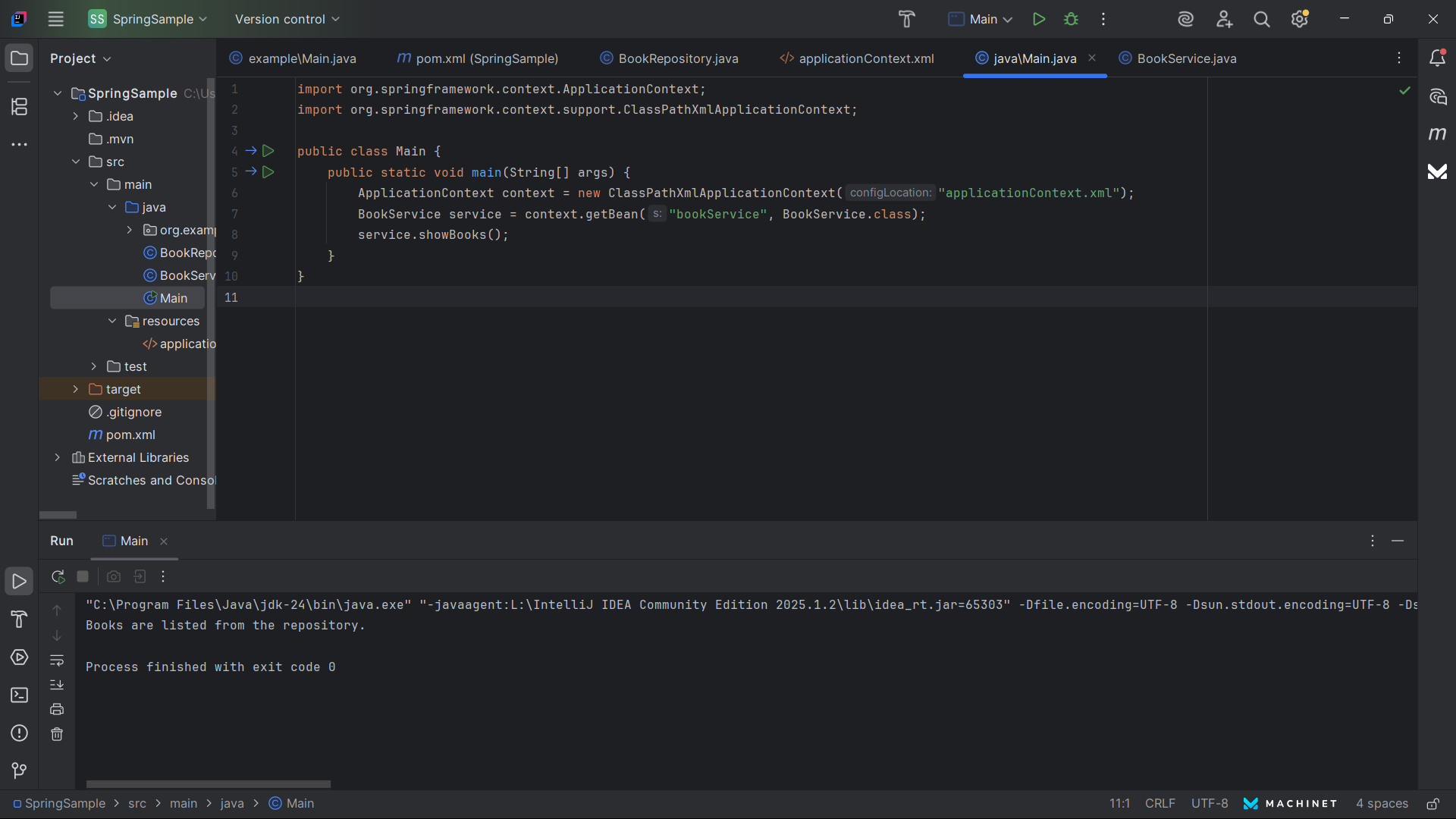
**BookService.java**

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 processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**LibraryManagementApplication.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = (BookService) context.getBean("bookService");  
 bookService.processBook();  
 }  
}

**Output :**



# Exercise 3: Implementing Logging with Spring AOP

**CODE :**

**pom.xml (Add dependency)**

<dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aspects</artifactId>  
 <version>5.3.33</version>  
</dependency>

**applicationContext.xml**

<aop:aspectj-autoproxy/>  
<bean id="loggingAspect" class="com.library.aspect.LoggingAspect"/>  
<bean id="bookRepository" class="com.library.repository.BookRepository"/>  
<bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
</bean>

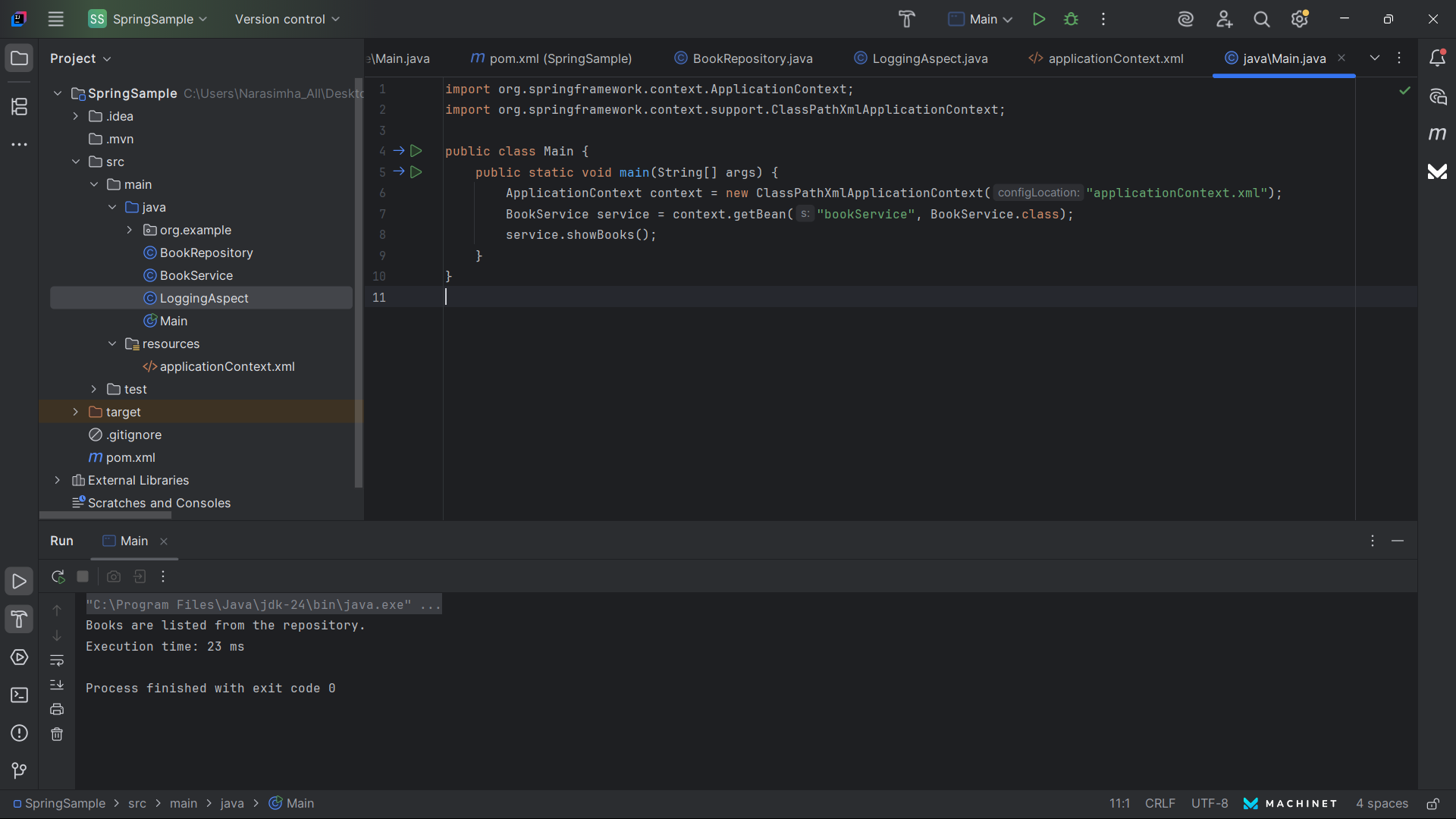
**LoggingAspect.java**

package com.library.aspect;  
  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.Around;  
import org.aspectj.lang.annotation.Aspect;  
  
@Aspect  
public class LoggingAspect {  
 @Around("execution(\* com.library.service.BookService.\*(..))")  
 public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {  
 long start = System.currentTimeMillis();  
 Object result = joinPoint.proceed();  
 long end = System.currentTimeMillis();  
 System.out.println("Execution time: " + (end - start) + "ms");  
 return result;  
 }  
}

**BookService.java**

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 processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**Output :**



# Exercise 4: Creating and Configuring a Maven Project

**CODE :**

**pom.xml**

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
</dependencies>  
  
<build>  
 <plugins>  
 <plugin>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.11.0</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
</build>

**Output :**

# Exercise 5: Configuring the Spring IoC Container

**CODE :**

**applicationContext.xml**

<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>

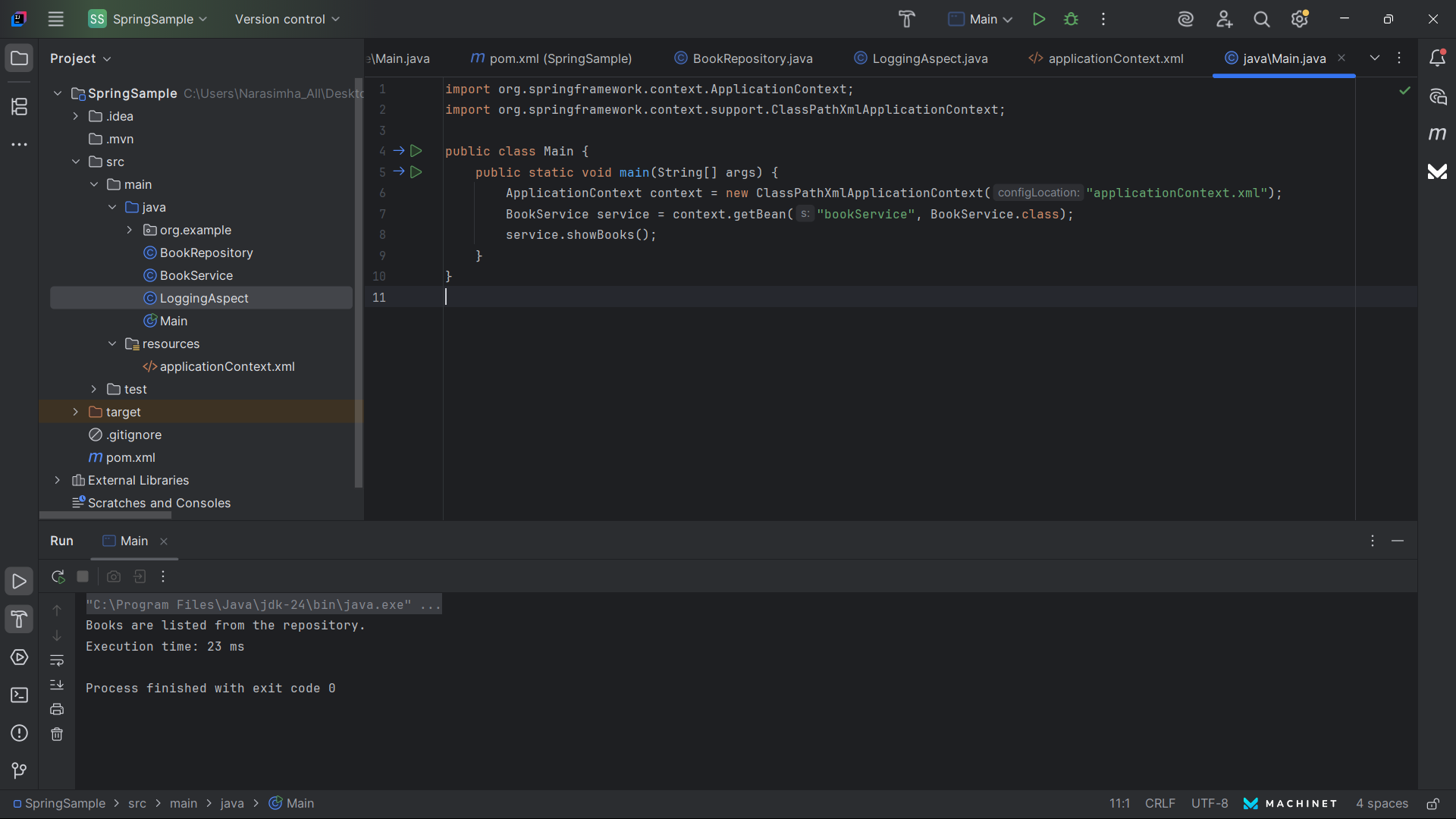
**BookService.java**

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 processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**LibraryManagementApplication.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = context.getBean(BookService.class);  
 bookService.processBook();  
 }  
}

**Output :**



# Exercise 6: Configuring Beans with Annotations

**CODE :**

**applicationContext.xml**

<context:component-scan base-package="com.library"/>  
<aop:aspectj-autoproxy/>

**BookRepository.java**

package com.library.repository;  
  
import org.springframework.stereotype.Repository;  
  
@Repository  
public class BookRepository {  
 public void save() {  
 System.out.println("Saving book to database...");  
 }  
}

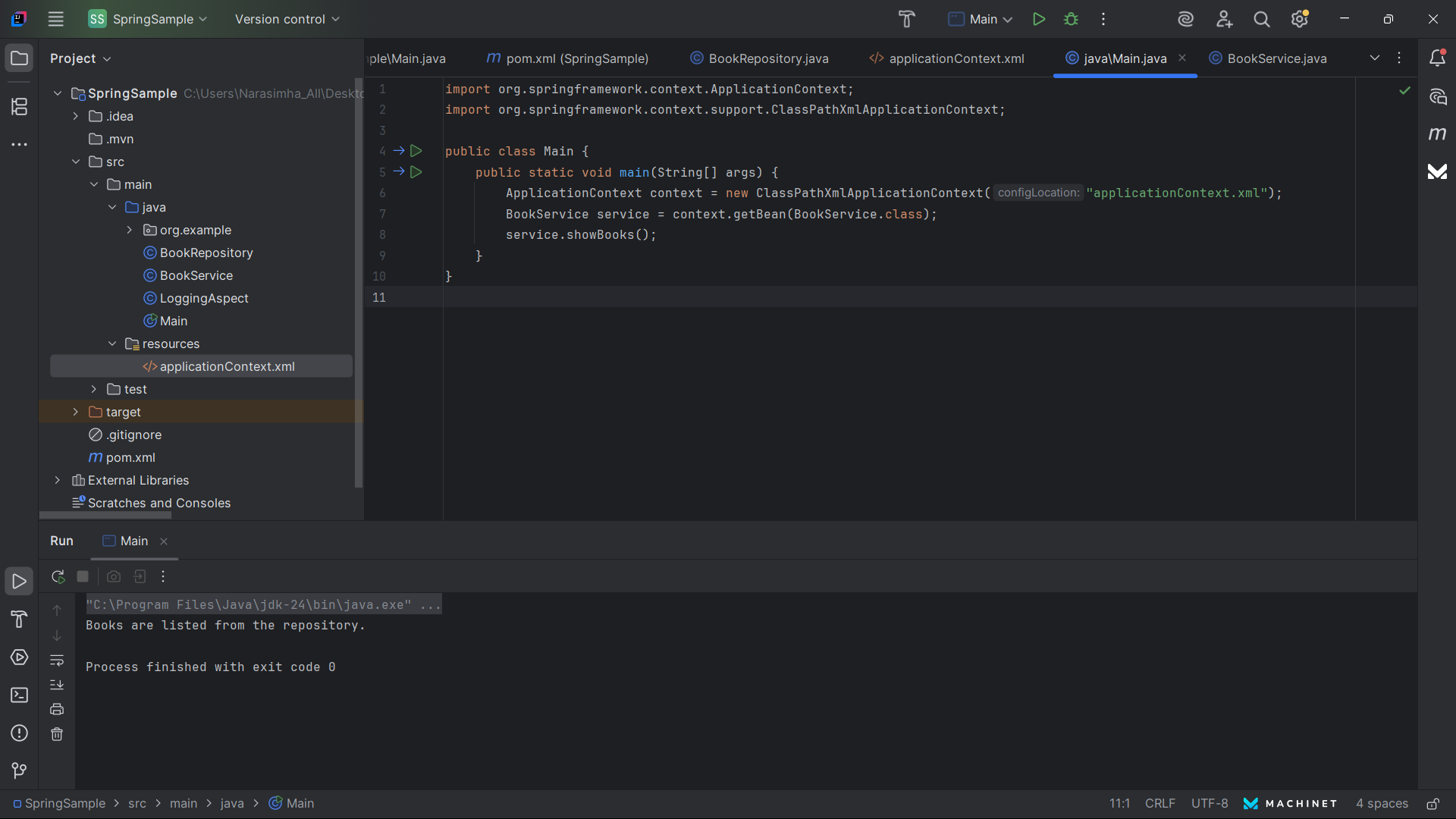
**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class BookService {  
 @Autowired  
 private BookRepository bookRepository;  
  
 public void processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**LibraryManagementApplication.java**

package com.library;  
  
import com.library.service.BookService;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class LibraryManagementApplication {  
 public static void main(String[] args) {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService = context.getBean(BookService.class);  
 bookService.processBook();  
 }  
}

**Output :**



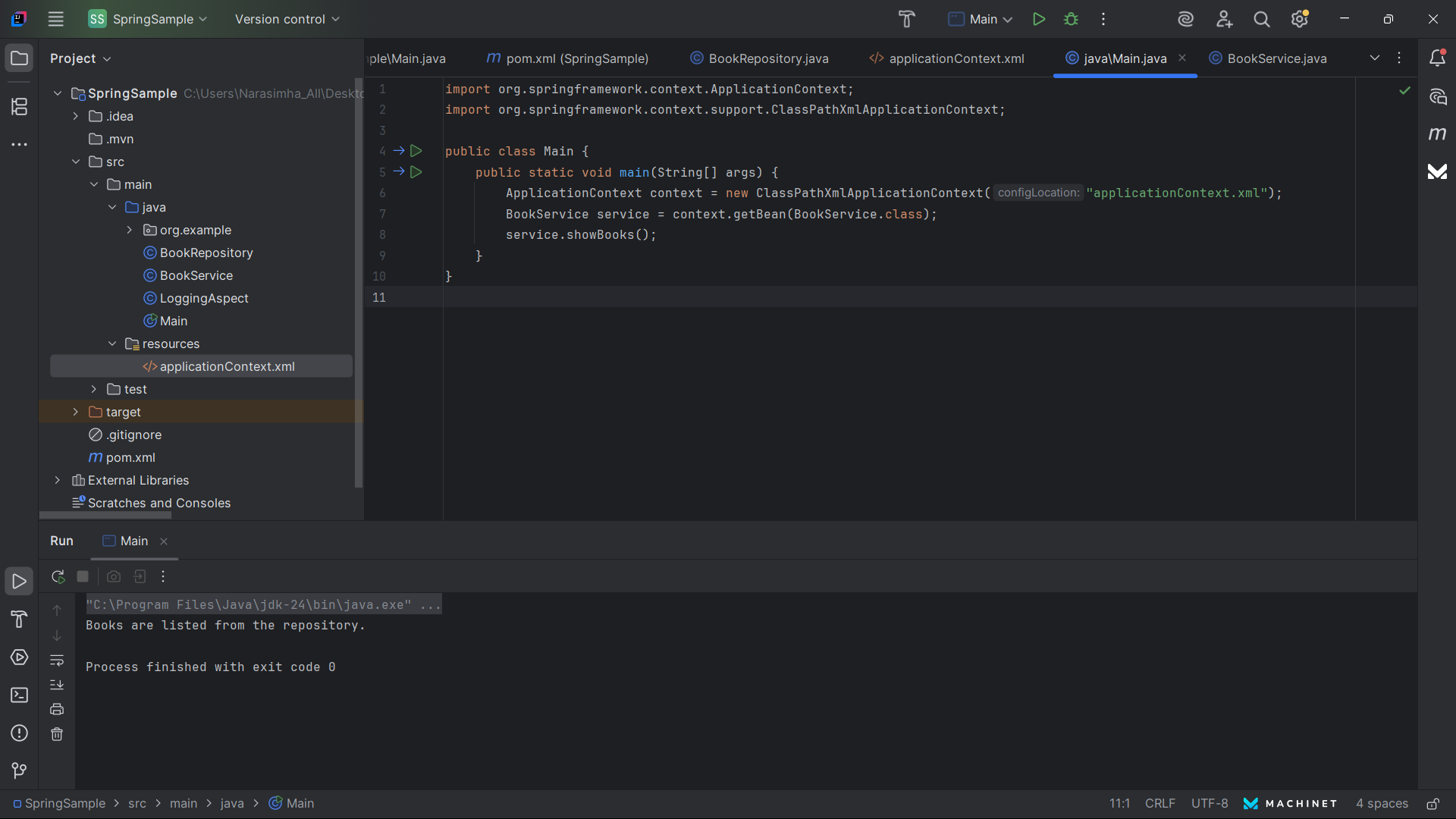
# Exercise 7: Implementing Constructor and Setter Injection

**CODE :**

**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class BookService {  
 private final BookRepository bookRepository;  
  
 @Autowired  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void processBook() {  
 System.out.println("Processing book...");  
 bookRepository.save();  
 }  
}

**Output :**



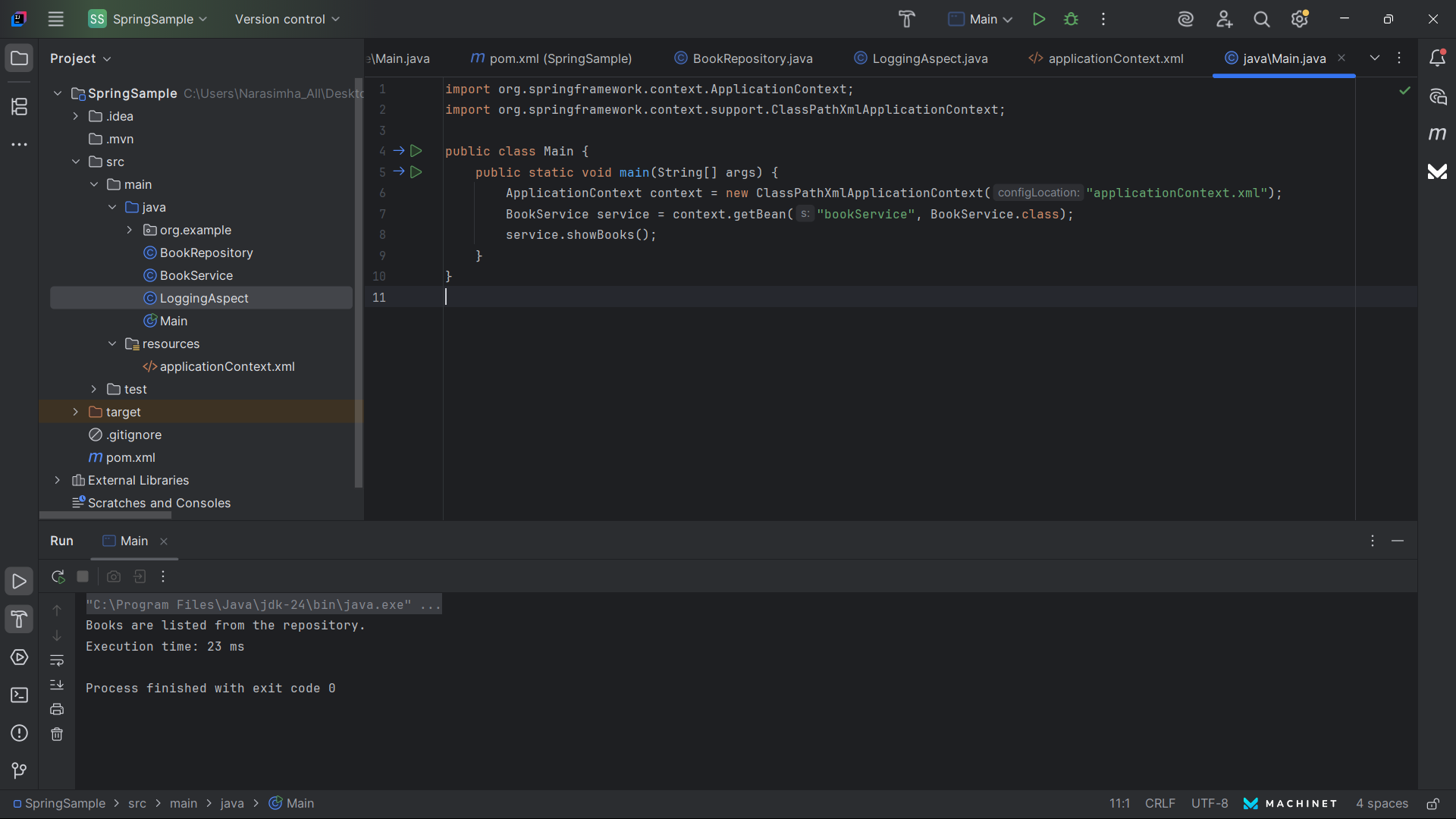
# Exercise 8: Implementing Basic AOP with Spring

**CODE :**

**LoggingAspect.java**

package com.library.aspect;  
  
import org.aspectj.lang.ProceedingJoinPoint;  
import org.aspectj.lang.annotation.Around;  
import org.aspectj.lang.annotation.Aspect;  
  
@Aspect  
public class LoggingAspect {  
 @Around("execution(\* com.library.service.BookService.\*(..))")  
 public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {  
 long start = System.currentTimeMillis();  
 Object result = joinPoint.proceed();  
 long end = System.currentTimeMillis();  
 System.out.println("Execution time: " + (end - start) + "ms");  
 return result;  
 }  
}

**Output :**



# Exercise 9: Creating a Spring Boot Application

**CODE :**

**Book.java**

package com.library.entity;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
  
@Entity  
public class Book {  
 @Id  
 private int id;  
 private String title;  
  
 public Book() {}  
  
 public Book(int id, String title) {  
 this.id = id;  
 this.title = title;  
 }  
  
 public int getId() { return id; }  
 public String getTitle() { return title; }  
}

**BookRepository.java**

package com.library.repository;  
  
import com.library.entity.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface BookRepository extends JpaRepository<Book, Integer> {}

**BookController.java**

package com.library.controller;  
  
import com.library.entity.Book;  
import com.library.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.List;  
  
@RestController  
@RequestMapping("/books")  
public class BookController {  
 @Autowired  
 private BookRepository repo;  
  
 @PostMapping  
 public Book save(@RequestBody Book book) {  
 return repo.save(book);  
 }  
  
 @GetMapping  
 public List<Book> getAll() {  
 return repo.findAll();  
 }  
}

**LibraryManagementBootApplication.java**

package com.library;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementBootApplication {  
 public static void main(String[] args) {  
 SpringApplication.run(LibraryManagementBootApplication.class, args);  
 }  
}

**application.properties**

spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.driverClassName=org.h2.Driver  
spring.jpa.hibernate.ddl-auto=update  
spring.h2.console.enabled=true

**Output :**

