**Spring\_Core\_Maven exercises**

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

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">

    <!-- Define BookRepository bean -->

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

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

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

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

    </bean>

</beans>

package com.library.repository;

public class BookRepository {

    public void saveBook(String bookName) {

        System.out.println("Book saved: " + bookName);

    }

}

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter for dependency injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String name) {

        System.out.println("Adding book: " + name);

        bookRepository.saveBook(name);

    }

}

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) {

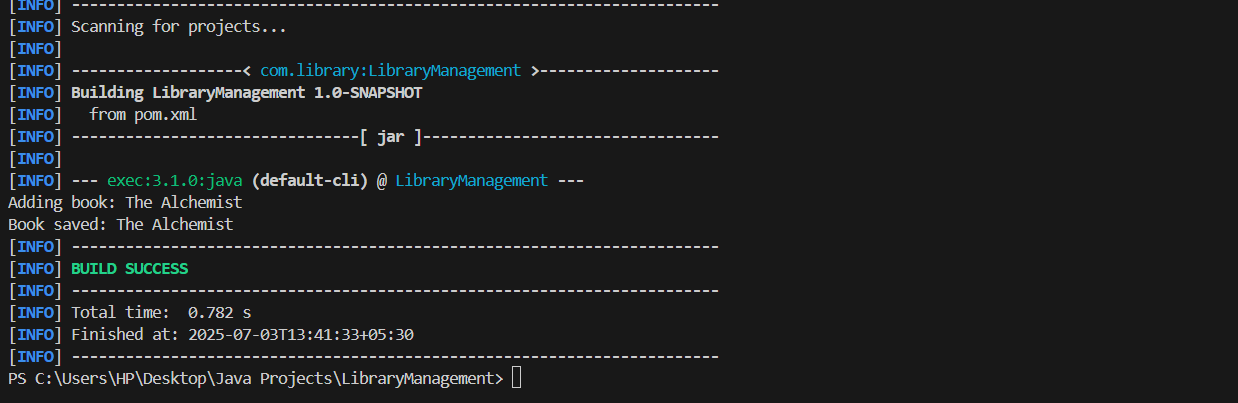
        ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");

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

        bookService.addBook("The Alchemist");

    }

}



**Exercise 2: Implementing Dependency Injection**

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">

    <!-- Define BookRepository bean -->

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

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

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

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

    </bean>

</beans>

package com.library.repository;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

    private List<String> books = new ArrayList<>();

    public void addBook(String bookName) {

        books.add(bookName);

    }

      public List<String> getBooks() {

        return books;

    }

}

package com.library.service;

import com.library.repository.BookRepository;

public class BookService {

    private BookRepository bookRepository;

    // Setter method for Spring Dependency Injection

    public void setBookRepository(BookRepository bookRepository) {

        this.bookRepository = bookRepository;

    }

    public void addBook(String bookName) {

        bookRepository.addBook(bookName);

    }

    public void listBooks() {

        bookRepository.getBooks().forEach(System.out::println);

    }

}

package com.library;

import com.library.service.BookService;

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);

        // Sample usage

        bookService.addBook("The Catcher in the Rye");

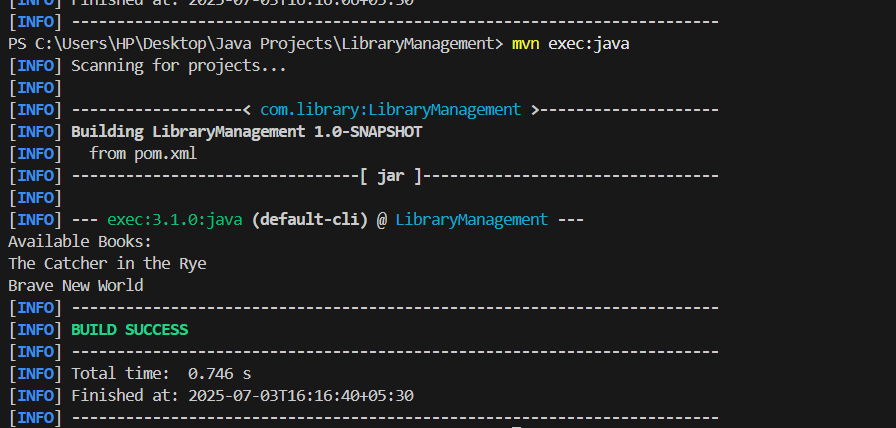
        bookService.addBook("Brave New World");

        System.out.println("Available Books:");

        bookService.listBooks();

    }

}



**Exercise 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>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>

        <!-- Spring Core & Context -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-context</artifactId>

            <version>5.3.36</version>

        </dependency>

        <!-- Spring AOP -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-aop</artifactId>

            <version>5.3.36</version>

        </dependency>

        <!-- Spring WebMVC -->

        <dependency>

            <groupId>org.springframework</groupId>

            <artifactId>spring-webmvc</artifactId>

            <version>5.3.36</version>

        </dependency>

        <!-- Servlet API (Required for WebMVC) -->

        <dependency>

            <groupId>javax.servlet</groupId>

            <artifactId>javax.servlet-api</artifactId>

            <version>4.0.1</version>

            <scope>provided</scope>

        </dependency>

    </dependencies>

    <build>

        <plugins>

            <!-- Compiler Plugin for Java 1.8 -->

            <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>

            <!-- Exec Plugin for running main class -->

            <plugin>

                <groupId>org.codehaus.mojo</groupId>

                <artifactId>exec-maven-plugin</artifactId>

                <version>3.1.0</version>

                <configuration>

                    <mainClass>com.library.LibraryManagementApplication</mainClass>

                </configuration>

            </plugin>

        </plugins>

    </build>

</project>

package com.library.repository;

import java.util.ArrayList;

import java.util.List;

public class BookRepository {

    private final List<String> books = new ArrayList<>();

    public void addBook(String book) {

        books.add(book);

    }

    public List<String> getBooks() {

        return books;

    }

}

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 book) {

        bookRepository.addBook(book);

    }

    public void listBooks() {

        System.out.println("Books in Library:");

        bookRepository.getBooks().forEach(System.out::println);

    }

}

package com.library;

import com.library.repository.BookRepository;

import com.library.service.BookService;

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);

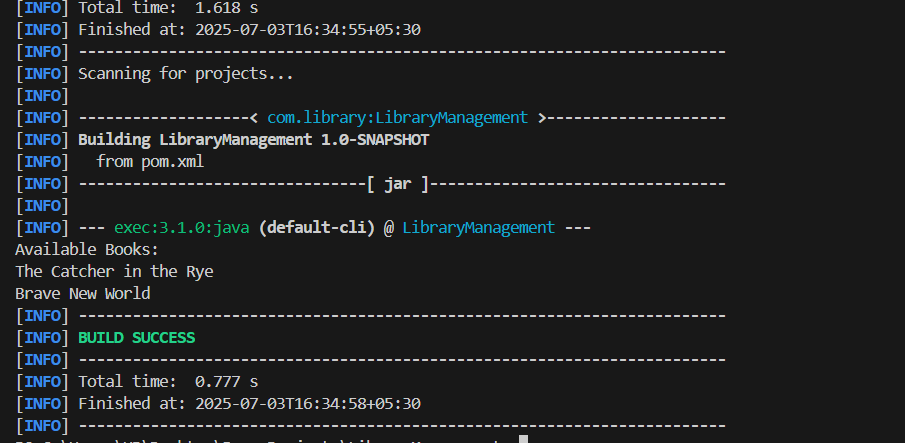
        bookService.addBook("The Alchemist");

        bookService.addBook("Clean Code");

        bookService.listBooks();

    }

}



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

**Spring Data JPA - Quick Example**Country.java

package com.cognizant.ormlearn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

*@Entity*

*@Table*(name = "country")

public class Country {

*@Id*

*@Column*(name = "co\_code")

private String code;

*@Column*(name = "co\_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.boot.autoconfigure.domain.EntityScan;

import org.springframework.context.ApplicationContext;

import org.springframework.data.jpa.repository.config.EnableJpaRepositories;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

*@SpringBootApplication*

*@EntityScan*("com.cognizant.ormlearn.model")

*@EnableJpaRepositories*("com.cognizant.ormlearn.repository")

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);

***LOGGER***.info("Inside main");

*countryService* = context.getBean(CountryService.class);

*testGetAllCountries*();

}

private static void testGetAllCountries() {

System.***out***.println("Start");

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

for (Country c : countries) {

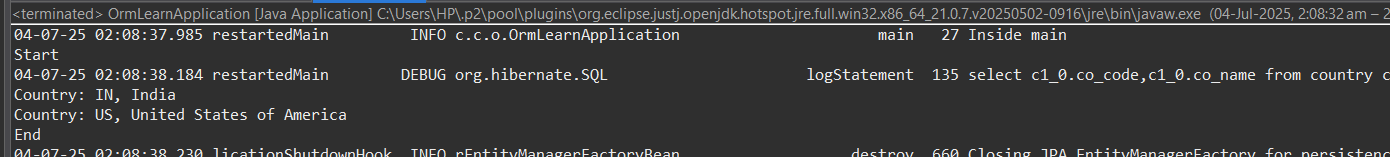
System.***out***.println("Country: " + c.getCode() + ", " + c.getName());

}

System.***out***.println("End");

}

}



**Difference between JPA, Hibernate and Spring Data JPA**

**JPA (Java Persistence API)** is a specification provided by Oracle for object-relational mapping in Java. It defines a set of interfaces and annotations (like @Entity, @Id, etc.) to interact with relational databases. However, JPA itself doesn’t provide any implementation—only the contract.

**Hibernate** is an implementation of the JPA specification and adds many powerful features like HQL (Hibernate Query Language), caching, and lazy loading. It can be used directly or as the JPA provider under the JPA API.

**Spring Data JPA** is a Spring framework that builds on top of JPA and Hibernate to simplify database access. It reduces boilerplate by allowing developers to define interfaces like JpaRepository with methods such as findByName(), and Spring will generate the implementation automatically. It integrates seamlessly with Spring Boot and typically uses Hibernate under the hood.