**SOLUTION WEEK 3**

**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.

CODE

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibraryManagement {

    public static void main( String[] args )

    {

        System.out.println("Starting the Application");

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

    }

}

package com.example.service;

public class BookService {

     public  BookService() {

        System.out.println("This is BookService");

    }

}

package com.example.repository;

public class BookRepository {

    public BookRepository() {

        System.out.println("This is BookRepository");

    }

}

<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 class="com.example.service.BookService">

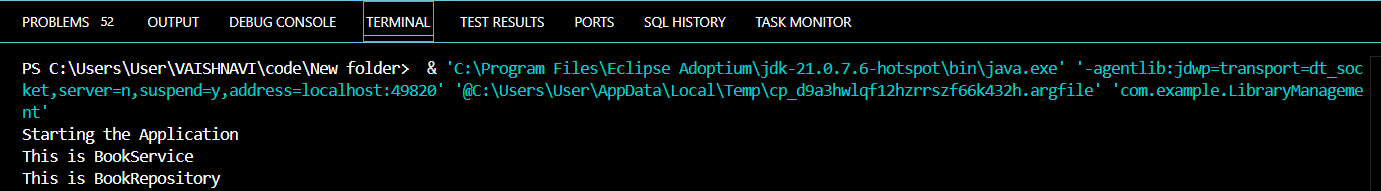
    </bean>

    <bean class="com.example.repository.BookRepository">

    </bean>

</beans>

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.

CODE

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.service.BookService;

public class LibraryManagement {

    public static void main( String[] args )

    {

        System.out.println("Starting the Application");

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

        BookService service = context.getBean(BookService.class);

        service.printRepository();

        System.out.println("Injection Done");

    }

}

package com.example.service;

import com.example.repository.BookRepository;

public class BookService {

    private BookRepository repository;

    public BookService() {

        System.out.println("BookService Created");

    }

    public void setrepository(BookRepository repository) {

        this.repository = repository;

    }

    public void printRepository() {

        System.out.println(repository);

    }

}

package com.example.repository;

public class BookRepository {

    public BookRepository() {

        System.out.println("BookRepository Created");

    }

    @Override

    public String toString(){

        return "BookRepository Injected";

    }

}

<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 class="com.example.repository.BookRepository" id="repository">

    </bean>

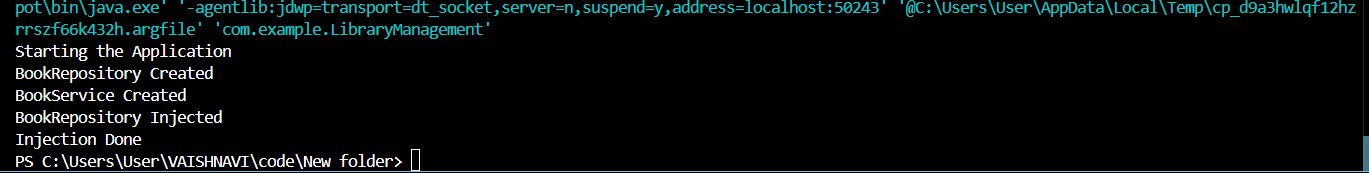
      <bean class="com.example.service.BookService">

        <property name="repository" ref="repository" />

    </bean>

</beans>

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.

CODE

pom.xml

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->

<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-context</artifactId>

    <version>6.2.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->

<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-aop</artifactId>

    <version>6.2.8</version>

</dependency>

<!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->

<dependency>

    <groupId>org.springframework</groupId>

    <artifactId>spring-webmvc</artifactId>

    <version>6.2.8</version>

</dependency>

  </dependencies>

    <build>

    <plugins>

      <plugin>

        <groupId>org.apache.maven.plugins</groupId>

        <artifactId>maven-compiler-plugin</artifactId>

        <configuration>

          <source>1.8</source>

          <target>1.8</target>

        </configuration>

      </plugin>

    </plugins>

  </build>

package com.example;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.example.service.BookService;

public class LibraryManagement {

    public static void main( String[] args )

    {

        System.out.println("Starting the Application");

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

        BookService service = context.getBean(BookService.class);

        service.printRepository();

        System.out.println("Injection Done");

    }

}

OUTPUT

