**SPRING CORE – MAVEN**

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

Library Management System

**Filename:** *pom.xml(added dependency)*

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.31</version>  
 </dependency>  
</dependencies>

**Filename:** *BookRepository.java*

package com.library.Repository;  
public class BookRepository {  
 public String getBookData() {  
 return "Clean Code by Robert C. Martin";  
 }  
}

**Filename:** *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 printBook() {  
 System.*out*.println("Book: " + bookRepository.getBookData());  
 }  
}

**Filename:** *applicationContext.xml (create and connects beans)*

<bean id="bookRepository" class="com.library.Repository.BookRepository"/>  
<bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
</bean>

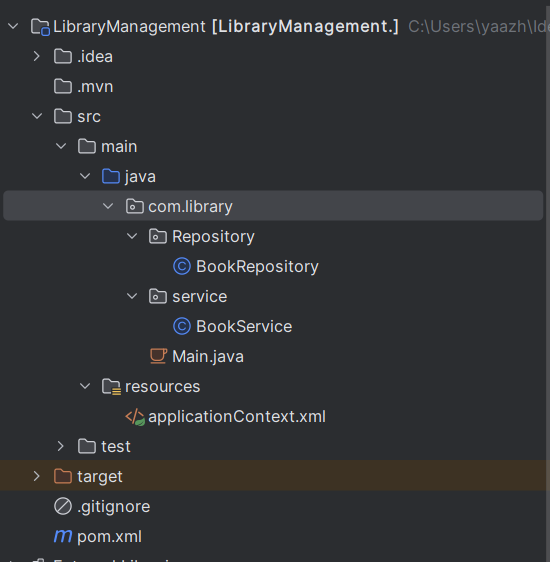
**Filename:** *Main.java*

package com.library;  
import com.library.service.BookService;

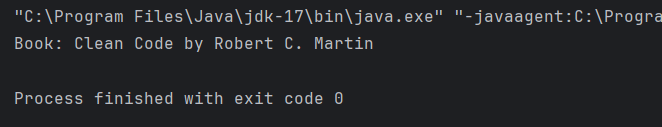
import org.springframework.context.ApplicationContext;  
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class Main {  
 public static void main(String[] args) {   
 ApplicationContext context = new ClassPathXmlApplicationContext("applicationContext.xml");  
 BookService service = (BookService) context.getBean("bookService");  
 service.printBook();  
 }  
}

**File Structure:**

****

**OUTPUT SCREENSHOTS :**

****

**Exercise 2: Implementing Dependency Injection**

* *Modified code of Library Management System*

**Filename:** *pom.xml(added dependency)*

<dependencies>  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.31</version>  
 </dependency>  
</dependencies>

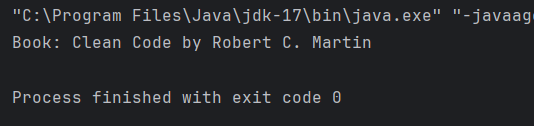
**Filename *:*** *applicationContext.xml (create and connects beans)*

<bean id="bookRepository" class="com.library.Repository.BookRepository"/>  
<bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
</bean>

**Filename:** *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 printBook() {  
 System.*out*.println("Book: " + bookRepository.getBookData());  
 }  
}

**OUTPUT SCREENSHOTS :**

****

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

* Modified the pom.xml file

**Filename:** *pom.xml*

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-aop</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>${spring.version}</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

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

<version>4.0.1</version>

<scope>provided</scope>

</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>

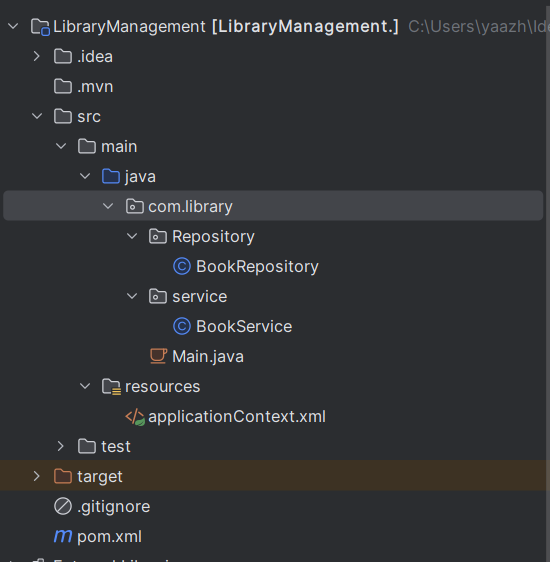
</configuration>

</plugin>

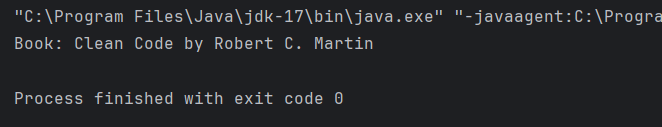
</plugins>

</build>

**File Structure:**

****

**OUTPUT SCREENSHOTS :**

****

**Exercise - Spring Data JPA - Quick Example**

**File:** *OrmLearnApplication.java*  
package com.cognizant.ormlearn;  
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.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);  
 LOGGER.info("Inside main");  
 testGetAllCountries();  
 }  
  
 private static void testGetAllCountries() {  
 LOGGER.info("Start");  
 List<Country> countries = countryService.getAllCountries();  
 LOGGER.debug("countries={}", countries);  
 LOGGER.info("End");  
 }  
}

**File:** *Country.java*

package com.cognizant.ormlearn.model;  
  
import javax.persistence.Column;  
import javax.persistence.Entity;  
import javax.persistence.Id;  
import javax.persistence.Table;  
  
@Entity  
@Table(name="country")  
public class Country {  
  
 @Id  
 @Column(name="code")  
 private String code;  
  
 @Column(name="name")  
 private String name;  
  
 // Getters and Setters  
  
 @Override  
 public String toString() {  
 return "Country [code=" + code + ", name=" + name + "]";  
 }  
}

**File:** *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> {  
  
}

**File:** *CountryService.java*

package com.cognizant.ormlearn.service;  
import java.util.List;  
import javax.transaction.Transactional;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
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();  
 }  
}

**File:** *application.properties*

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

**SPRIN DATA JPA**

* Spring Boot with Spring Data JPA simplifies interaction with the database using repository interfaces.
* @SpringBootApplication is a meta-annotation that enables component scanning, auto-configuration, and Spring Boot application configuration.
* Entity classes like Country represent database tables and use annotations such as @Entity, @Table, and @Id for mapping.
* Repositories like CountryRepository provide built-in methods to perform CRUD operations without writing custom SQL or HQL.
* Services like CountryService encapsulate business logic and make use of @Transactional to ensure atomicity of operations.

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

**HIBERNATE**

public Integer addEmployee(Employee employee) {

Session session = factory.openSession();

Transaction tx = null;

Integer employeeID = null;

try {

tx = session.beginTransaction();

employeeID = (Integer) session.save(employee);

tx.commit();

} catch (HibernateException e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

return employeeID;

}

**SPRING DATA JPA**

**Filename:** *EmployeeService.java*

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee); // Spring handles everything

}

**Filename :** *EmployeeRepository.java*

Public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

**Hibernate**

1. Requires manual handling of sessions and transactions.
2. Uses low-level APIs like Session and Transaction.
3. Involves more boilerplate and repetitive code.
4. Provides fine-grained control over database operations.
5. Requires explicit configuration and error handling.

**Spring Data JPA**

1. Abstracts JPA and simplifies database interactions.
2. Automatically manages sessions and transactions.
3. Reduces boilerplate through built-in repository methods.
4. Encourages cleaner and more maintainable code.
5. Easily integrates with Spring Boot and other Spring modules.