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.

Steps:

1. **Set Up a Spring Project:**
    - Create a Maven project named **LibraryManagement**.
    - Add Spring Core dependencies in the **pom.xml** file.

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>
 <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>
 </dependencies>
</project>
```

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.

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>
```

BookRepository.java

```
package com.library.repository;
public class BookRepository {
        public String getBookDetails() {
                return "Book details from repository";
        }
}
```

BookService.java

```
package com.library.service;
import com.library.repository.BookRepository;
public class BookService {
  private BookRepository bookRepository;
  // Setter for Dependency Injection
  public void setBookRepository(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
  }
  public void displayBook() {
```

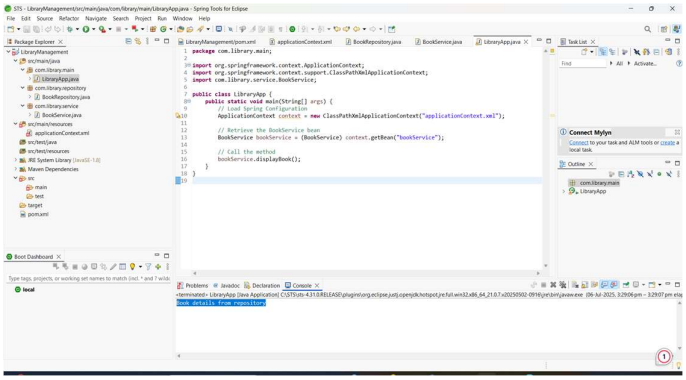
```
        System.out.println(bookRepository.getBookDetails());
    }
}
```

3. **Run the Application:**
   - Create a main class to load the Spring context and test the configuration.

LibraryApp.java

```
package com.library.main;

import org.springframework.context.ApplicationContext;

import
org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class LibraryApp {

   public static void main(String[] args) {

      // Load Spring Configuration

      ApplicationContext context = new
ClassPathXmlApplicationContext("applicationContext.xml");

      // Retrieve the BookService bean

      BookService bookService = (BookService)
context.getBean("bookService");

      // Call the method

      bookService.displayBook();

   }

}
```

OUTPUT:



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**.

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>
```

2. **Update the BookService Class:**
   - Ensure that **BookService** class has a setter method for **BookRepository**.

BookRepository.java

```
package com.library.repository;
public class BookRepository {
   public String getBookDetails() {
      return "Book details from repository";
   }
}
```

BookService.java

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

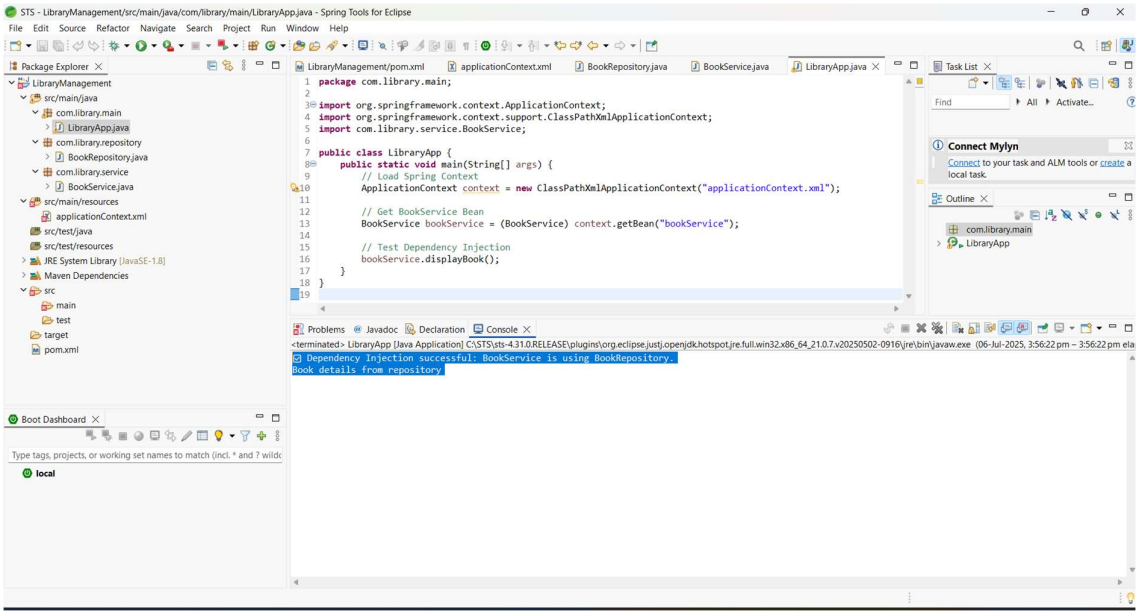
```
import com.library.repository.BookRepository;
public class BookService {
   private BookRepository bookRepository;
   // Setter for Dependency Injection
   public void setBookRepository(BookRepository bookRepository) {
      this.bookRepository = bookRepository;
   }
   public void displayBook() {
      System.out.println("✅ Dependency Injection successful: BookService is using BookRepository.");
      System.out.println(bookRepository.getBookDetails());
   }
}
```

3. **Test the Configuration:**
   - Run the **LibraryManagementApplication** main class to verify the dependency injection.

LibraryApp.java

```
package com.library.main;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
import com.library.service.BookService;
public class LibraryApp {
   public static void main(String[] args) {
      // Load Spring Context
      ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");
      // Get BookService Bean
      BookService bookService = (BookService) context.getBean("bookService");
      // Test Dependency Injection
      bookService.displayBook();
   }
}
```

OUTPUT:



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.

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

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

```
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>5.3.25</version>
    </dependency>
  </dependencies>
</project>
```

Output: