Spring Core\_Maven

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

**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.arijit.test</groupId>  
 <artifactId>Library\_Management</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>Library\_Management</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>7.0.0-M6</version>  
 </dependency>  
  
 </dependencies>  
</project>

**src/main/resources:**

**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="com.library.repository.BookRepository" />  
  
 <!-- Service Bean with Dependency Injection -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
</beans>

**com.library.service:**

**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 displayBook(int id) {  
 String book = bookRepository.getBookById(id);  
 System.*out*.println("Book fetched: " + book);  
 }  
}

**com.library.repository:**

**BookRepository.java**

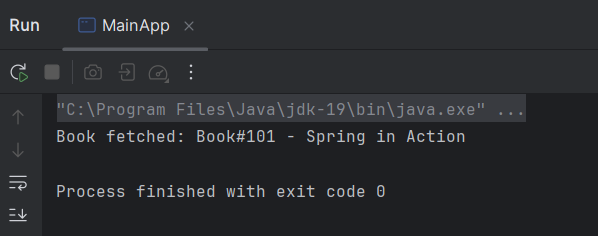
package com.library.repository;  
  
public class BookRepository {  
 public String getBookById(int id) {  
 return "Book#" + id + " - Spring in Action";  
 }  
}

**com.library:**

**MainApp.java:**

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 service = context.getBean("bookService", BookService.class);  
 service.displayBook(101);  
 }  
}

**OUTPUT:**



**Exercise 2**: **Implementing Dependency Injection:**

**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.arijit.test</groupId>  
 <artifactId>Library\_Management</artifactId>  
 <version>1.0-SNAPSHOT</version>  
 <packaging>jar</packaging>  
  
 <name>Library\_Management</name>  
 <url>http://maven.apache.org</url>  
  
 <properties>  
 <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>  
 </properties>  
  
 <dependencies>  
 <dependency>  
 <groupId>junit</groupId>  
 <artifactId>junit</artifactId>  
 <version>4.13.2</version>  
 <scope>test</scope>  
 </dependency>  
  
 <!-- https://mvnrepository.com/artifact/org.springframework/spring-webmvc -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>7.0.0-M6</version>  
 </dependency>  
  
 </dependencies>  
</project>

**src/main/resources:**

**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 via setter -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

**com.library.service:**

**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter-based dependency injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook(int id) {  
 String book = bookRepository.getBookById(id);  
 System.*out*.println("Book fetched: " + book);  
 }  
}

**com.library.repository:**

**BookRepository.java**

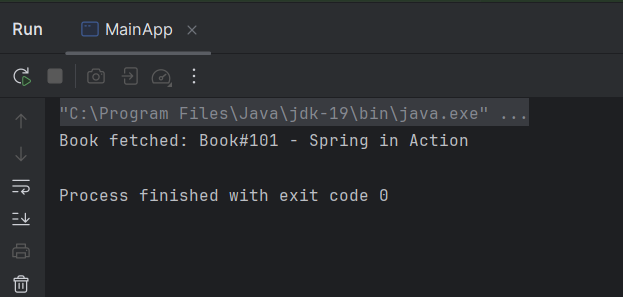
package com.library.repository;  
  
public class BookRepository {  
 public String getBookById(int id) {  
 return "Book#" + id + " - Spring in Action";  
 }  
}

**com.library:**

**MainApp.java:**

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 service = context.getBean("bookService", BookService.class);  
 service.displayBook(101); // Output should be: Book fetched: Book#101 - Spring in Action  
 }  
}

**OUTPUT:**



**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>  
  
 <dependencies>  
 <!-- Spring Core + Beans + Context -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-context</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
  
 <!-- Spring AOP -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-aop</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
  
 <!-- Spring Web MVC -->  
 <dependency>  
 <groupId>org.springframework</groupId>  
 <artifactId>spring-webmvc</artifactId>  
 <version>5.3.33</version>  
 </dependency>  
 </dependencies>  
  
 <build>  
 <plugins>  
 <!-- Java Compiler Plugin -->  
 <plugin>  
 <groupId>org.apache.maven.plugins</groupId>  
 <artifactId>maven-compiler-plugin</artifactId>  
 <version>3.11.0</version>  
 <configuration>  
 <source>1.8</source>  
 <target>1.8</target>  
 </configuration>  
 </plugin>  
 </plugins>  
 </build>  
</project>

**src/main/resources:**

**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 via setter -->  
 <bean id="bookService" class="com.library.service.BookService">  
 <property name="bookRepository" ref="bookRepository"/>  
 </bean>  
  
</beans>

**com.library.service:**

**BookService.java**

package com.library.service;  
  
import com.library.repository.BookRepository;  
  
public class BookService {  
 private BookRepository bookRepository;  
  
 // Setter-based dependency injection  
 public void setBookRepository(BookRepository bookRepository) {  
 this.bookRepository = bookRepository;  
 }  
  
 public void displayBook(int id) {  
 String book = bookRepository.getBookById(id);  
 System.*out*.println("Book fetched: " + book);  
 }  
}

**com.library.repository:**

**BookRepository.java**

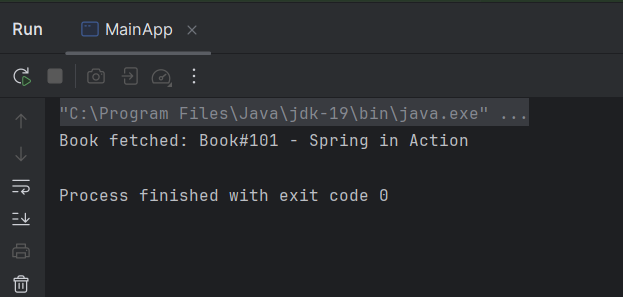
package com.library.repository;  
  
public class BookRepository {  
 public String getBookById(int id) {  
 return "Book#" + id + " - Spring in Action";  
 }  
}

**com.library:**

**MainApp.java:**

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 service = context.getBean("bookService", BookService.class);  
 service.displayBook(101); // Output should be: Book fetched: Book#101 - Spring in Action  
 }  
}

**OUTPUT:**

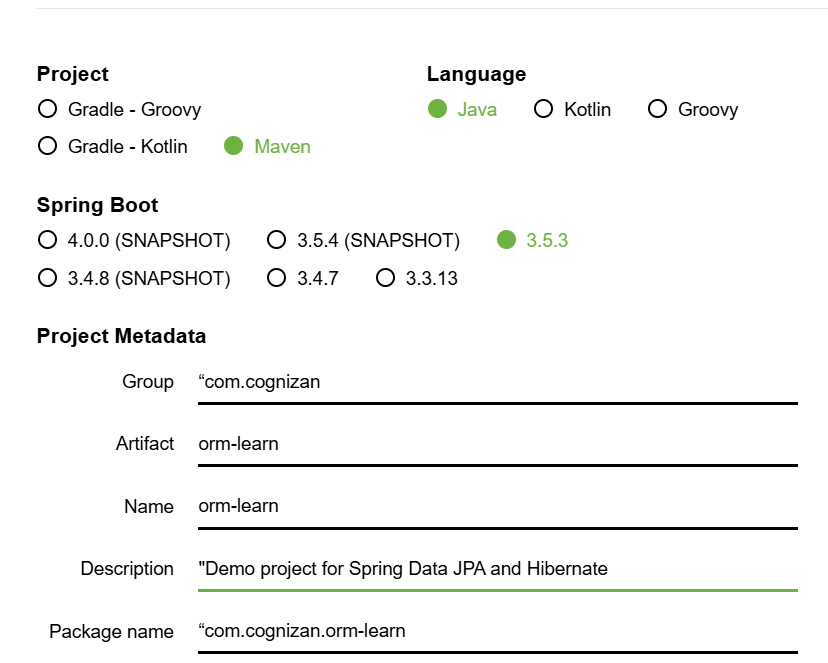


**Spring Data JPA with Spring Boot, Hibernate**

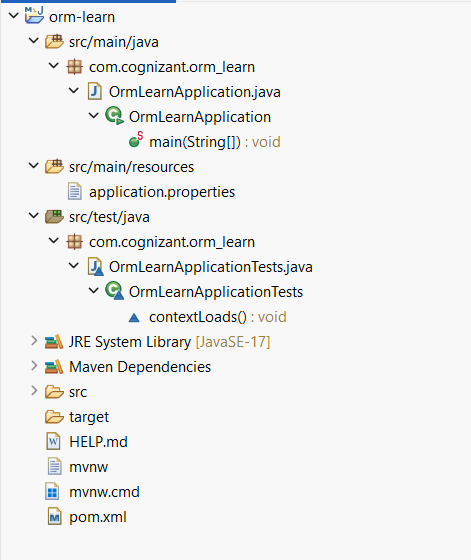
**Spring Data, jpa and handson**

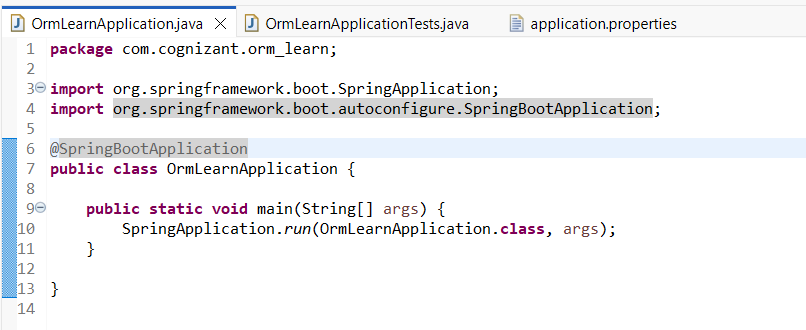
**Handson 1**: **Spring Data Jpa-Quick Example:**

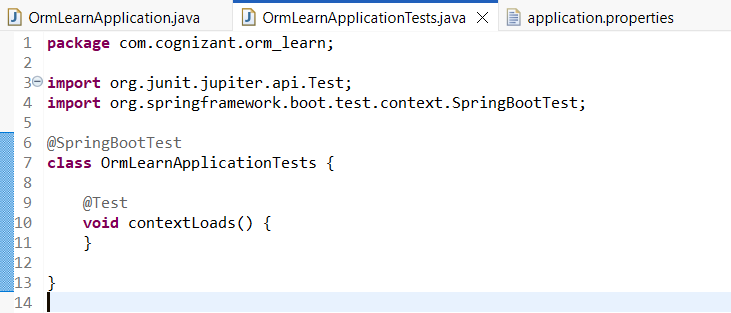
**https://start.spring.io**

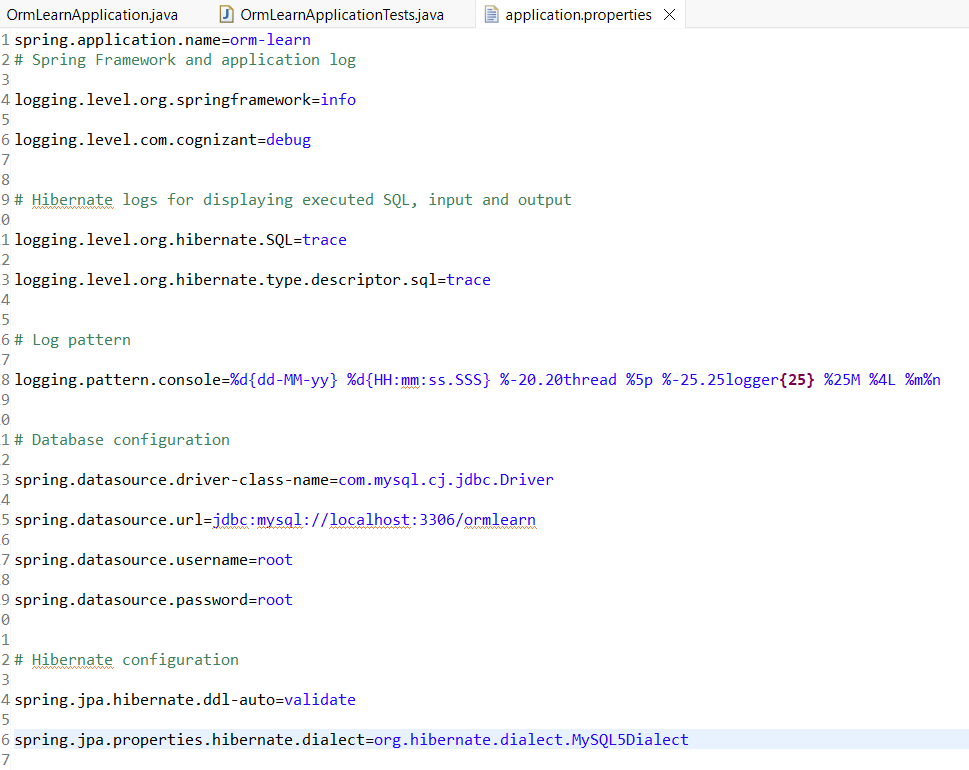


Eclipse-IDE

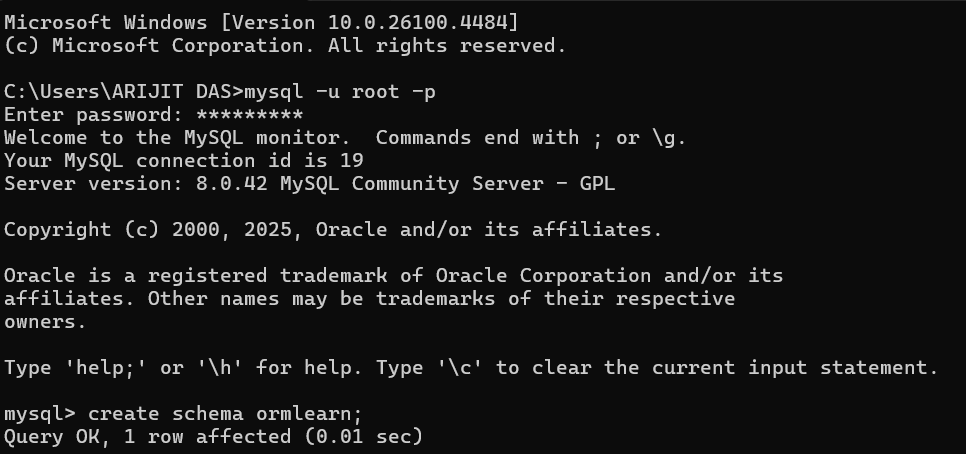








**MySQL server 8.0:**



**Handson -4 : Difference between JPA, Hibernate and Spring Data JPA**

**JPA vs Hibernate vs Spring Data JPA**

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification / API | Implementation of JPA + extras | Framework built on JPA + Hibernate |
| **Part of** | Java EE / Jakarta EE | Independent ORM tool (by Red Hat) | Spring Framework |
| **Implements JPA?** | N/A | ✅ Yes | ✅ Uses JPA (via Hibernate) |
| **Reduces Boilerplate?** | ❌ No | ❌ Partially | ✅ Yes (Repositories auto-generated) |
| **Default in Spring Boot** | ❌ No | ✅ Hibernate | ✅ Spring Data JPA |
| **Used Directly In Code?** | ✅ Yes (via annotations) | ✅ Yes (via Session API or JPA) | ✅ Yes (via JpaRepository) |
| **Querying** | JPQL / Criteria API | JPQL + HQL + Native SQL | Derived queries + JPQL + Native |
| **Additional Features** | Only standard ORM mapping | Caching, Batching, HQL, etc. | Paging, Sorting, Custom Repos, etc. |
| **Learning Curve** | Moderate | Moderate to High | Easy (most logic abstracted) |
| **Example Usage** | @Entity, @Id, @OneToMany | Session.save(), HQL queries | findByName(), JpaRepository<T, ID> |