**WEEK-3**

**Spring Core and Maven**

**Exercise 1: Configuring a Basic Spring Application**

* Create a new maven project with group Id as com.library and artifact Id as LibraryManagement.
* Add dependencies in the pom.xml file.
* Create a package **com.library.service** and add a class **BookService**.
* Create a package **com.library.repository** and add a class **BookRepository**.
* Create an ApplicationContext.xml in the resources directory.
* Create MainApp.java to test the configurations.

**BookRepository.java**

package com.library.repository;

public class BookRepository {

   public void displayRepository() {

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

   }

}

**BookService.java**

package com.library.service;

public class BookService {

   public void displayService() {

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

   }

}

**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

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

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

   <bean id="bookService" class="com.library.service.BookService" />

</beans>

**pom.xml**

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="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>

 <dependencies>

 <dependency>

       <groupId>org.springframework</groupId>

       <artifactId>spring-context</artifactId>

       <version>5.3.32</version>

   </dependency>

 </dependencies>

</project>

**MainApp.java**

package com.library;

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("applicationContext.xml");

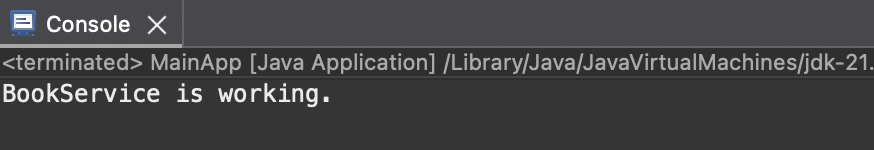
       BookService service = (BookService) context.getBean("bookService");

       service.displayService();

}

}

**Output:**



**Exercise 2: Implementing Dependency Injection**

* Update the files of the above maven 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">

   <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 void displayBooks() {

       System.out.println("BookRepository: Displaying list of books...");

   }

}

**BookService.java**

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

   private BookRepository bookRepository;

   public void setBookRepository(BookRepository bookRepository) {

       this.bookRepository = bookRepository;

   }

   public void listBooks() {

       System.out.println("BookService: Calling BookRepository...");

       bookRepository.displayBooks();

   }

}

**MainApp.java**

package com.library;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

import com.library.service.BookService;

public class MainApp {

   public static void main(String[] args) {

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

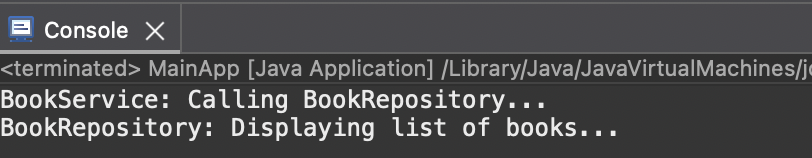
       BookService bookService = context.getBean("bookService", BookService.class);

       bookService.listBooks();

   }

}

**Output:**

****

**Exercise 4: Creating and Configuring a Maven Project**

* Add dependencies of Spring Context, Spring AOP, and Spring WebMVC.
* 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

                            https://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 Context -->

       <dependency>

           <groupId>org.springframework</groupId>

           <artifactId>spring-context</artifactId>

           <version>5.3.22</version>

       </dependency>

       <!-- Spring AOP -->

       <dependency>

           <groupId>org.springframework</groupId>

           <artifactId>spring-aop</artifactId>

           <version>5.3.22</version>

       </dependency>

       <!-- Spring Web MVC -->

       <dependency>

           <groupId>org.springframework</groupId>

           <artifactId>spring-webmvc</artifactId>

           <version>5.3.22</version>

       </dependency>

   </dependencies>

   <build>

       <plugins>

           <!-- Maven Compiler Plugin -->

           <plugin>

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

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

               <version>3.8.1</version>

               <configuration>

                   <source>1.8</source>

                   <target>1.8</target>

               </configuration>

           </plugin>

       </plugins>

   </build>

</project>

**Spring Data JPA - Quick Example**

* Create a new maven project and add dependencies in pom.xml file.
* Add controller, repository and model packages in the src/main/java directory.

**SpringJpaApplication.java**

package com.example.springjpa;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class SpringJpaApplication {

   public static void main(String[] args) {

       SpringApplication.run(SpringJpaApplication.class, args);

   }

}

**Book.java**

package com.example.springjpa.model;

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 Long id;

   private String title;

   private String author;

   // Getters

   public Long getId() {

       return id;

   }

   public String getTitle() {

       return title;

   }

   public String getAuthor() {

       return author;

   }

   public void setId(Long id) {

       this.id = id;

   }

   public void setTitle(String title) {

       this.title = title;

   }

   public void setAuthor(String author) {

       this.author = author;

   }

}

**BookRepository.java**

package com.example.springjpa.repository;

import com.example.springjpa.model.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**BookController.java**

package com.example.springjpa.controller;

import com.example.springjpa.model.Book;

import com.example.springjpa.repository.BookRepository;

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 BookRepository bookRepository;

   @PostMapping

   public Book addBook(@RequestBody Book book) {

       return bookRepository.save(book);

   }

   @GetMapping

   public List<Book> getAllBooks() {

       return bookRepository.findAll();

   }

}

**application.properties**

server.port=9090

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.database-platform=org.hibernate.dialect.H2Dialect

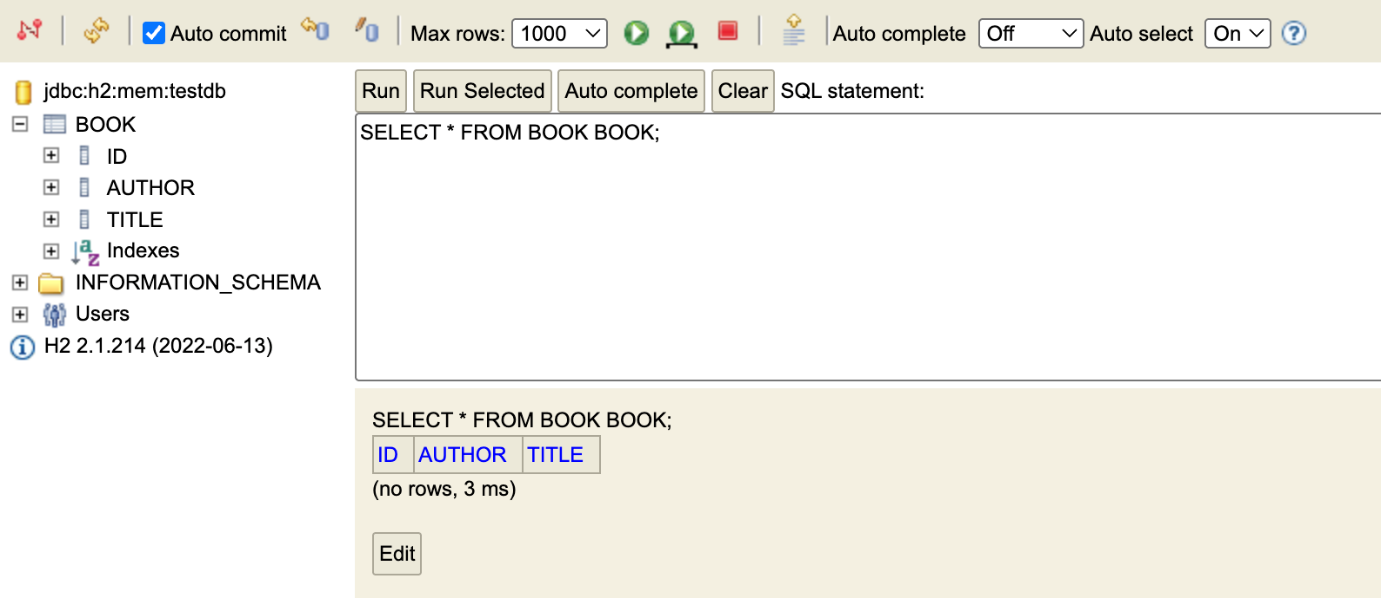
spring.jpa.hibernate.ddl-auto=update

spring.sql.init.mode=always

spring.jpa.defer-datasource-initialization=true

**Output:**

The output of table “BOOK” can be seen in the h2 console.

****