**Spring Core and Maven**

Ex1>

Code:

**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">  
  
 <!-- Repository Bean -->  
 <bean id="bookRepository" class="org.library.librarymanagement.repository.BookRepository"/>  
  
 <!-- Service Bean with dependency injection -->  
 <bean id="bookService" class="org.library.librarymanagement.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository.java**

package org.library.librarymanagement.repository;  
  
public class BookRepository {  
 public void save(String bookName) {  
 System.*out*.println("Book \"" + bookName + "\" saved to repository.");  
 }  
}

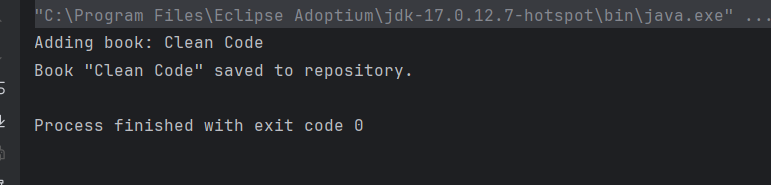
**BookService.java**

package org.library.librarymanagement.service;  
  
import org.library.librarymanagement.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter for Spring Dependency Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.save(bookName);  
 }  
}

**LibraryManagementApplication.java**

package org.library.librarymanagement;  
  
import org.library.librarymanagement.service.BookService;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("Clean Code");  
 }  
  
}

Output

****

**Ex2**

Code:

**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">  
  
 <!-- Repository Bean -->  
 <bean id="bookRepository" class="org.library.librarymanagement.repository.BookRepository"/>  
  
 <!-- Service Bean with dependency injection -->  
 <bean id="bookService" class="org.library.librarymanagement.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**BookRepository.java**

package org.library.librarymanagement.repository;  
  
public class BookRepository {  
 public void save(String bookName) {  
 System.*out*.println("Book \"" + bookName + "\" saved to repository.");  
 }  
}

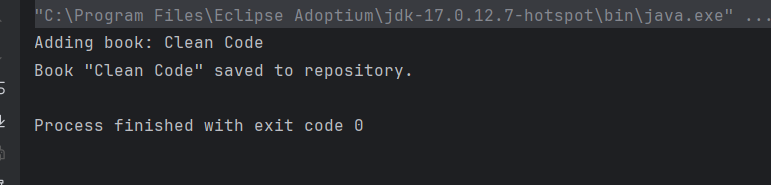
**BookService.java**

package org.library.librarymanagement.service;  
  
import org.library.librarymanagement.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter for Spring Dependency Injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void addBook(String bookName) {  
 System.*out*.println("Adding book: " + bookName);  
 bookRepository.save(bookName);  
 }  
}

**LibraryManagementApplication.java**

package org.library.librarymanagement;  
  
import org.library.librarymanagement.service.BookService;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;  
  
@SpringBootApplication  
public class LibraryManagementApplication {  
  
 public static void main(String[] args) {  
 ApplicationContext context =  
 new ClassPathXmlApplicationContext("applicationContext.xml");  
  
 BookService bookService = context.getBean("bookService", BookService.class);  
 bookService.addBook("Clean Code");  
 }  
  
}

**Output**

****

Ex 4>

Code:

**pom.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 https://maven.apache.org/xsd/maven-4.0.0.xsd">  
 <modelVersion>4.0.0</modelVersion>  
 <parent>  
 <groupId>org.springframework.boot</groupId>  
 <artifactId>spring-boot-starter-parent</artifactId>  
 <version>3.5.3</version>  
 <relativePath/> <!-- lookup parent from repository -->  
 </parent>  
 <groupId>org.library</groupId>  
 <artifactId>LibraryManagement</artifactId>  
 <version>0.0.1-SNAPSHOT</version>  
 <name>LibraryManagement</name>  
 <description>LibraryManagement</description>  
 <url/>  
 <licenses>  
 <license/>  
 </licenses>  
 <developers>  
 <developer/>  
 </developers>  
 <scm>  
 <connection/>  
 <developerConnection/>  
 <tag/>  
 <url/>  
 </scm>  
 <properties>  
 <java.version>17</java.version>  
 </properties>  
 <dependencies>  
 <!-- Spring Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.30</version>  
 </dependency>

<!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.30</version>  
 </dependency>  
  
 <!-- Spring Web MVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.30</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>

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

**Spring Data JPA - Quick Example**

No Question provided in this name

Q2> Difference between JPA, Hibernate and Spring Data JPA

Ans:

| **Aspect** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Definition** | A Java specification for object-relational mapping (ORM) | An ORM framework that implements JPA (and provides extra features) | A Spring project that builds on JPA to simplify data access |
| **Type** | Specification (API contract) | Framework / Implementation | Framework / Abstraction |
| **Provides ORM Engine?** | No | Yes | No (relies on Hibernate or other JPA provider) |
| **Who Maintains It?** | Oracle / Java Community Process | Red Hat | Spring Team |
| **Annotations Defined?** | Yes (e.g., @Entity, @Id, @Table) | Yes (JPA annotations + proprietary Hibernate annotations) | Yes (uses JPA annotations) |
| **APIs Provided** | EntityManager, EntityTransaction, JPQL | Session, Transaction, HQL, plus JPA APIs | Repository interfaces (JpaRepository, CrudRepository) |
| **Query Language** | JPQL (Java Persistence Query Language) | JPQL + HQL (Hibernate Query Language) + native SQL | JPQL + method name queries + @Query annotations |
| **Repository Abstraction** | No | No | Yes (auto-implementation of repositories) |
| **Boilerplate Level** | High (manual EntityManager handling) | High (manual Session handling) | Low (auto repositories, minimal configuration) |
| **Transaction Management** | Manual via EntityTransaction | Manual via Transaction or JTA | Declarative via @Transactional |
| **Pagination and Sorting** | Manual | Manual | Built-in (Pageable, Sort) |
| **Spring Integration** | Requires configuration | Requires configuration | Native integration (auto-configuration, Spring Boot support) |
| **Use Case** | When you need standard, portable ORM | When you need full control or advanced ORM capabilities | When you want rapid development with minimal boilerplate in Spring projects |
| **Extra Features** | None (standard only) | Many (caching, batching, interceptors, custom types) | Inherits Hibernate features + Spring Data enhancements |
| **Learning Curve** | Medium | High (more powerful and complex) | Low to Medium (simplified API, conventions) |