SPRING CORE AND MAVEN

Exercise 1: Configuring a Basic Spring Application

File Name: 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 service = context.getBean("bookService", BookService.class);
      service.saveBook();
   }
}
```

File Name: BookRepository.java

```
package com.library.repository;

public class BookRepository {
   public void saveBook() {
      System.out.println("Book saved to the database.");
   }
}
```

File Name: BookService.java

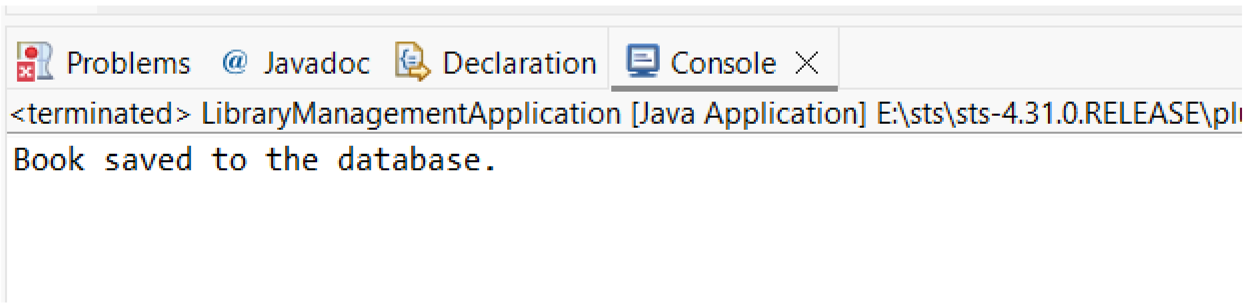
```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {
   private BookRepository bookRepository;
```

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

    public void saveBook() {
        bookRepository.saveBook();
    }
}
```

File Name: ApplicationContext.xml

```
<?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>
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.0.RELEASE\pl
Book saved to the database.
```

Exercise 2: Implementing Dependency Injection

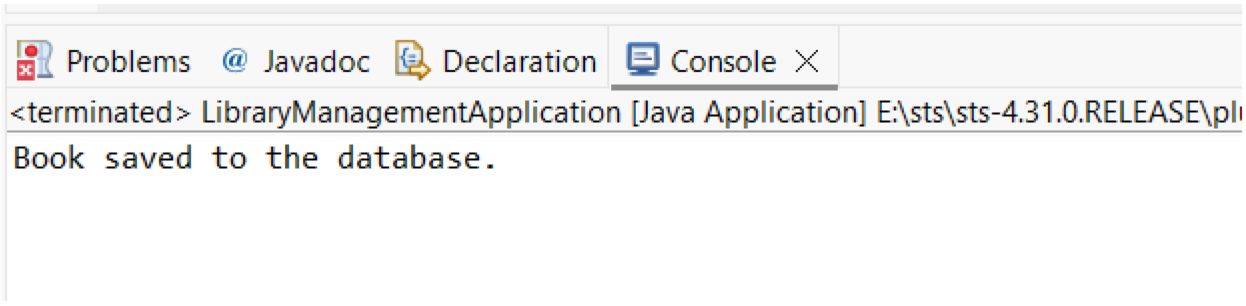
File name: BookService.java **-----it has a setter method**

```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 saveBook() {
        bookRepository.saveBook();
    }
}
```

File name: applicationContext.xml ---- proper wiring

```xml
<?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>
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.0.RELEASE\pl
Book saved to the database.
```

Exercise 3: Implementing Logging with Spring AOP

File name: pom.xml-- Add Spring AOP dependency

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

  <modelVersion>4.0.0</modelVersion>

  <groupId>com.library</groupId>
  <artifactId>LibraryManagement</artifactId>
  <version>1.0-SNAPSHOT</version>

  <dependencies>
    <!-- Spring Core Dependency -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>
    <dependency>
  <groupId>org.springframework</groupId>
  <artifactId>spring-aop</artifactId>
  <version>5.3.34</version>
</dependency>
```

```
<dependency>
    <groupId>org.aspectj</groupId>
    <artifactId>aspectjweaver</artifactId>
    <version>1.9.21.1</version>
</dependency>

    </dependencies>

    <build>
        <plugins>
            <!-- Maven Compiler Plugin -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.8.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

File Name: applicationContext.xml ----updated xml file for enable AOP

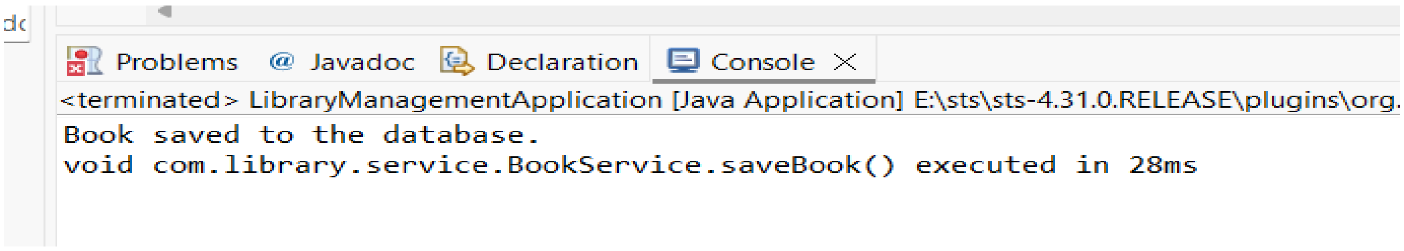
```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="
         http://www.springframework.org/schema/beans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

```
    <!-- Repository Bean -->
    <bean id="bookRepository" class="com.library.repository.BookRepository" />


    <!-- Service Bean with DI -->
    <bean id="bookService" class="com.library.service.BookService">
        <property name="bookRepository" ref="bookRepository" />
    </bean>


    <!-- AOP Aspect Bean -->
    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />


    <!-- Enable AspectJ auto-proxy support -->
    <aop:aspectj-autoproxy />


</beans>
```

File name: LoggingAspect.java ---creating aspect class

```
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.*.*(..))")
    public Object logExecutionTime(ProceedingJoinPoint joinPoint) throws Throwable {
        long start = System.currentTimeMillis();
        Object result = joinPoint.proceed();  // call the actual method
        long end = System.currentTimeMillis();
        System.out.println(joinPoint.getSignature() + " executed in " + (end - start) + "ms");
        return result;
    }
}
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.0.RELEASE\plugins\org.
Book saved to the database.
void com.library.service.BookService.saveBook() executed in 28ms
```

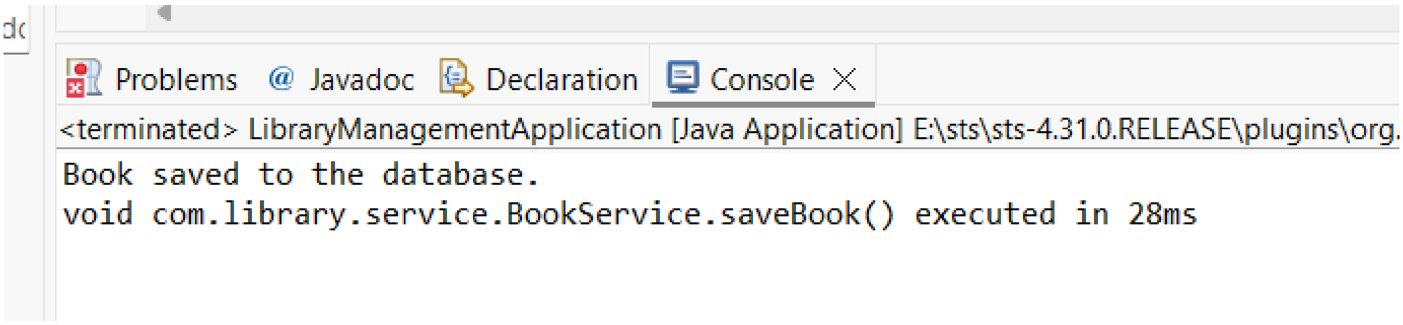
Exercise 4: Creating and Configuring a Maven Project

File name: pom.xml ---updated

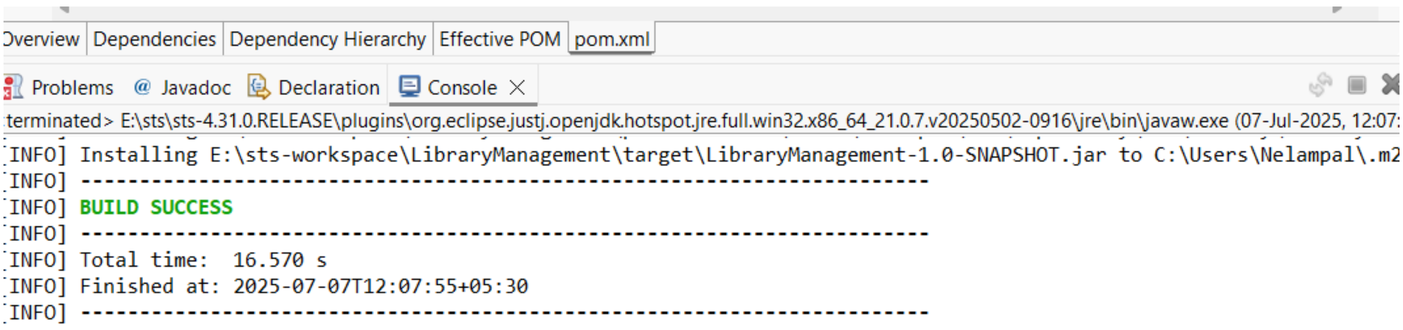
```
<project xmlns="http://maven.apache.org/POM/4.0.0"

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

     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0

                    http://maven.apache.org/xsd/maven-4.0.0.xsd">


  <modelVersion>4.0.0</modelVersion>


  <groupId>com.library</groupId>

  <artifactId>LibraryManagement</artifactId>

  <version>1.0-SNAPSHOT</version>


  <dependencies>

  <!-- Spring Core -->

  <dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-context</artifactId>

    <version>5.3.34</version>

  </dependency>


  <!-- Spring AOP -->

  <dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-aop</artifactId>

    <version>5.3.34</version>

  </dependency>


  <!-- AspectJ Weaver (required for AOP to work) -->

  <dependency>
```

```
        <groupId>org.aspectj</groupId>
        <artifactId>aspectjweaver</artifactId>
        <version>1.9.21.1</version>
    </dependency>

    <!-- Optional: Spring WebMVC if needed later -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-webmvc</artifactId>
        <version>5.3.34</version>
    </dependency>
</dependencies>


    <build>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-compiler-plugin</artifactId>
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.0.RELEASE\plugins\org.
Book saved to the database.
void com.library.service.BookService.saveBook() executed in 28ms
```



```
terminated> E:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\javaw.exe (07-Jul-2025, 12:07:
INFO] Installing E:\sts-workspace\LibraryManagement\target\LibraryManagement-1.0-SNAPSHOT.jar to C:\Users\Nelampal\.m2
INFO] ------------------------------------------------------------------------
INFO] BUILD SUCCESS
INFO] ------------------------------------------------------------------------
INFO] Total time:  16.570 s
INFO] Finished at: 2025-07-07T12:07:55+05:30
INFO] ------------------------------------------------------------------------
```

Exercise 5: Configuring the Spring IoC Container

File name: applicationContext.xml

```
<?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">
  <!-- Define BookRepository bean -->
  <bean id="bookRepository" class="com.library.repository.BookRepository" />

  <!-- Define BookService and inject BookRepository -->
  <bean id="bookService" class="com.library.service.BookService">
    <property name="bookRepository" ref="bookRepository" />
  </bean>
</beans>
```

File name: BookRepository.java

```
package com.library.repository;
public class BookRepository {
  public void saveBook() {
    System.out.println("Book saved to the database.");
  }
}
```

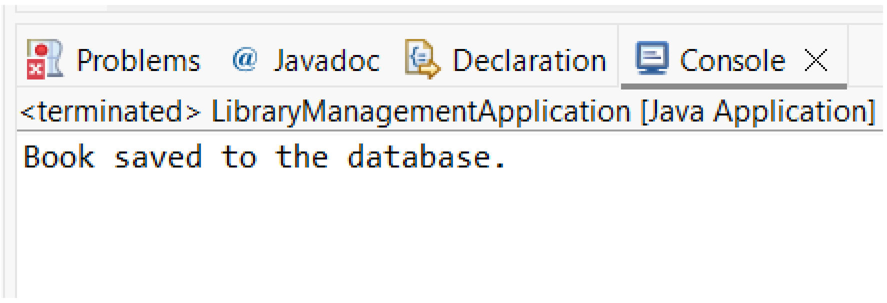
File name: BookService.java

```
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 saveBook() {

        bookRepository.saveBook();

    }

}
```

File name: 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 service = context.getBean("bookService", BookService.class);

        service.saveBook();

    }

}
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application]
Book saved to the database.
```

Exercise 6: Configuring Beans with Annotations

File name: applicationContext.xml

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

  <!-- Enable annotation-based scanning -->
  <context:component-scan base-package="com.library" />

</beans>
```

File name: BookRepository.java

```
package com.library.repository;

import org.springframework.stereotype.Repository;

@Repository
public class BookRepository {
  public void saveBook() {
    System.out.println("Book saved to the database.");
  }
}
```

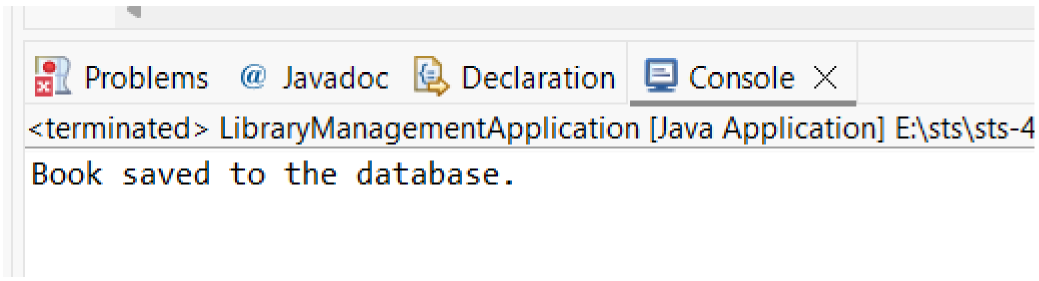
File name: BookService.java

```
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 saveBook() {

        bookRepository.saveBook();

    }

}
```

File name: 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 service = context.getBean(BookService.class);

        service.saveBook();

    }

}
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4
Book saved to the database.
```

Exercise 7: Implementing Constructor and Setter Injection

File name: applicationContext.xml

```
<?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">

   <!-- Constructor + Setter Injection -->
   <bean id="bookRepository" class="com.library.repository.BookRepository" />

   <bean id="bookService" class="com.library.service.BookService">
      <constructor-arg ref="bookRepository" />
      <property name="serviceType" value="Premium Service" />
   </bean>

</beans>
```

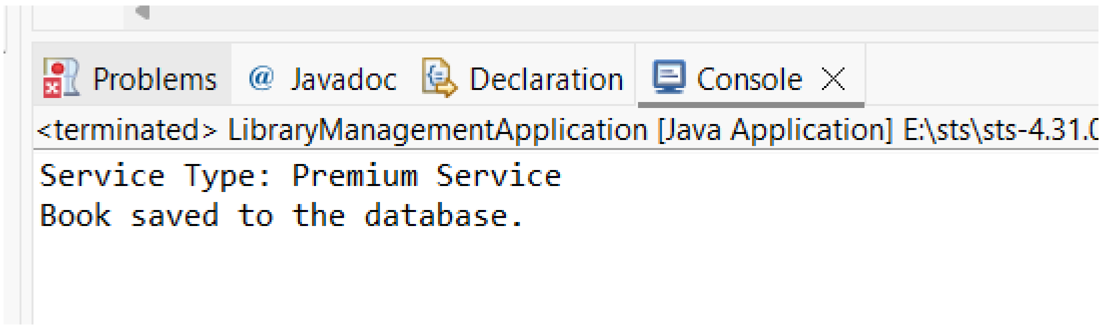
File name: BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
   private BookRepository bookRepository;
   private String serviceType;
   //  Constructor Injection
   public BookService(BookRepository bookRepository) {
      this.bookRepository = bookRepository;
   }
   //  Setter Injection
   public void setServiceType(String serviceType) {
        this.serviceType = serviceType;
    }
    public void saveBook() {
        System.out.println("Service Type: " + serviceType);
        bookRepository.saveBook();
    }
}
```

File name: 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 service = context.getBean(BookService.class);
        service.saveBook();
    }
}
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.
Service Type: Premium Service
Book saved to the database.
```

Exercise 8: Implementing Basic AOP with Spring

File name: pom.xml

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
                      http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
```

```xml
    <groupId>com.library</groupId>
    <artifactId>LibraryManagement</artifactId>
    <version>1.0-SNAPSHOT</version>
    <dependencies>
    <!-- Spring Core -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.34</version>
    </dependency>
    <!-- Spring AOP -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-aop</artifactId>
      <version>5.3.34</version>
    </dependency>
    <!-- AspectJ Weaver (required for AOP to work) -->
    <dependency>
      <groupId>org.aspectj</groupId>
      <artifactId>aspectjweaver</artifactId>
      <version>1.9.21.1</version>
    </dependency>
    <!-- Optional: Spring WebMVC if needed later -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.34</version>
    </dependency>
  </dependencies>
    <build>
    <plugins>
      <plugin>
        <groupId>org.apache.maven.plugins</groupId>
        <artifactId>maven-compiler-plugin</artifactId>
```

```
            <version>3.8.1</version>
            <configuration>
                <source>1.8</source>
                <target>1.8</target>
            </configuration>
        </plugin>
    </plugins>
</build>

</project>
```

File name: LoggingAspect.java

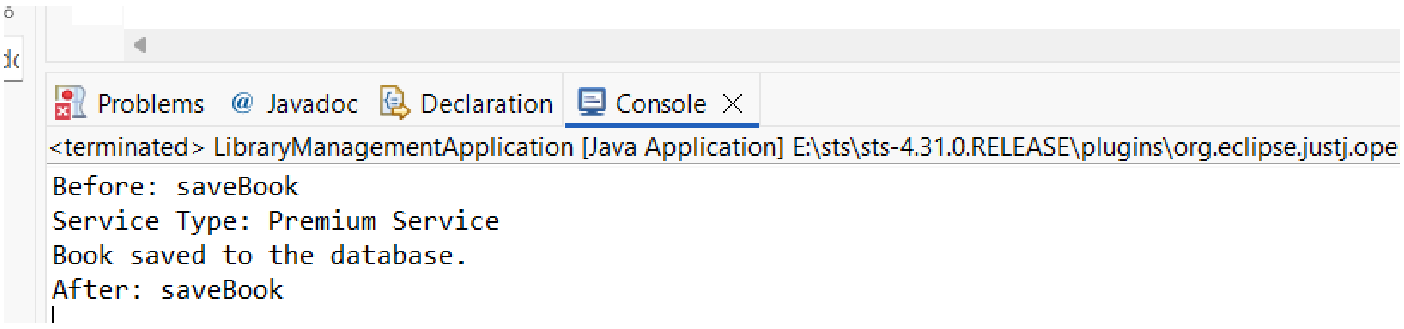
```java
package com.library.aspect;
import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
@Aspect
public class LoggingAspect {
    @Before("execution(* com.library.service.*.*(..))")
    public void logBefore(JoinPoint joinPoint) {
        System.out.println("Before: " + joinPoint.getSignature().getName());
    }
    @After("execution(* com.library.service.*.*(..))")
    public void logAfter(JoinPoint joinPoint) {
        System.out.println("After: " + joinPoint.getSignature().getName());
    }
}
```

File name: applicationContext.xml

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

```
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop.xsd">

    <!-- Enable AOP -->
    <aop:aspectj-autoproxy />

    <!-- Define Repository -->
    <bean id="bookRepository" class="com.library.repository.BookRepository" />
    <!-- Define Service with constructor and setter injection -->
    <bean id="bookService" class="com.library.service.BookService">
        <constructor-arg ref="bookRepository" />
        <property name="serviceType" value="Premium Service" />
    </bean>
    <!-- Register Aspect -->
    <bean id="loggingAspect" class="com.library.aspect.LoggingAspect" />
</beans>
```

OUTPUT



```
<terminated> LibraryManagementApplication [Java Application] E:\sts\sts-4.31.0.RELEASE\plugins\org.eclipse.justj.ope
Before: saveBook
Service Type: Premium Service
Book saved to the database.
After: saveBook
```

Exercise 9: Creating a Spring Boot Application

File name: 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;
```

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

   @Autowired
   private BookRepository bookRepository;

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

   @PostMapping
   public Book addBook(@RequestBody Book book) {
      return bookRepository.save(book);
   }

   @GetMapping("/{id}")
   public Book getBookById(@PathVariable int id) {
      return bookRepository.findById(id).orElse(null);
   }

   @PutMapping("/{id}")
   public Book updateBook(@PathVariable int id, @RequestBody Book book) {
      book.setBookId(id);
      return bookRepository.save(book);
   }

   @DeleteMapping("/{id}")
   public void deleteBook(@PathVariable int id) {
      bookRepository.deleteById(id);
   }
}
```

File name: Book.java

```java
package com.library.entity;

import jakarta.persistence.Entity;
import jakarta.persistence.Id;

@Entity
public class Book {
   @Id
   private int bookId;
   private String title;
   private String author;

   // Getters and Setters
   public int getBookId() { return bookId; }
   public void setBookId(int bookId) { this.bookId = bookId; }
   public String getTitle() { return title; }
   public void setTitle(String title) { this.title = title; }
   public String getAuthor() { return author; }
   public void setAuthor(String author) { this.author = author; }
}
```

File name: BookRepository.java

```java
package com.library.repository;
import com.library.entity.Book;
import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Integer> {
}
```

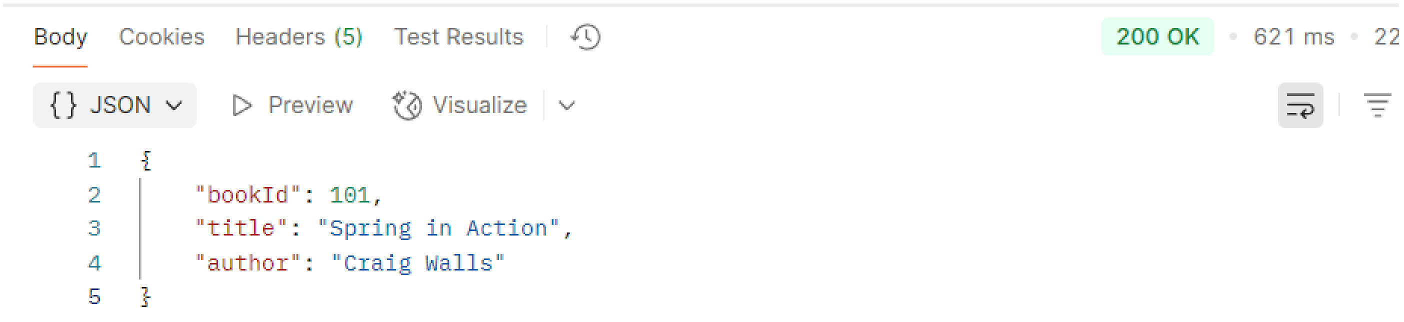
File name: application.properties

```properties
spring.datasource.url=jdbc:h2:mem:librarydb
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect
spring.jpa.hibernate.ddl-auto=update
spring.h2.console.enabled=true

server.port=8081
```

File name: LibraryManagemntApplication

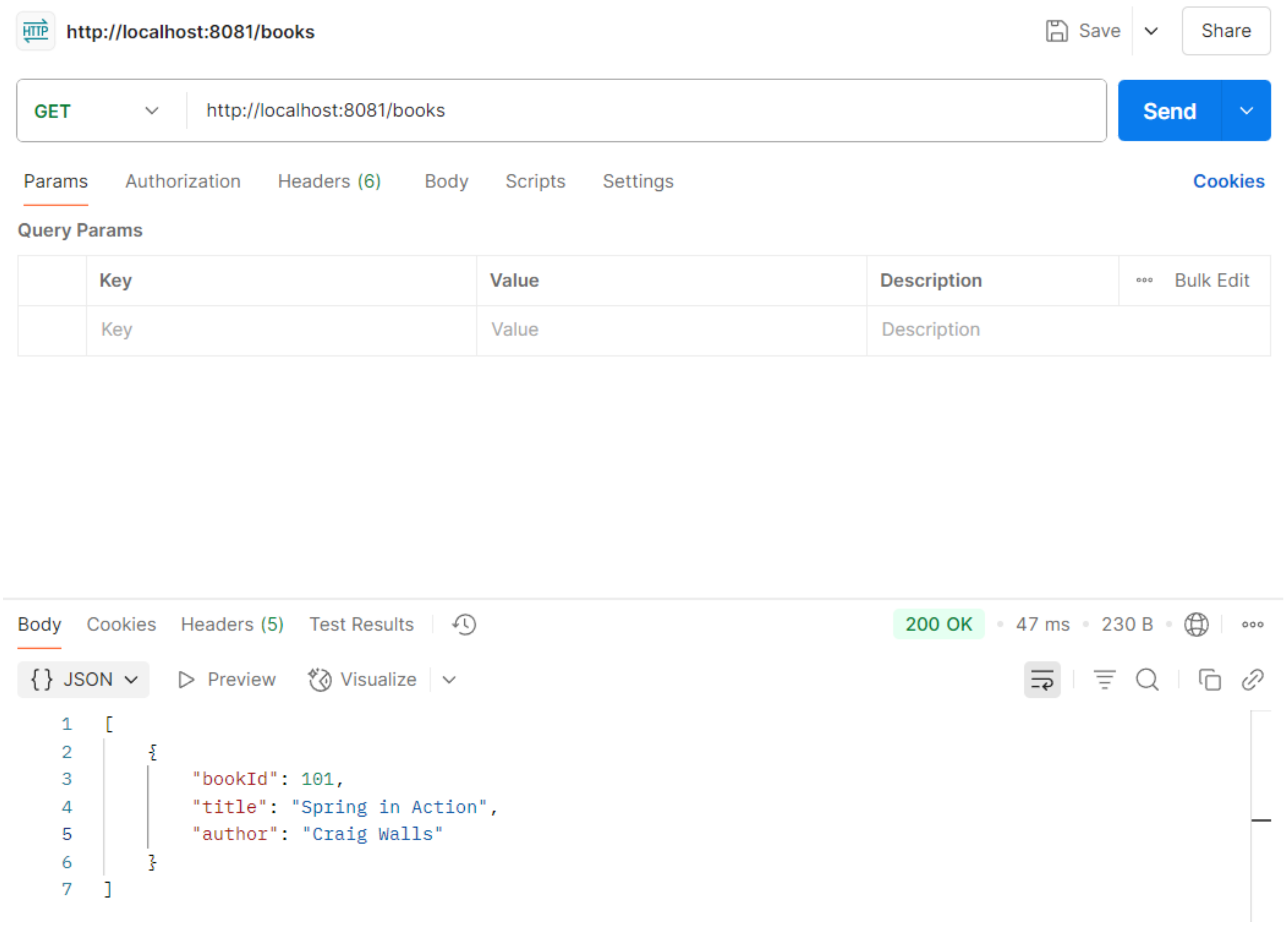
```java
package com.library;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;


@SpringBootApplication

public class LibraryManagementApplication {

    public static void main(String[] args) {

        SpringApplication.run(LibraryManagementApplication.class, args);

    }

}
```

OUTPUT



Body Cookies Headers (5) Test Results 200 OK 621 ms 22

```
{
    "bookId": 101,
    "title": "Spring in Action",
    "author": "Craig Walls"
}
```



http://localhost:8081/books

GET http://localhost:8081/books

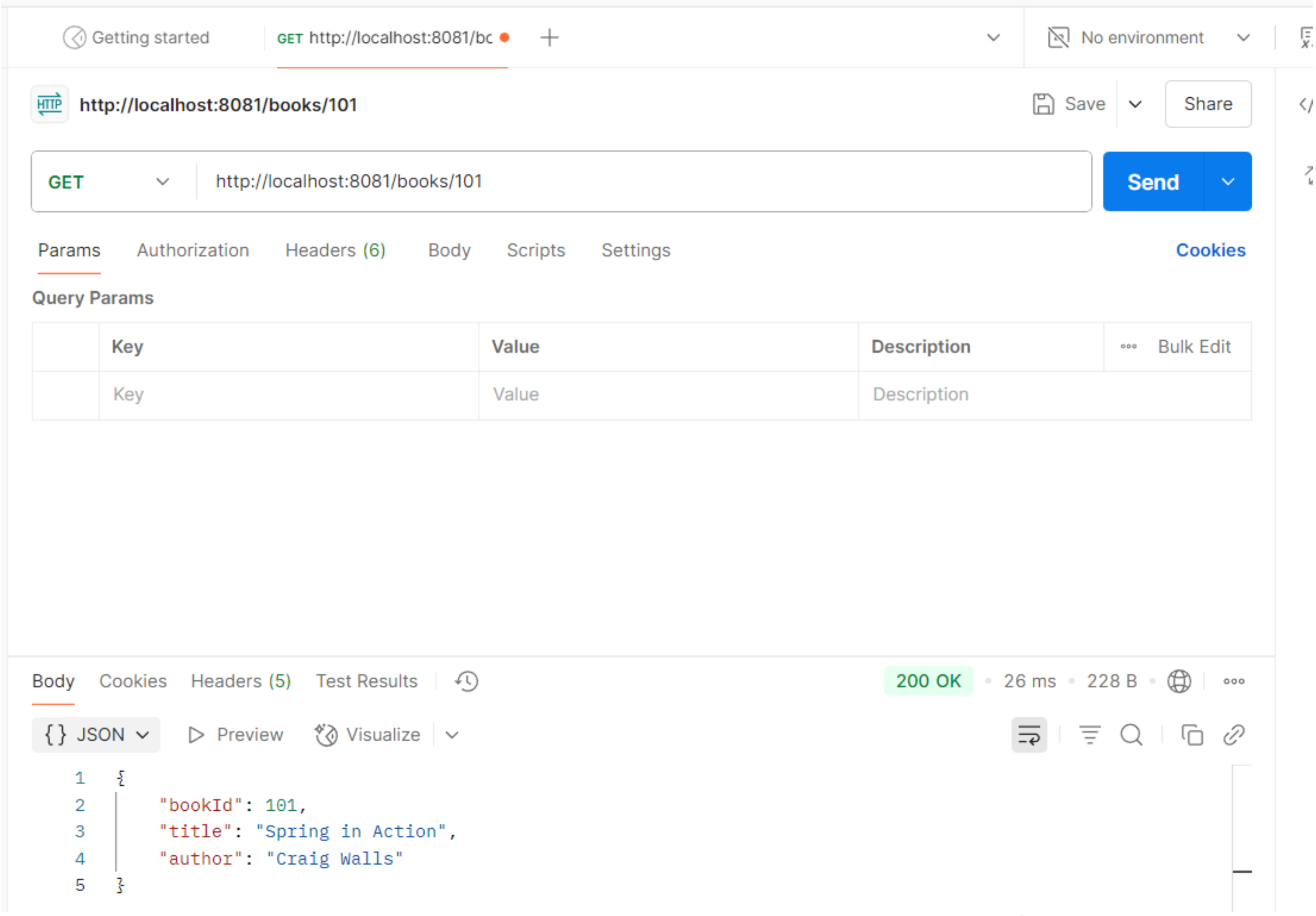
Params Authorization Headers (6) Body Scripts Settings Cookies

Query Params

	Key	Value	Description	Bulk Edit
	Key	Value	Description	

Body Cookies Headers (5) Test Results 200 OK 47 ms 230 B

```
[
    {
        "bookId": 101,
        "title": "Spring in Action",
        "author": "Craig Walls"
    }
]
```



http://localhost:8081/books/101

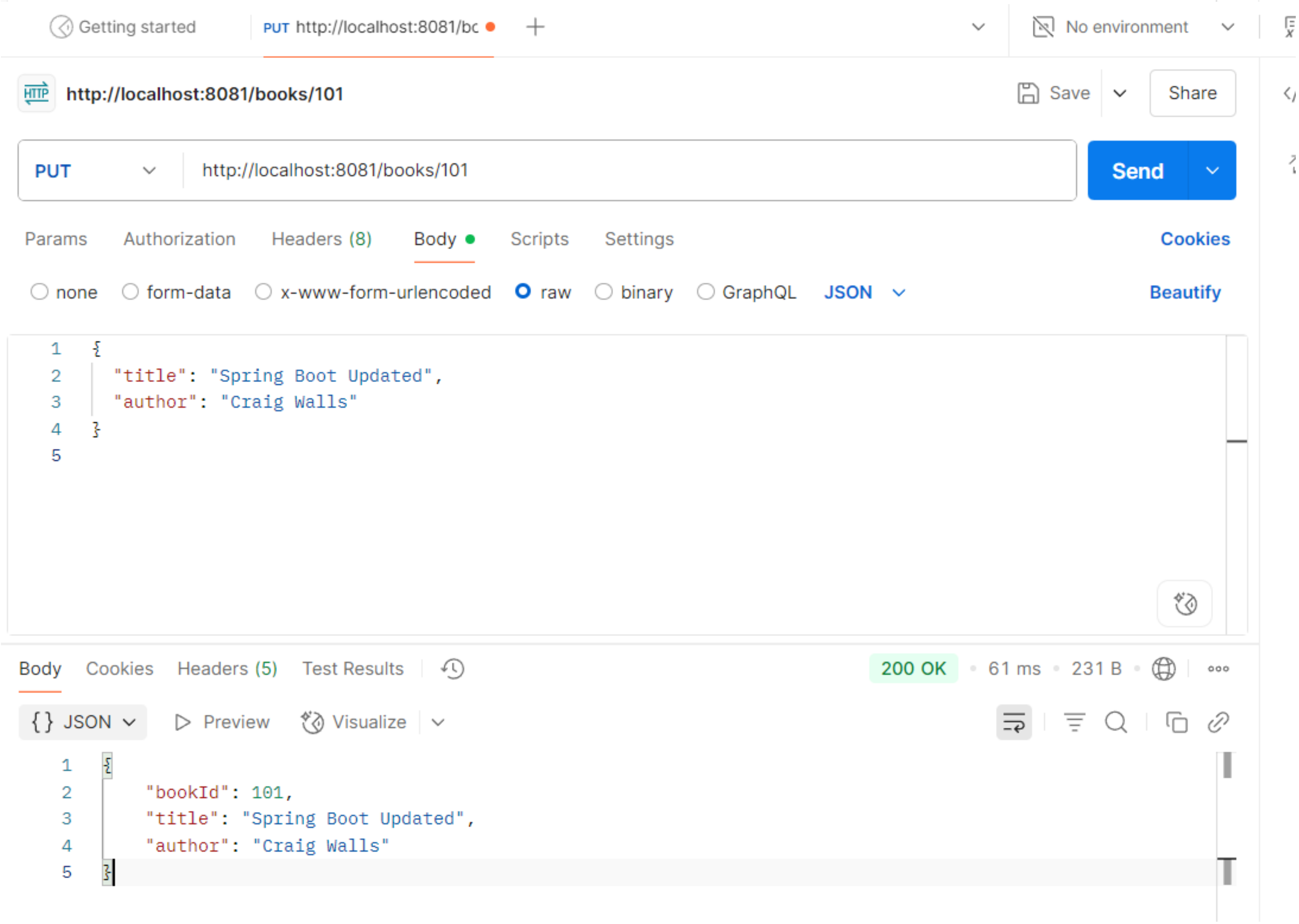
GET http://localhost:8081/books/101

Key	Value	Description
Key	Value	Description

200 OK • 26 ms • 228 B

{} JSON

```
{
    "bookId": 101,
    "title": "Spring in Action",
    "author": "Craig Walls"
}
```



http://localhost:8081/books/101

PUT http://localhost:8081/books/101

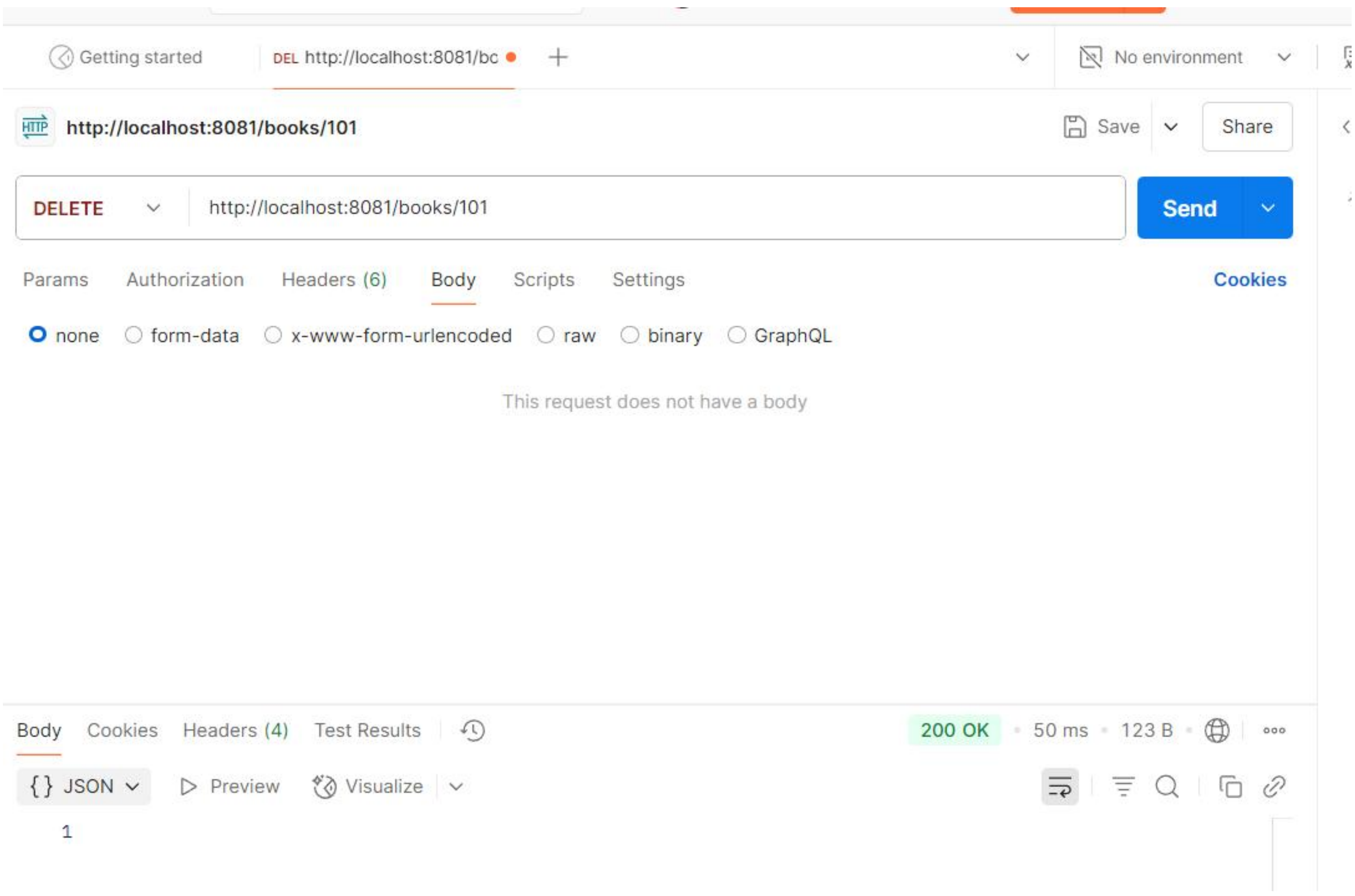
none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{
  "title": "Spring Boot Updated",
  "author": "Craig Walls"
}

```

200 OK • 61 ms • 231 B

{} JSON

```
{
    "bookId": 101,
    "title": "Spring Boot Updated",
    "author": "Craig Walls"
}
```



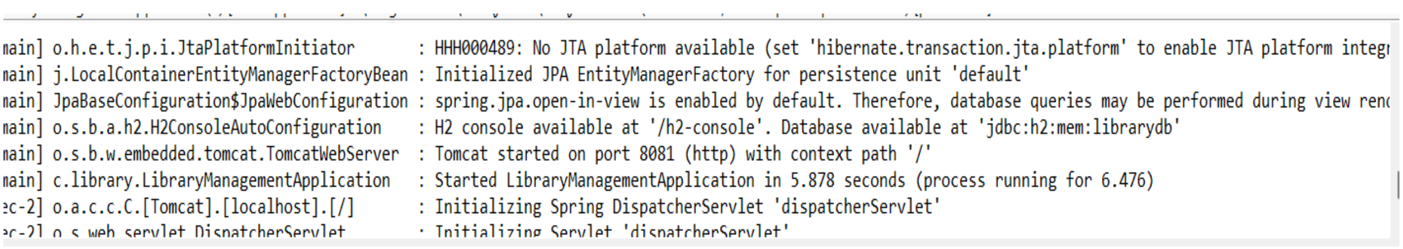
http://localhost:8081/books/101

DELETE http://localhost:8081/books/101

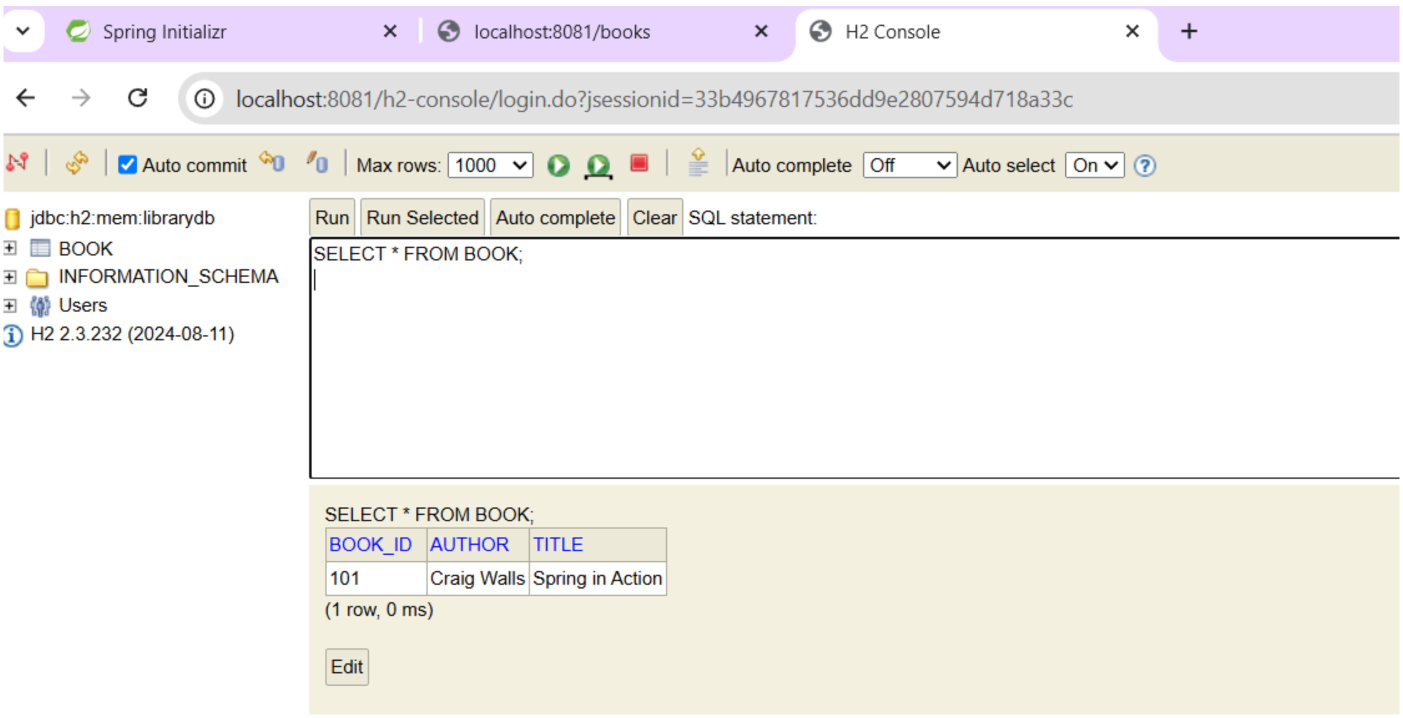
This request does not have a body

200 OK • 50 ms • 123 B

```
1
```



```
main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integr
main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may be performed during view renc
main] o.s.b.a.h2.H2ConsoleAutoConfiguration    : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:librarydb'
main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8081 (http) with context path '/'
main] c.library.LibraryManagementApplication   : Started LibraryManagementApplication in 5.878 seconds (process running for 6.476)
ec-2] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
ec-2] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
```



localhost:8081/h2-console/login.do?jsessionid=33b4967817536dd9e2807594d718a33c

Auto commit [✓] Max rows: 1000 Auto complete: Off Auto select: On

SQL statement:

```
SELECT * FROM BOOK;
```

SELECT * FROM BOOK;

BOOK_ID	AUTHOR	TITLE
101	Craig Walls	Spring in Action

(1 row, 0 ms)