**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.

Code:

XML:

<?xml version="1.0" encoding="UTF-8"?>  
<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>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
  
</project>

Repository:

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.*out*.println("Book saved: " + title);  
 }  
}

Service:

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 addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);  
 }  
}

Main:

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) {  
 // Load context  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get the BookService bean  
 BookService bookService = context.getBean("bookService", BookService.class);  
  
 // Call method  
 bookService.addBook("Java Programming");  
 }  
}

OUTPUT:

A black screen with a black background

AI-generated content may be incorrect.

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.

CODE:

XML:

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) {

// Load Spring container

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

// Get the bookService bean

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

// Use it

bookService.addBook("Spring Framework Guide");

}

}

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">  
  
 <!-- BookRepository Bean -->  
 <bean id="bookRepository" class="com.library.repository.BookRepository" />  
  
 <!-- BookService Bean with DI -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <!-- Setter Injection using property tag -->  
 <property name="bookRepository" ref="bookRepository" />

</bean>  
</beans>

Repository:

package com.library.repository;  
  
public class BookRepository {  
 public void saveBook(String title) {  
 System.*out*.println("Book saved: " + title);  
 }  
}

Service:

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 addBook(String title) {  
 System.*out*.println("Adding book: " + title);  
 bookRepository.saveBook(title);   
 }  
}

Main:

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) {  
 // Load Spring container  
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 // Get the bookService bean  
 BookService bookService = context.getBean("bookService", BookService.class);  
  
 // Use it  
 bookService.addBook("Spring Framework Guide");  
 }  
}

OUTPUT:

A black screen with many small colored lights

AI-generated content may be incorrect.

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.

Code:

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>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
   
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.33</version>

</dependency>   
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <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>

OUTPUT:



**Spring Data JPA with Spring Boot, Hibernate:**

Spring Data JPA - Quick Example :

CODE:

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.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>2.7.5</version> <!-- Use 2.x or 3.x as needed -->  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-data-jpa</artifactId>  
 </dependency>  
 <dependency>  
 <groupId>com.mysql</groupId>  
 <artifactId>mysql-connector-j</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-devtools</artifactId>  
 <scope>runtime</scope>  
 </dependency>  
 <dependency>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-test</artifactId>  
 <scope>test</scope>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <plugin>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-maven-plugin</artifactId>  
 </plugin>  
 </plugins>  
 </build>  
</project>

APPLICATION PROPERTIES:

logging.level.com.cognizant=debug  
logging.level.org.springframework=warn

logging.level.org.hibernate=warn  
logging.level.org.springframework.boot=warn

logging.level.org.springframework.data=warn  
logging.level.org.apache=warn  
spring.main.banner-mode=off  
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=

spring.jpa.hibernate.ddl-auto=validate  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect

CLASS:

package com.cognizant.orm\_learn.model;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
import javax.persistence.\*;  
  
@Entity  
@Table(name="country")  
public class country {  
  
 @Id  
 @Column(name="co\_code")  
 private String code;  
  
 @Column(name="co\_name")  
 private String name;  
  
 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 + "]";  
 }  
}

REPOSITORY:

package com.cognizant.orm\_learn.repository;  
  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
import com.cognizant.orm\_learn.model.country;  
  
@Repository  
public interface CountryRepository extends JpaRepository<country, String> {  
}

SERVICE:

package com.cognizant.orm\_learn.service;  
  
import java.util.List;

import javax.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import com.cognizant.orm\_learn.model.country;  
import com.cognizant.orm\_learn.repository.CountryRepository;  
  
@Service  
public class CountryService {  
  
 @Autowired  
 private CountryRepository countryRepository;  
  
 @Transactional  
 public List<country> getAllCountries() {  
 return countryRepository.findAll();  
 }  
}

MAIN:

package com.cognizant.orm\_learn;  
  
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.orm\_learn.model.country;  
import com.cognizant.orm\_learn.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");  
  
 }  
}

SQL(Database):

CREATE SCHEMA ormlearn;

USE ormlearn;

CREATE TABLE country (

co\_code VARCHAR(2) PRIMARY KEY,

co\_name VARCHAR(50)

);

INSERT INTO country VALUES ('IN', 'India');

INSERT INTO country VALUES ('US', 'United States of America');

OUTPUT:

A screenshot of a computer program

AI-generated content may be incorrect.

Difference between JPA, Hibernate and Spring Data JPA:

CODE:

APPLICATION PROPERTIES:

spring.datasource.url=jdbc:mysql://localhost:3306/Employee  
spring.datasource.username=root  
spring.datasource.password=  
  
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect  
  
logging.level.org.springframework=info  
logging.level.com.cognizant.orm\_learn=debug

CLASS:

package com.cognizant.orm\_learn.model;  
  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.GeneratedValue;  
import javax.persistence.GenerationType;  
import javax.persistence.Table;  
import javax.persistence.Column;  
  
@Entity  
@Table(name = "employee")  
public class Employee {  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.*IDENTITY*)  
 private int id;  
  
 @Column(name = "name", nullable = false)  
 private String name;  
  
 @Column(name = "department", nullable = false)  
 private String department;  
 public Employee() {  
 }  
  
 public Employee(int id, String name, String department) {  
 this.id = id;  
 this.name = name;  
 this.department = department;  
 }  
  
  
 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 String getDepartment() {  
 return department;  
 }  
  
 public void setDepartment(String department) {  
 this.department = department;  
 }  
  
 @Override  
 public String toString() {  
 return "Employee{" +  
 "id=" + id +  
 ", name='" + name + '\'' +  
 ", department='" + department + '\'' +  
 '}';  
 }  
}

REPOSITORY:

package com.cognizant.orm\_learn.repository;  
  
import com.cognizant.orm\_learn.model.Employee;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;  
  
@Repository  
public interface EmployeeRepository extends JpaRepository<Employee, Integer> {  
   
}

SERVICE:

package com.cognizant.orm\_learn.service;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.repository.EmployeeRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import javax.transaction.Transactional;  
import java.util.List;  
  
@Service  
public class EmployeeService {  
  
 @Autowired  
 private EmployeeRepository employeeRepository;  
  
 @Transactional  
 public void addEmployee(Employee employee) {  
 employeeRepository.save(employee); // no need for session/transaction  
 }  
  
 public List<Employee> getAll() {  
 return employeeRepository.findAll();  
 }  
}

MAIN:

package com.cognizant.orm\_learn;  
  
import java.util.List;  
  
import com.cognizant.orm\_learn.model.Employee;  
import com.cognizant.orm\_learn.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;  
  
import com.cognizant.orm\_learn.model.country;  
import com.cognizant.orm\_learn.service.CountryService;  
  
@SpringBootApplication  
public class OrmLearnApplication {  
  
 private static EmployeeService *employeeService*;  
  
 private static final Logger *LOGGER* = LoggerFactory.*getLogger*(OrmLearnApplication.class);  
  
 public static void main(String[] args) {  
 ApplicationContext context = SpringApplication.*run*(OrmLearnApplication.class, args);  
 *employeeService* = context.getBean(EmployeeService.class);  
 //testAdd();  
 *testGetAll*();  
 }  
  
 public static void testAdd() {  
 *LOGGER*.info("Start Add");  
 Employee e = new Employee();  
 e.setName("John");  
 e.setDepartment("HR");  
 *employeeService*.addEmployee(e);  
 *LOGGER*.info("End Add");  
 }  
  
 public static void testGetAll() {  
 *LOGGER*.info("Start Get");  
 List<Employee> list = *employeeService*.getAll();  
 *LOGGER*.debug("Employees={}", list);  
 *LOGGER*.info("End Get");  
 }  
}

OUTPUT:

