Scenario:

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

Steps:

1. **Set Up a Spring Project:**
   - Create a Maven project named **LibraryManagement**.
   - Add Spring Core dependencies in the **pom.xml** file.
2. **Configure the Application Context:**
   - Create an XML configuration file named **applicationContext.xml** in the **src/main/resources** directory.
   - Define beans for **BookService** and **BookRepository** in the XML file.
3. **Define Service and Repository Classes:**
   - Create a package **com.library.service** and add a class **BookService**.
   - Create a package **com.library.repository** and add a class **BookRepository**.
4. **Run the Application:**
   - Create a main class to load the Spring context and test the configuration.

Code:

POM.XML FILE:

```
<?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>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
```

```xml
<groupId>com.library</groupId>

<artifactId>libraray_management</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>libraray_management</name>

<description>Spring Library Management App</description>


<properties>

   <java.version>17</java.version>

</properties>


<dependencies>

   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter</artifactId>

   </dependency>


   <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-devtools</artifactId>

      <scope>runtime</scope>

      <optional>true</optional>

   </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>
            <plugin>
                <groupId>org.codehaus.mojo</groupId>
                <artifactId>exec-maven-plugin</artifactId>
                <version>3.1.0</version>
                <configuration>
                    <mainClass>com.library.libraray_management.LibraryManagementApplication</mainClas
s>
                </configuration>
            </plugin>
        </plugins>
    </build>
</project>
```

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.libraray_management.repository.BookRepository"/>

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

```
</beans>
```

LibraryManagementApplication.java

```
package com.library.libraray_management;
import com.library.libraray_management.service.BookService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.CommandLineRunner;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class LibraryManagementApplication implements CommandLineRunner {

    @Autowired
    private BookService bookService;

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

    @Override
    public void run(String... args) throws Exception {
        bookService.addBook("Spring Boot In Depth");
    }
}
```

BookService.java

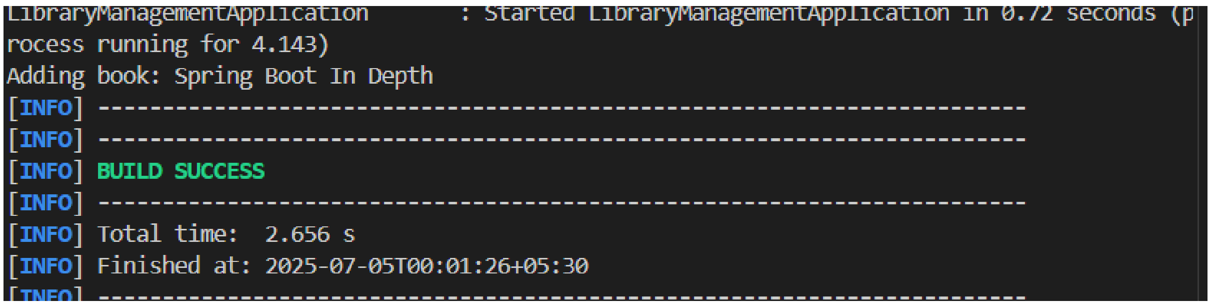
```
package com.library.libraray_management.service;
```

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


import org.springframework.stereotype.Service;
import com.library.libraray_management.repository.BookRepository;

@Service
public class BookService {
    private BookRepository bookRepository;

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

    public void addBook(String name) {
        System.out.println("Adding book: " + name);
        bookRepository.save(name);
    }
}
```

BookRepository.java

```
package com.library.libraray_management.repository;

import org.springframework.stereotype.Repository;

@Repository
public class BookRepository {
    public void save(String name) {
        System.out.println("Book saved: " + name);
    }
}
```

OUTPUT:



```
LibraryManagementApplication       : Started LibraryManagementApplication in 0.72 seconds (p
rocess running for 4.143)
Adding book: Spring Boot In Depth
[INFO] ------------------------------------------------------------------------
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  2.656 s
[INFO] Finished at: 2025-07-05T00:01:26+05:30
[INFO] ------------------------------------------------------------------------
```

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.

Steps:

1. **Modify the XML Configuration:**
   - Update **applicationContext.xml** to wire **BookRepository** into **BookService**.
2. **Update the BookService Class:**
   - Ensure that **BookService** class has a setter method for **BookRepository**.
3. **Test the Configuration:**
   - Run the **LibraryManagementApplication** main class to verify the dependency injection.

CODE:

ApplicationContext.xml

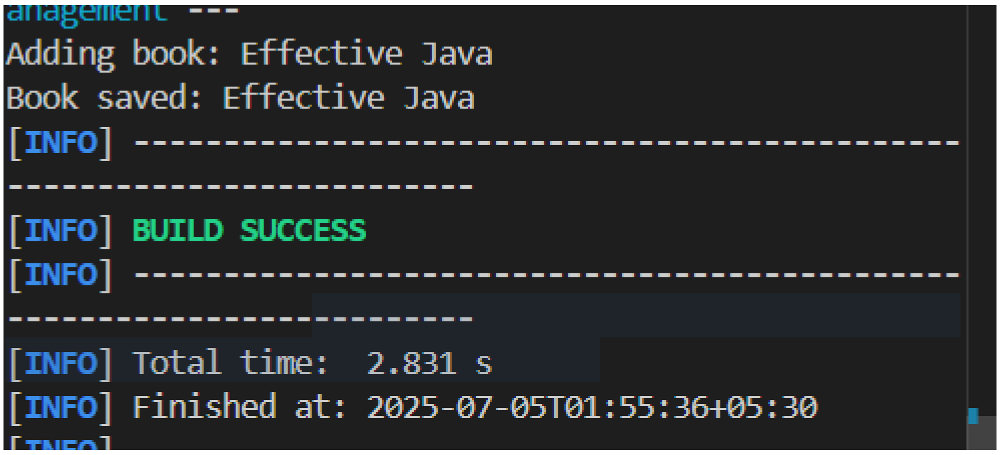
```
<?xml version="1.0" encoding="UTF-8"?>

<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 the BookRepository bean -->
```

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


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

</beans>
```

BookService.java

```
package com.library.libraray_management.service;
import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;
import com.library.libraray_management.repository.BookRepository;

@Service
public class BookService {
    private BookRepository bookRepository;

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

    public void addBook(String name) {
        System.out.println("Adding book: " + name);
        bookRepository.save(name);
    }
}
```

OUTPUT:



```
anagement ---
Adding book: Effective Java
Book saved: Effective Java
[INFO] ------------------------------------------------
--------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------
--------------------------
[INFO] Total time:  2.831 s
[INFO] Finished at: 2025-07-05T01:55:36+05:30
[INFO]
```

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.

Steps:

1. **Create a New Maven Project:**
    - Create a new Maven project named **LibraryManagement**.
2. **Add Spring Dependencies in pom.xml:**
    - Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
3. **Configure Maven Plugins:**
    - Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

Pom.xml file:

```
<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>
```

```
<groupId>com.library</groupId>
<artifactId>LibraryManagement</artifactId>
<version>1.0-SNAPSHOT</version>
<name>Library Management</name>

<properties>
  <maven.compiler.source>1.8</maven.compiler.source>
  <maven.compiler.target>1.8</maven.compiler.target>
</properties>

<dependencies>
  <!-- Spring Context -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.3.36</version>
  </dependency>

  <!-- Spring AOP -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-aop</artifactId>
    <version>5.3.36</version>
  </dependency>

  <!-- Spring Web MVC -->
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.36</version>
    </dependency>

    <!-- Servlet API -->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>4.0.1</version>
      <scope>provided</scope>
    </dependency>
  </dependencies>

  <build>
    <plugins>
      <!-- Maven Compiler Plugin -->
      <plugin>
        <artifactId>maven-compiler-plugin</artifactId>
        <version>3.8.1</version>
        <configuration>
          <source>1.8</source>
          <target>1.8</target>
        </configuration>
      </plugin>
    </plugins>
  </build>

</project>
```

Folder Structure:

```
LibraryManagement/
 ├── pom.xml
 ├── src/
 |   ├── main/
 |   |   ├── java/
 |   |   |   └── com/
 |   |   |       └── library/
 |   |   |           └── management/
 |   |   |               ├── AppConfig.java        # (optional) Java-based config
 |   |   |               ├── MainApp.java          # Main class to run Spring context
 |   |   |               ├── service/
 |   |   |               |   └── BookService.java
 |   |   |               └── repository/
 |   |   |                   └── BookRepository.java
 |   |   └── resources/
 |   |       └── applicationContext.xml            # XML-based Spring configuration
 |
 |   └── test/
 |       └── java/
 |           └── com/
 |               └── library/
 |                   └── management/
 |                       └── LibraryManagementTests.java
```