**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.

**BookRepository.java**

package com.library.librarymanagement;

import org.springframework.stereotype.Repository;

@Repository

public class BookRepository {

    public String getBookTitle() {

        return "Spring Boot in Practice";

    }

}

**BookService.java:**

package com.library.librarymanagement;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

@Service

public class BookService {

    private final BookRepository bookRepository;

    @Autowired

    public BookService(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void showBook() {

        System.out.println("Book Title: " + bookRepository.getBookTitle());

    }

}

**LibrarymanagementApplication.java:**

package com.library.librarymanagement;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class LibrarymanagementApplication implements CommandLineRunner {

    @Autowired

    private BookService bookService;

    public static void main(String[] args) {

        SpringApplication.run(LibrarymanagementApplication.class, args);

    }

    @Override

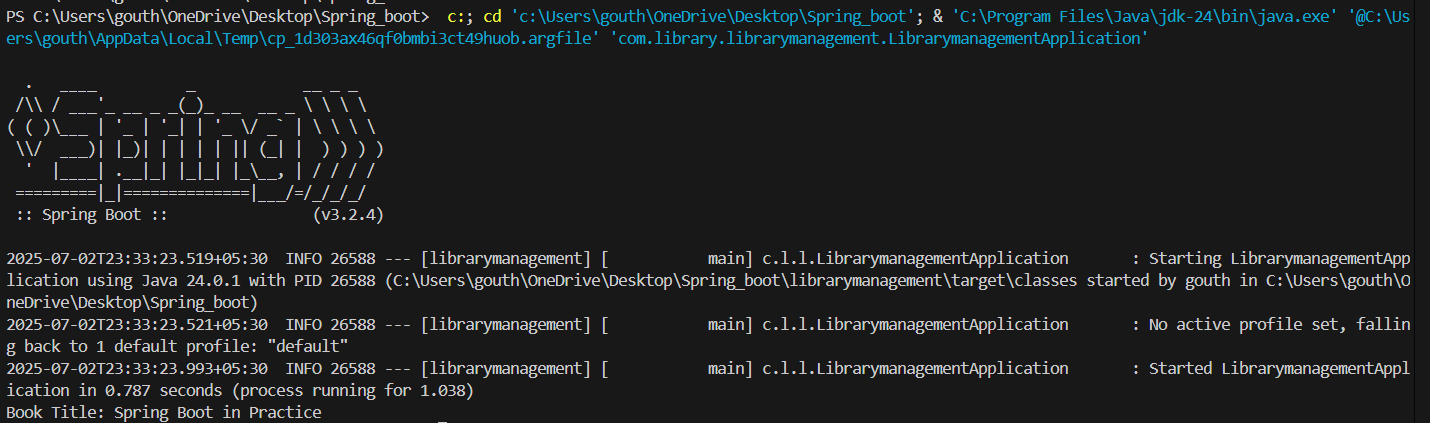
    public void run(String... args) throws Exception {

        bookService.showBook();  // This runs automatically after app starts

    }

}

**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.

**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">

<!-- Define BookRepository bean -->

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

<!-- Define BookService bean and inject BookRepository using setter -->

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

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

</bean>

</beans>

**BookService.java:**

package com.library.librarymanagement;

public class BookService {

private BookRepository bookRepository;

public void setBookRepository(BookRepository bookRepository) {

this.bookRepository = bookRepository;

}

public void showBook() {

System.out.println("Book Title: " + bookRepository.getBookTitle());

}

}

**BookRepository.java:**

package com.library.librarymanagement;

public class BookRepository {

public String getBookTitle() {

return "Spring in Action";

}

}

**LibrarymanagementApplication.java:**

package com.library.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibrarymanagementApplication {

public static void main(String[] args) {

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

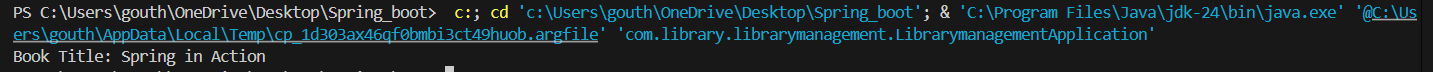
BookService bookService = context.getBean("bookService", BookService.class);

bookService.showBook();

}

}

**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.

**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>0.0.1-SNAPSHOT</version>

  <packaging>jar</packaging>

  <name>librarymanagement</name>

  <description>Library Management Spring Boot App</description>

  <parent>

    <groupId>org.springframework.boot</groupId>

    <artifactId>spring-boot-starter-parent</artifactId>

    <version>3.2.4</version> <!-- or latest compatible -->

    <relativePath/> <!-- lookup parent from repository -->

  </parent>

  <properties>

    <java.version>17</java.version> <!-- ⚠️ Use 17 or 21, not 24 -->

  </properties>

  <dependencies>

    <!-- 🟢 Spring Boot Starter -->

    <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter</artifactId>

    </dependency>

    <!-- 🧪 Spring Boot Testing (optional) -->

    <dependency>

      <groupId>org.springframework.boot</groupId>

      <artifactId>spring-boot-starter-test</artifactId>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- 🔧 Spring Boot Maven Plugin -->

      <plugin>

        <groupId>org.springframework.boot</groupId>

        <artifactId>spring-boot-maven-plugin</artifactId>

      </plugin>

    </plugins>

  </build>

</project>

**BookRepository.java:**

package com.library.librarymanagement;

public class BookRepository {

    public String getBookTitle() {

        return "This is Exercise 3";

    }

}

**BookService.java:**

package com.library.librarymanagement;

public class BookService {

    private BookRepository bookRepository;

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void showBook() {

        System.out.println("Book Title: " + bookRepository.getBookTitle());

    }

}

**LibrarymanagementApplication.java:**

package com.library.librarymanagement;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class LibrarymanagementApplication {

    public static void main(String[] args) {

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

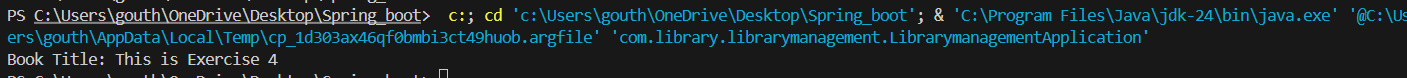
        BookService bookService = context.getBean("bookService", BookService.class);

        bookService.showBook();

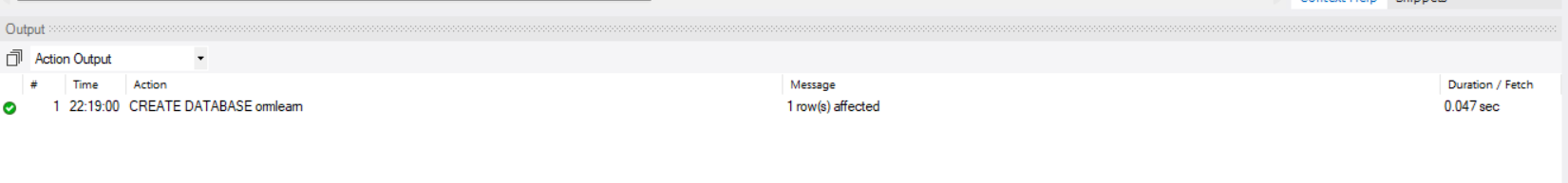
    }

}

**Output:**

****

1. **Spring Data JPA - Quick Example**create schema ormlearn;

****

**application.properties:**

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace

logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-20.20thread %5p %-25.25logger{25} %25M %4L %m%n

# Database

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

# Hibernate

spring.jpa.hibernate.ddl-auto=validate

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**Create table:**

create table country(

code varchar(2) primary key,

name varchar(50)

);

insert into country values ('IN', 'India');

insert into country values ('US', 'United States of America');

**Country.java:**

package com.cognizant.ormlearn.model;

import javax.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

// Getters & Setters

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

**CountryRepository.java:**

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

**CountryService.java:**

package com.cognizant.ormlearn.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryService countryService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("countries={}", countries);

LOGGER.info("End");

}

}

**Output:**

****

1. **Difference between JPA, Hibernate and Spring Data JPA**:

**pom.xml:**

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

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>orm-learn</name>

<description>Demo project for Spring Data JPA and Hibernate</description>

<parent>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-parent</artifactId>

<version>3.2.0</version>

</parent>

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-data-jpa</artifactId>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties:**

# Logging

logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=debug

logging.level.org.hibernate.type.descriptor.sql=trace

# DB Connection

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.jpa.hibernate.ddl-auto=update

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

**Employee.java:**

package com.cognizant.ormlearn.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

private int id;

@Column(name = "name")

private String name;

@Column(name = "salary")

private double salary;

public int getId() { return id; }

public void setId(int id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + "]";

}

}

**EmployeeRepository.java:**

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**EmployeeService.java:**

package com.cognizant.ormlearn.service;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.repository.EmployeeRepository;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee emp) {

employeeRepository.save(emp);

}

}

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Employee;

import com.cognizant.ormlearn.service.EmployeeService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static EmployeeService employeeService;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

employeeService = context.getBean(EmployeeService.class);

LOGGER.info("Inside main");

testAddEmployee();

}

private static void testAddEmployee() {

LOGGER.info("Start addEmployee");

Employee emp = new Employee();

emp.setId(1);

emp.setName("Siddu");

emp.setSalary(50000);

employeeService.addEmployee(emp);

LOGGER.info("End addEmployee");

}

}

**SQL Table:**

CREATE DATABASE IF NOT EXISTS ormlearn;

USE ormlearn;

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(100),

salary DOUBLE

);

**Output:**

