WEEK-3

**1.spring-data-jpa-handson**

**Spring Data JPA - Quick Example**

LibraryManagement/

├── pom.xml

├── src/

│ ├── main/

│ │ ├── java/

│ │ │ └── com/library/

│ │ │ ├── model/Book.java

│ │ │ ├── repository/BookRepository.java

│ │ │ └── App.java

│ │ └── resources/

│ │ ├── applicationContext.xml

│ │ └── META-INF/persistence.xml

**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>

  <name>LibraryManagement</name>

  <properties>

    <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

    <maven.compiler.source>1.8</maven.compiler.source>

    <maven.compiler.target>1.8</maven.compiler.target>

  </properties>

  <dependencies>

    <!-- Spring Context -->

    <dependency>

      <groupId>org.springframework</groupId>

      <artifactId>spring-context</artifactId>

      <version>5.3.34</version>

    </dependency>

    <!-- Spring Data JPA -->

    <dependency>

      <groupId>org.springframework.data</groupId>

      <artifactId>spring-data-jpa</artifactId>

      <version>2.7.10</version>

    </dependency>

    <!-- Hibernate -->

    <dependency>

      <groupId>org.hibernate</groupId>

      <artifactId>hibernate-core</artifactId>

      <version>5.6.15.Final</version>

    </dependency>

    <!-- H2 Database -->

    <dependency>

      <groupId>com.h2database</groupId>

      <artifactId>h2</artifactId>

      <version>2.2.224</version>

      <scope>runtime</scope>

    </dependency>

    <!-- JPA API -->

    <dependency>

      <groupId>jakarta.persistence</groupId>

      <artifactId>jakarta.persistence-api</artifactId>

      <version>2.2.3</version>

    </dependency>

    <!-- JUnit -->

    <dependency>

      <groupId>junit</groupId>

      <artifactId>junit</artifactId>

      <version>4.13.2</version>

      <scope>test</scope>

    </dependency>

  </dependencies>

  <build>

    <plugins>

      <!-- Maven Compiler -->

      <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 -->

      <plugin>

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

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

        <version>3.1.0</version>

      </plugin>

    </plugins>

  </build>

</project>

**Book.java:**

package com.library.model;

import javax.persistence.\*;

@Entity

public class Book {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private Long id;

    private String title;

    // Constructors

    public Book() {}

    public Book(String title) {

        this.title = title;

    }

    // Getters and setters

    public Long getId() { return id; }

    public String getTitle() { return title; }

    public void setTitle(String title) { this.title = title; }

}

**BookRepository.java:**

package com.library.repository;

import com.library.model.Book;

import org.springframework.data.jpa.repository.JpaRepository;

public interface BookRepository extends JpaRepository<Book, Long> {

}

**App.java:**

package com.library;

import com.library.model.Book;

import com.library.repository.BookRepository;

import org.springframework.context.ApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class App {

    public static void main(String[] args) {

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

        BookRepository bookRepo = context.getBean(BookRepository.class);

        bookRepo.save(new Book("Effective Java"));

        bookRepo.save(new Book("Spring in Action"));

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

        bookRepo.findAll().forEach(book -> System.out.println(book.getId() + ": " + book.getTitle()));

    }

}

**applicationContext.xml:**

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

       xmlns:context="http://www.springframework.org/schema/context"

       xmlns:tx="http://www.springframework.org/schema/tx"

       xmlns:jpa="http://www.springframework.org/schema/data/jpa"

       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

          http://www.springframework.org/schema/context

          http://www.springframework.org/schema/context/spring-context.xsd

          http://www.springframework.org/schema/tx

          http://www.springframework.org/schema/tx/spring-tx.xsd

          http://www.springframework.org/schema/data/jpa

          http://www.springframework.org/schema/data/jpa/spring-jpa.xsd">

    <!-- Enable component scanning -->

    <context:component-scan base-package="com.library"/>

    <!-- JPA entity manager factory -->

    <bean id="entityManagerFactory" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">

        <property name="persistenceUnitName" value="libraryPU"/>

    </bean>

    <!-- Transaction manager -->

    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">

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

    </bean>

    <tx:annotation-driven transaction-manager="transactionManager"/>

    <!-- Enable Spring Data JPA repositories -->

    <jpa:repositories base-package="com.library.repository"/>

</beans>

**persistence.xml:**<?xml version="1.0" encoding="UTF-8"?>

<persistence version="2.2"

             xmlns="http://xmlns.jcp.org/xml/ns/persistence"

             xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence

                                 http://xmlns.jcp.org/xml/ns/persistence/persistence\_2\_2.xsd">

    <persistence-unit name="libraryPU" transaction-type="RESOURCE\_LOCAL">

        <provider>org.hibernate.jpa.HibernatePersistenceProvider</provider>

        <class>com.library.model.Book</class>

        <properties>

            <property name="javax.persistence.jdbc.driver" value="org.h2.Driver"/>

            <property name="javax.persistence.jdbc.url" value="jdbc:h2:mem:testdb"/>

            <property name="javax.persistence.jdbc.user" value="sa"/>

            <property name="javax.persistence.jdbc.password" value=""/>

            <property name="hibernate.dialect" value="org.hibernate.dialect.H2Dialect"/>

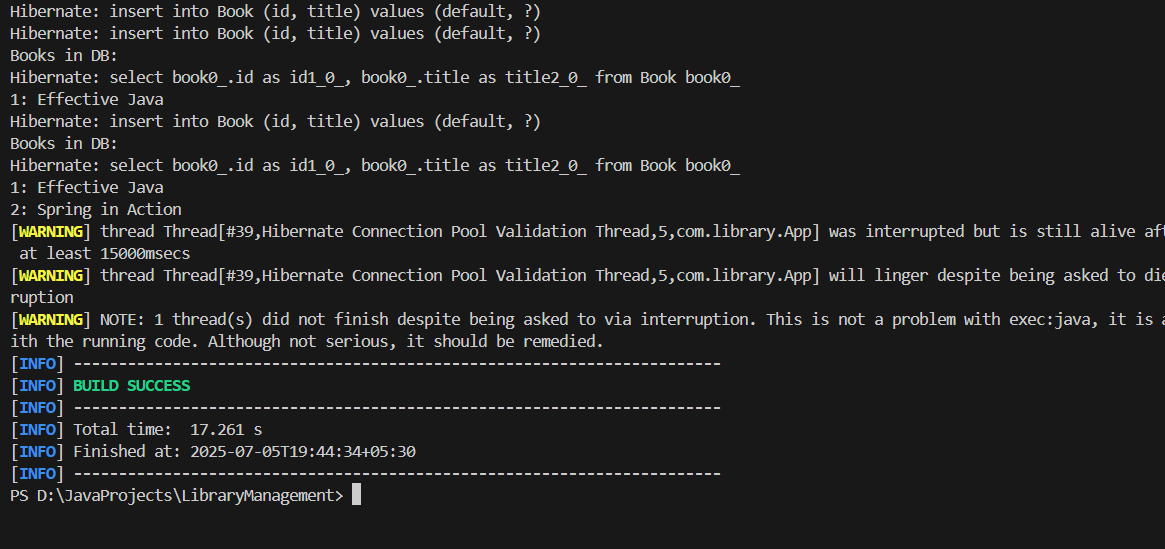
            <property name="hibernate.hbm2ddl.auto" value="update"/>

            <property name="hibernate.show\_sql" value="true"/>

        </properties>

    </persistence-unit>

</persistence>

**Output:  
**

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

| **Feature** | **JPA** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- | --- |
| **Type** | Specification (Interface) | Implementation (Library) | Abstraction over JPA/Hibernate |
| **Maintained By** | Oracle (Jakarta EE) | Red Hat | Spring Team |
| **What it does** | Defines standard for ORM | Implements JPA & extra features | Simplifies JPA with auto-repositories |
| **Needs configuration?** | Yes | Yes | Minimal – uses Spring auto-configuration |
| **Boilerplate Code** | More | Moderate | Very little (auto-generated methods) |
| **Query Options** | JPQL | HQL + Criteria | JPQL + method names + @Query + Specifications |

While JPA acts as a **standard specification** that defines how Java objects interact with relational databases, it does not provide an actual implementation. Hibernate, on the other hand, is the most widely used **implementation of JPA**, offering additional features such as lazy loading, caching, and a powerful query language called HQL (Hibernate Query Language). Developers often use Hibernate to interact with the database while still conforming to the JPA standards.

Spring Data JPA builds on top of JPA and Hibernate, offering a much more **developer-friendly and productive approach** to data access. It eliminates boilerplate code by providing repository interfaces that are auto-implemented at runtime. Spring Data JPA supports query derivation from method names, pagination, sorting, and custom queries using the @Query annotation. It is ideal for rapid development and works seamlessly with the rest of the Spring ecosystem, making it the preferred choice for modern Spring-based applications.