EXERCISE 1(Configuring a Basic Spring Application)

//BOOKREPOSITORY CLASS

package org.library;  
  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

//BOOKSERVICE CLASS

package org.library;  
  
public class BookService{  
 private BookRepository bookRepository;  
  
 public BookService(BookRepository bookRepository) {  
 System.*out*.println("BookService Constructor is called.....");  
 this.bookRepository = bookRepository;  
 }  
  
public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
}  
}

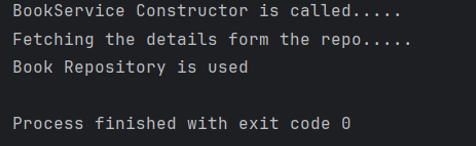
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=(BookService) context.getBean("bs1");  
  
 bookService.display();  
 }  
}

//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="bs1" class="org.library.BookService">  
 <constructor-arg ref="br1"/>  
  
</bean>  
 <bean id="br1" class="org.library.BookRepository"></bean>  
  
</beans>

//OUTPUT



EXERCISE 2(Implementing Dependency Injection)

//BOOKSERVICE CLASS

package org.library;  
  
public class BookService {  
 private BookRepository bookRepository;  
 public BookRepository getBookRepiository() {  
 return bookRepository;  
 }  
  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
 public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
 }  
  
}

//BOOKREPOSITORY CLASS

package org.library;  
  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

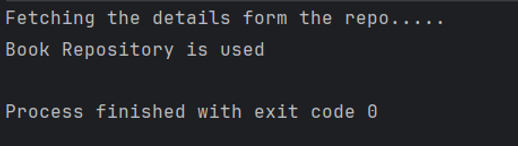
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=(BookService) context.getBean("bs1");  
  
 bookService.display();  
 }  
}

//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="bs1" class="org.library.BookService">  
 <property name="bookRepository" ref="br1"/>  
  
</bean>  
 <bean id="br1" class="org.library.BookRepository"></bean>  
  
</beans>

//OUTPUT



EXCERCISE 4(Creating and Configuring a Maven Project)

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>org.example</groupId>  
 <artifactId>untitled</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>untitled</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>3.8.1</version>  
 <scope>test</scope>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>6.2.7</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-aop -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>6.2.7</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>6.2.7</version>  
 </dependency>  
 <!-- https://mvnrepository.com/artifact/org.apache.maven.plugins/maven-compiler-plugin -->  
 <dependency>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.8.1</version>  
 </dependency>  
  
 </dependencies>  
</project>

EXERCISE 3(IMPLEMENTING LOGGING WITH SPRING AOP) EXERCISE 8 ALSO IMPLEMENTED(Implementing Basic AOP with Spring USING AUTO PROXY)

//REPOSITORY CLASS

package org.library;  
  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

//BOOK SERVICE

package org.library;  
  
public class BookService{  
 private BookRepository bookRepository;  
  
 public BookService(BookRepository bookRepository) {  
 System.*out*.println("BookService Constructor is called.....");  
 this.bookRepository = bookRepository;  
 }  
  
public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
}  
}

//LOGGING ASPECT CLASS

package org.library;  
  
public class LoggingAspect {  
 public void beforeLoggingAspect(){  
 System.*out*.println("Before Logging Aspect called....");  
  
 }  
 public void afterLoggingAspect(){  
 System.*out*.println("After Logging Aspect");  
 }  
}

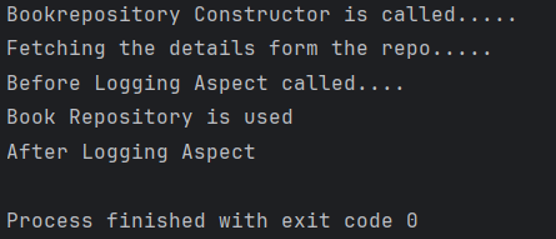
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=(BookService) context.getBean("bs1");  
  
 bookService.display();  
 }  
}

//APPLICATION CONTEXT.XML

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 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  
 http://www.springframework.org/schema/aop  
 http://www.springframework.org/schema/aop/spring-aop.xsd">  
 <aop:aspectj-autoproxy/> //// EXCECRCISE 8 AUTO PROXY  
  
 <bean id="br1" class="org.library.BookRepository"/>  
  
 <bean id="bs1" class="org.library.BookService">  
 <constructor-arg ref="br1"/>  
 </bean>  
 <bean id="LoggingAspect" class="org.library.LoggingAspect"/>  
 <aop:config>  
 <aop:aspect ref="LoggingAspect">  
 <aop:pointcut id="repoMethods" expression="execution(\* org.library.BookRepository.\*(..))"/>  
 <aop:before method="beforeLoggingAspect" pointcut-ref="repoMethods"/>  
 <aop:after method="afterLoggingAspect" pointcut-ref="repoMethods"/>  
 </aop:aspect>  
 </aop:config>  
  
</beans>

//OUTPUT



EXERCISE 5(CONFIGURING THE SPRING IOC CONTAINER)

//BOOK REPOSITORY CLASS

package org.library;  
  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

//BOOK SERVICE CLASS

package org.library;  
  
public class BookService{  
 private BookRepository bookRepository;  
 public void setBookRepository(BookRepository bookRepository) {  
 System.*out*.println("Set bookrepository is called");  
 this.bookRepository = bookRepository;  
 }  
  
 public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
}  
}

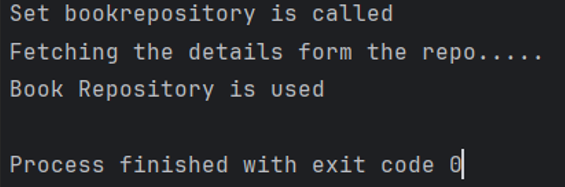
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=(BookService) context.getBean("bs1");  
  
 bookService.display();  
 }  
}

// APPLICATION CONTEXT.XML

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:aop="http://www.springframework.org/schema/aop"  
 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  
 http://www.springframework.org/schema/aop  
 http://www.springframework.org/schema/aop/spring-aop.xsd">  
 <bean id="br1" class="org.library.BookRepository"/>  
  
 <bean id="bs1" class="org.library.BookService">  
  
 <property name="bookRepository" ref="br1"/>  
 </bean>  
   
</beans>

//OUTPUT



EXERCISE 6(CONFIGURING BEANS WITH ANNOTATIONS)

//BOOKREPOSITORY CLASS

package org.library;  
  
import org.springframework.stereotype.Component;  
import org.springframework.stereotype.Repository;  
@Component  
@Repository  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

//BOOKSERVICE CLASS

package org.library;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Component;  
import org.springframework.stereotype.Service;  
@Service  
@Component  
public class BookService{  
 @Autowired  
 private BookRepository bookRepository;  
 public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
}  
}

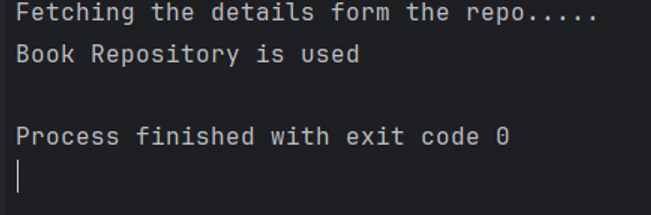
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=context.getBean(BookService.class);  
  
 bookService.display();  
 }  
}

//APPLICATION CONTEXT.XML

<?xml version="1.0" encoding="UTF-8"?>  
<beans xmlns="http://www.springframework.org/schema/beans"  
 xmlns:context="http://www.springframework.org/schema/context"  
 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  
 http://www.springframework.org/schema/context  
 http://www.springframework.org/schema/context/spring-context.xsd">  
 <context:component-scan base-package="org.library"/>  
</beans>

//OUTPUT



EXERCISE 7(Implementing Constructor and Setter Injection)

// BOOK REPOSITORY CLASS

package org.library;  
  
  
public class BookRepository {  
 public void getDetails(){  
 System.*out*.println("Book Repository is used");  
 }  
}

//BOOK SERVICE CLASS

package org.library;  
public class BookService{  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 System.*out*.println("Setter is used");  
 }  
  
 public BookService(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 System.*out*.println("Constructor is called");  
 }  
  
 private BookRepository bookRepository;  
 public void display(){  
 System.*out*.println("Fetching the details form the repo.....");  
 bookRepository.getDetails();  
}  
}

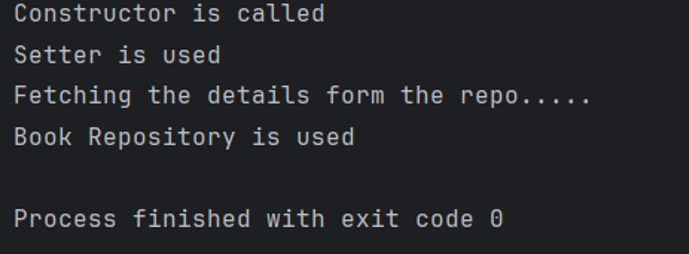
//MAIN CLASS

package org.library;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
public class Main  
{  
 public static void main( String[] args )  
 {  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService bookService=(BookService) context.getBean("bs1");  
  
 bookService.display();  
 }  
}

//APPLICATION CONTEXT.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="br1" class="org.library.BookRepository"/>  
 <bean id="bs1" class="org.library.BookService">  
 <property name="bookRepository" ref="br1"/>  
 <constructor-arg ref="br1"/>  
 </bean>  
  
</beans>

//OUTPUT



EXERCISE 9

//BookController Class

package com.example.LibraryManagement.controller;  
import com.example.LibraryManagement.Book;  
import com.example.LibraryManagement.service.BookService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
import java.util.List;  
@RestController  
@RequestMapping("/books")  
public class BookController{  
 @Autowired  
 private BookService bookService;  
 @GetMapping  
 public List<Book> getAllBooks() {  
 return bookService.getAllBooks();  
 }  
 @PostMapping  
 public Book addBook(@RequestBody Book book) {  
 return bookService.addBook(book);  
 }  
  
 @PutMapping("/{id}")  
 public Book updateBook(@PathVariable int id, @RequestBody Book book) {  
 return bookService.updateBook(id, book);  
 }  
}

//BookService

package com.example.LibraryManagement.service;  
  
import com.example.LibraryManagement.repository.BookRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import com.example.LibraryManagement.Book;  
import java.util.List;  
  
@Service  
public class BookService {  
 @Autowired  
 private BookRepository bookRepository;  
 public List<Book> getAllBooks(){  
 return bookRepository.findAll();  
 }  
 public Book addBook(Book book) {  
 return bookRepository.save(book);  
 }  
 public Book updateBook(int id, Book updatedBook)  
 {  
 List<Book> books = bookRepository.findAll();  
  
 for (Book book : books) {  
 if (book.getId()==(id)) {  
 book.setTitle(updatedBook.getTitle());  
 return bookRepository.save(book);  
 }  
 }  
 return null;  
 }  
  
}

//Book class

package com.example.LibraryManagement;  
  
import jakarta.persistence.Entity;  
import jakarta.persistence.GeneratedValue;  
import jakarta.persistence.GenerationType;  
import jakarta.persistence.Id;  
  
@Entity  
public class Book {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
 private String title;  
  
 public String getTitle() {  
 return title;  
 }  
  
 public void setTitle(String title) {  
 this.title = title;  
 }  
  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
}

//Bookrepository Class

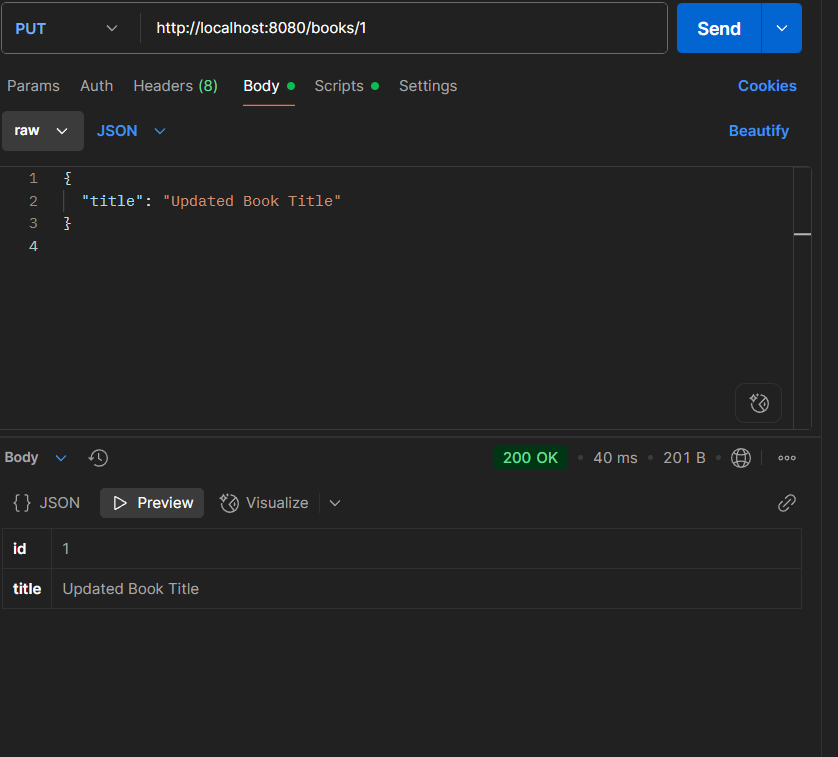
package com.example.LibraryManagement.repository;  
import com.example.LibraryManagement.Book;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
@Repository  
public interface BookRepository extends JpaRepository<Book,Integer> {  
}

//Library Management Class

package com.example.LibraryManagement;  
  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(LibraryManagementApplication.class, args);  
 }  
  
}

//Application properties

spring.datasource.url=jdbc:h2:mem:testdb  
spring.datasource.username=sa  
spring.datasource.password=  
spring.h2.console.enabled=true  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

//output

