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 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>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

  <properties>
    <maven.compiler.source>17</maven.compiler.source>
    <maven.compiler.target>17</maven.compiler.target>
  </properties>

  <dependencies>
    <!-- Spring Core & Context -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-context</artifactId>
      <version>6.1.6</version>
    </dependency>

    <!-- Logging for Spring -->
    <dependency>
      <groupId>org.slf4j</groupId>
      <artifactId>slf4j-simple</artifactId>
      <version>2.0.13</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.

      Bean Definitions for XML files

```
<?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
     https://www.springframework.org/schema/beans/spring-beans.xsd">

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

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

</beans>
```

3. **Define Service and Repository Classes:**
    - Create a package **com.library.service** and add a class **BookService**.

      BookService.java File

```
package com.library.repository;

import java.util.ArrayList;
import java.util.List;

public class BookRepository {

  private final List<String> books = new ArrayList<>();

  public void add(String title) {
    books.add(title);
  }

  public List<String> getAllBooks() {
    return books;
  }
}
```

- Create a package **com.library.repository** and add a class **BookRepository**.

BookRepository.java file

```
package com.library.repository;

import java.util.ArrayList;
import java.util.List;

public class BookRepository {

  private final List<String> books = new ArrayList<>();

  public void add(String title) {
    books.add(title);
  }

  public List<String> getAllBooks() {
    return books;
  }
}
```

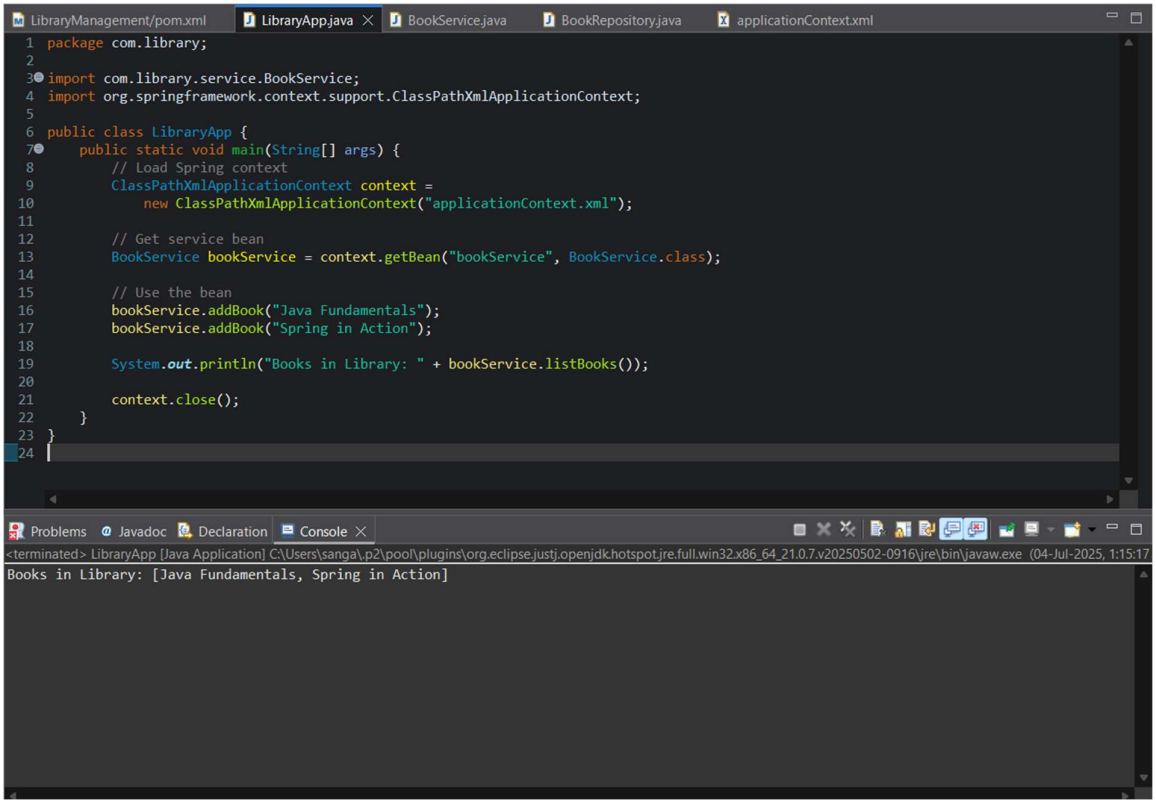
4. **Run the Application:**
   - Create a main class to load the Spring context and test the configuration.

LibraryApp.java

```
package com.library;


import com.library.service.BookService;

import org.springframework.context.support.ClassPathXmlApplicationContext;


public class LibraryApp {

  public static void main(String[] args) {

    // Load Spring XML configuration

    ClassPathXmlApplicationContext context =

      new ClassPathXmlApplicationContext("applicationContext.xml");


    // Retrieve the service bean

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


    // Use the service

    bookService.addBook("Java Fundamentals");
```

```
        bookService.addBook("Spring in Action");

        System.out.println("Books in Library: " + bookService.listBooks());

        // Close context
        context.close();
    }
}
```

Result:



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 file**

```
<?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
     https://www.springframework.org/schema/beans/spring-beans.xsd">

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

  <!-- Define BookService bean and inject 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 file**

```
package com.library.service;
import com.library.repository.BookRepository;
import java.util.List;
```

```
public class BookService {

  private BookRepository bookRepository;

  // Setter method for Spring to inject BookRepository
  public void setBookRepository(BookRepository bookRepository) {
    this.bookRepository = bookRepository;
  }

  public void addBook(String title) {
    bookRepository.add(title);
  }

  public List<String> listBooks() {
    return bookRepository.getAllBooks();
  }
}
```

3. **Test the Configuration:**
   - Run the **LibraryManagementApplication** main class to verify the dependency injection.
   - **LibraryManagamentApplication [**LibraryApp.java**] file**

```
package com.library;
import com.library.service.BookService;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryApp {
  public static void main(String[] args) {
    // Load Spring context
    ClassPathXmlApplicationContext context =
      new ClassPathXmlApplicationContext("applicationContext.xml");

    // Retrieve BookService bean (with BookRepository injected)
```

```
        BookService bookService = context.getBean("bookService", BookService.class);

        // Test service logic
        bookService.addBook("Head First Java");
        bookService.addBook("Effective Java");

        System.out.println("Books in Library: " + bookService.listBooks());

        context.close();
    }
}
```

Result:

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**.
    - **File Structure**

      LibraryManagement/

      — src/main/java

      — src/main/resources

      — pom.xml
2. **Add Spring Dependencies in pom.xml:**
    - Include dependencies for Spring Context, Spring AOP, and Spring WebMVC.
    - **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>0.0.1-SNAPSHOT</version>
  <packaging>jar</packaging>

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

  <dependencies>
    <!-- Spring Context (Core + Beans + 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>

    <!-- SLF4J Logging -->
    <dependency>
        <groupId>org.slf4j</groupId>
        <artifactId>slf4j-simple</artifactId>
        <version>2.0.13</version>
    </dependency>
  </dependencies>
</project>
```

3. **Configure Maven Plugins:**
    - Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.
    - **Configure Maven Compiler plugin**

```
<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>
```