SPRING CORE AND MAVEN

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.
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.

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>5.3.30</version>
        </dependency>
    </dependencies>
</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">

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

    <!-- Service Bean with dependency injection -->
    <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;

    // Setter for dependency injection
    public void setBookRepository(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void listBooks() {
        System.out.println("BookService: Listing books...");
        bookRepository.getAllBooks();
    }
}
```

BookRepository.java

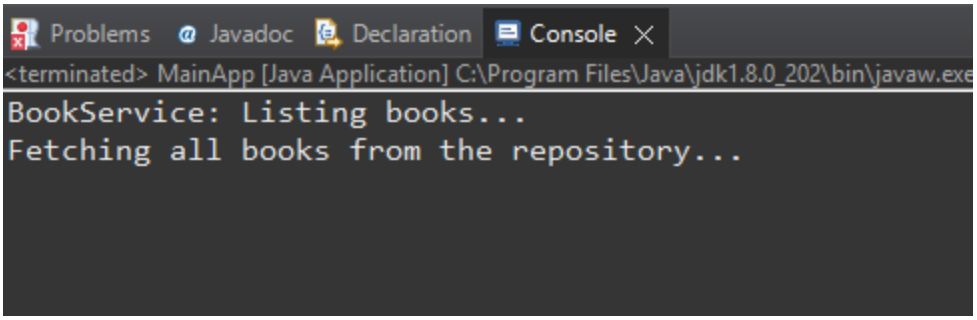
```
package com.library.repository;

public class BookRepository {
    public void getAllBooks() {
        System.out.println("Fetching all books from the repository...");
    }
}
```

MainApp.java

```
package com.library.demo;

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

```
public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applica

        BookService bookService = (BookService) context.getBean("bookService")
        bookService.listBooks();
    }
}
```

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**.
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.

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">

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

   <!-- Service Bean with constructor injection -->
   <bean id="bookService" class="com.library.service.BookService">
      <constructor-arg ref="bookRepository"/>
   </bean>

</beans>
```

BookRepository.java

```
package com.library.repository;

public class BookRepository {
   public void saveBook(String bookName) {
      System.out.println("BookRepository: Book \"" + bookName + "\" has been sa
   }
}
```

BookService.java

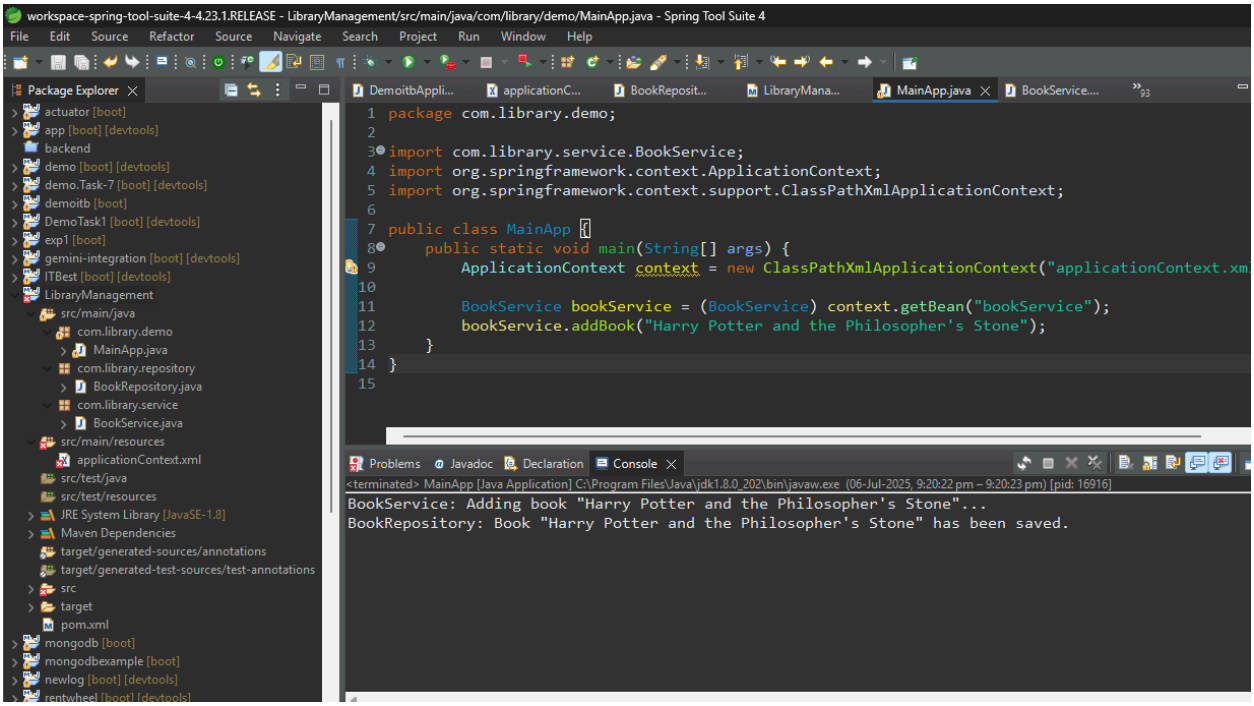
```
package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Constructor for DI
    public BookService(BookRepository bookRepository) {
        this.bookRepository = bookRepository;
    }

    public void addBook(String bookName) {
        System.out.println("BookService: Adding book \"" + bookName + "\"...");
        bookRepository.saveBook(bookName);
    }
}
```

MainApp.java

```
package com.library.demo;

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

public class MainApp {
    public static void main(String[] args) {
        ApplicationContext context = new ClassPathXmlApplicationContext("applica

        BookService bookService = (BookService) context.getBean("bookService")
        bookService.addBook("Harry Potter and the Philosopher's Stone");
```

```
    }
}
```

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.

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:**

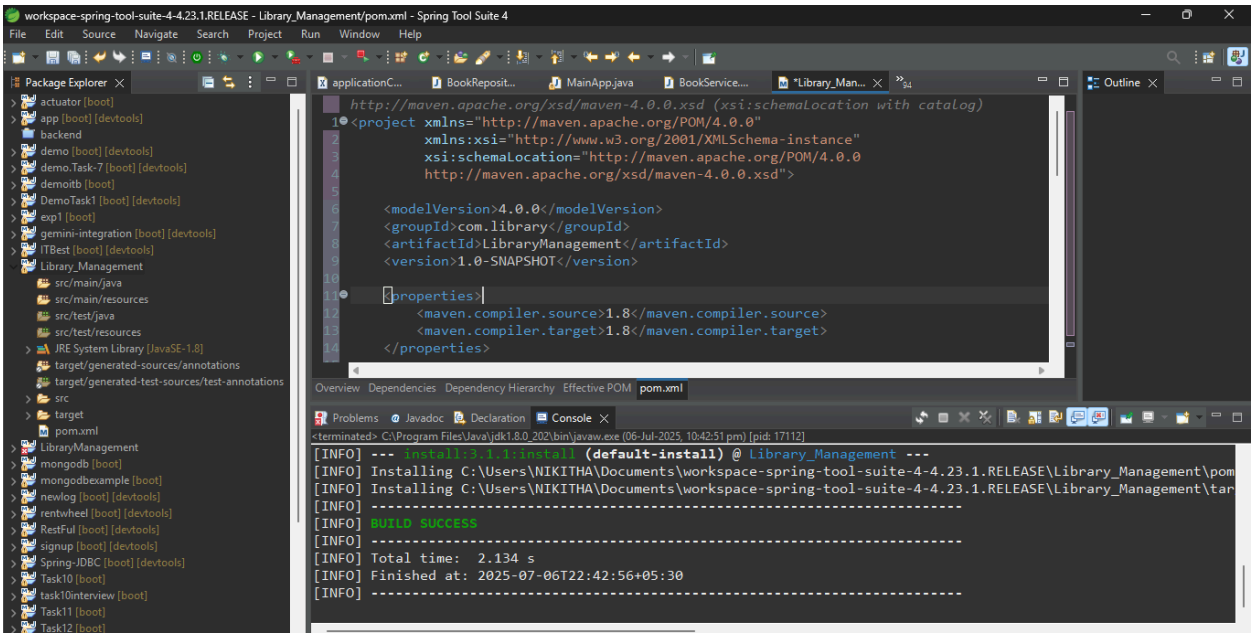
- Configure the Maven Compiler Plugin for Java version 1.8 in the pom.xml file.

pom.xml

```
<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>
        <!-- Spring Core Container -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-context</artifactId>
            <version>5.3.30</version>
        </dependency>

        <!-- Spring AOP -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>5.3.30</version>
        </dependency>

        <!-- Spring Web MVC -->
```

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.3.30</version>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <!-- Maven Compiler Plugin to set Java version -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.10.1</version>
                <configuration>
                    <source>1.8</source>
                    <target>1.8</target>
                </configuration>
            </plugin>
        </plugins>
    </build>

</project>
```

Output:


workspace-spring-tool-suite-4-4.23.1.RELEASE - Library_Management/pom.xml - Spring Tool Suite 4
File Edit Source Navigate Search Project Run Window Help
Package Explorer
actuator [boot]
app [boot] [devtools]
backend
demo [boot] [devtools]
demo.Task-7 [boot] [devtools]
demoitb [boot]
DemoTask1 [boot] [devtools]
exp1 [boot]
gemini-integration [boot] [devtools]
ITBest [boot] [devtools]
Library_Management
src/main/java
src/main/resources
src/test/java
src/test/resources
JRE System Library [JavaSE-1.8]
target/generated-sources/annotations
target/generated-test-sources/test-annotations
src
target
pom.xml
LibraryManagement
mongodb [boot]
mongodbexample [boot]
newlog [boot] [devtools]
rentwheel [boot] [devtools]
RestFul [boot] [devtools]
signup [boot] [devtools]
Spring-JDBC [boot] [devtools]
Task10 [boot]
task10interview [boot]
Task11 [boot]
Task12 [boot]
applicationC... BookReposit... MainApp.java BookService.... *Library_Man...
Outline
http://maven.apache.org/xsd/maven-4.0.0.xsd (xsi:schemaLocation with catalog)
<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>
Overview Dependencies Dependency Hierarchy Effective POM pom.xml
Problems Javadoc Declaration Console
<terminated> C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (06-Jul-2025, 10:42:51 pm) [pid: 17112]
[INFO] --- install:3.1.1:install (default-install) @ Library_Management ---
[INFO] Installing C:\Users\NIKITHA\Documents\workspace-spring-tool-suite-4-4.23.1.RELEASE\Library_Management\pom
[INFO] Installing C:\Users\NIKITHA\Documents\workspace-spring-tool-suite-4-4.23.1.RELEASE\Library_Management\tar
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 2.134 s
[INFO] Finished at: 2025-07-06T22:42:56+05:30
[INFO] --