**WEEK-3 Spring Data JPA**

**Hands on 1**

**Spring Data JPA - Quick Example**

**orm\_learn.sql:**

CREATE DATABASE orm\_learn;

USE orm\_learn;

CREATE TABLE country (

code VARCHAR(5) PRIMARY KEY,

name VARCHAR(100)

);

INSERT INTO country (code, name) VALUES ('IN', 'India');

INSERT INTO country (code, name) VALUES ('US', 'United States');

SELECT \* FROM country;

**pom.xml:**

<project xmlns="http://maven.apache.org/POM/4.0.0" ...>

<modelVersion>4.0.0</modelVersion>

<groupId>com.cognizant</groupId>

<artifactId>orm-learn</artifactId>

<version>0.0.1-SNAPSHOT</version>

<name>orm-learn</name>

<description>Spring Boot JPA Project</description>

<properties>

<java.version>21</java.version>

</properties>

<dependencies>

<!-- Spring Boot Starter -->

<dependency>

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

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

</dependency>

<!-- MySQL Driver -->

<dependency>

<groupId>com.mysql</groupId>

<artifactId>mysql-connector-j</artifactId>

</dependency>

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

<dependency>

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

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

<scope>runtime</scope>

</dependency>

<!-- Spring Boot Starter Test -->

<dependency>

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

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

<scope>test</scope>

</dependency>

</dependencies>

<build>

<plugins>

<!-- Spring Boot 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=your\_password

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

**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;

}

}

**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> {

}

**OrmLearnApplication.java:**

package com.cognizant.ormlearn;

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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);

System.out.println("Country Saved: " + country.getName());

}

}

**Output:**

Inside main

Start

countries=[Country [code=IN, name=India], Country [code=US, name=United States of America]]

End

**Hands on 4**

**Implement services for managing Country**

1. **Java Persistence API (JPA)**

* JPA is a Java specification (JSR 338) for object-relational mapping (ORM).
* It provides rules and annotations to map Java objects to database tables.
* JPA does not provide actual implementation; it requires a provider like Hibernate.
* Common annotations: @Entity, @Id, @Table, @Column.

1. **Hibernate**

* Hibernate is a popular implementation of the JPA specification.
* It is a full-featured ORM tool used to interact with databases using Java objects.
* Offers features like automatic SQL generation, transaction management, and caching.
* More boilerplate code is required compared to Spring Data JPA.
* Used with or without Spring Framework.

**Example using Hibernate:**

Session session = factory.openSession();

Transaction tx = session.beginTransaction();

session.save(employee);

tx.commit();

1. **Spring Data JPA**

* Spring Data JPA is not a JPA implementation but an abstraction built on top of JPA and Hibernate.
* It simplifies database operations and eliminates boilerplate code using repository interfaces.
* Automatically provides CRUD methods like save(), findAll(), deleteById() etc.
* Integrates smoothly with Spring Boot and handles transactions, sessions, and queries internally.
* **Example using Spring Data JPA:**

public interface EmployeeRepository extends JpaRepository<Employee, Integer> { }

@Autowired

private EmployeeRepository employeeRepository;

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}