**Week-3 : Spring Data JPA with Spring Boot, Hibernate – HandsOn**

**Spring Data JPA - Quick Example**

Code:

**pom.xml**

**Location:** Project root (orm-learn/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.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>1.0.0</version>,

<properties>

<java.version>1.8</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter for JPA -->

<dependency>

<groupId>org.springframework.boot</groupId>

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

</dependency>

<!-- MySQL JDBC Driver -->

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter</artifactId>

</dependency>

<!-- Spring Boot DevTools (optional) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

<!-- JUnit (optional for testing) -->

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-test</artifactId>

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot Maven Plugin -->

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

</plugin>

</plugins>

</build>

</project>

**application.properties**

spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn

spring.datasource.username=root

spring.datasource.password=root

spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

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

(Replace username and password with your MySQL credentials)

**Country.java (Entity)**

**Location:** src/main/java/com/cognizant/ormlearn/model/Country.java

package com.cognizant.ormlearn.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

private String code;

private String name;

public Country() {}

public Country(String code, String name) {

this.code = code;

this.name = name;

}

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 + "]";

}

}

**4. CountryRepository.java (Repository)**

**Location:** src/main/java/com/cognizant/ormlearn/repository/CountryRepository.java

package com.cognizant.ormlearn.repository;

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

import com.cognizant.ormlearn.model.Country;

public interface CountryRepository extends JpaRepository<Country, String> {

}

**5. OrmLearnApplication.java (Main Class)**

**Location:** src/main/java/com/cognizant/ormlearn/OrmLearnApplication.java

package com.cognizant.ormlearn;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.repository.CountryRepository;

@SpringBootApplication

public class OrmLearnApplication implements CommandLineRunner {

@Autowired

private CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.run(OrmLearnApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

Country country = new Country("IN", "India");

countryRepository.save(country);

Country result = countryRepository.findById("IN").get();

System.out.println(result);

}

}

**6. MySQL Table SQL**

CREATE TABLE country (

code VARCHAR(2) PRIMARY KEY,

name VARCHAR(100)

);

Output:

A screenshot of a computer program

AI-generated content may be incorrect.

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

Code:

**JPA (Java Persistence API)**

* It is a **specification**, not an implementation.
* Provides a set of **interfaces and annotations** for object-relational mapping (ORM).
* Requires a provider (like Hibernate) to work.
* Common JPA annotations: @Entity, @Table, @Id, @GeneratedValue, etc.

**Example with JPA (using EntityManager):**

EntityManagerFactory emf = Persistence.createEntityManagerFactory("persistenceUnit");

EntityManager em = emf.createEntityManager();

Country country = new Country("IN", "India");

em.getTransaction().begin();

em.persist(country);

em.getTransaction().commit();

Country saved = em.find(Country.class, "IN");

System.out.println(saved);

**Hibernate**

* Hibernate is a **popular implementation** of JPA.
* It also provides **extra features** beyond JPA, like:
  + Caching
  + Lazy loading
  + Native SQL and HQL (Hibernate Query Language)
* Can be used standalone or as a JPA provider.

**Example (Hibernate config in XML):**

<hibernate-configuration>

<session-factory>

<property name="hibernate.dialect">org.hibernate.dialect.MySQLDialect</property>

<property name="hibernate.connection.driver\_class">com.mysql.cj.jdbc.Driver</property>

<property name="hibernate.connection.url">jdbc:mysql://localhost:3306/ormlearn</property>

<property name="hibernate.connection.username">root</property>

<property name="hibernate.connection.password">root</property>

</session-factory>

</hibernate-configuration>

**Hibernate Session Example:**

java

CopyEdit

Session session = sessionFactory.openSession();

Transaction tx = session.beginTransaction();

Country country = new Country("IN", "India");

session.save(country);

tx.commit();

session.close();

**Spring Data JPA**

* It is **built on top of JPA and Hibernate**.
* Simplifies data access layer with:

Automatic CRUD methods

Query generation from method names

Paging and sorting support

* Reduces boilerplate code

**Spring Data JPA Example (Repository interface):**

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

}

**Usage in Service or CommandLineRunner:**

@Autowired

private CountryRepository countryRepository;

public void run(String... args) throws Exception {

Country country = new Country("IN", "India");

countryRepository.save(country);

Country found = countryRepository.findById("IN").orElse(null);

System.out.println(found);

}

Output:

**Implement services for managing Country**

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

Adding a new Country

A screenshot of a computer

AI-generated content may be incorrect.