**WEEK-3**

**Difference between Hibernate and Spring Data JPA**

**Hibernate** is an Object-Relational Mapping (ORM) framework and a **JPA implementation**.

**Spring Data JPA** is a **high-level abstraction** built on top of JPA to simplify data access logic in Spring applications.

**Spring Data JPA uses Hibernate (or any JPA provider)** under the hood.

### 🧩 Core Differences

| **Feature** | **Hibernate** | **Spring Data JPA** |
| --- | --- | --- |
| **Type** | ORM Framework and JPA Implementation | JPA Abstraction Framework |
| **Purpose** | Implements JPA and provides ORM features | Simplifies data access layers using repository abstraction |
| **Boilerplate Code** | Requires manual implementation of DAOs and queries | Reduces boilerplate using repository interfaces |
| **Query Language** | Uses HQL, JPQL, or native SQL | Supports method-name-based queries and @Query annotations |
| **Transaction Management** | Manual or Spring-integrated | Fully managed via Spring’s @Transactional |
| **Integration with Spring** | Requires manual setup | Fully integrated with Spring Boot/Spring Framework |
| **Learning Curve** | Steeper (low-level control) | Easier (high-level abstraction) |
| **Custom Behavior** | Highly customizable | Customizable but with some limitations due to abstraction |
| **Underlying JPA Provider** | Hibernate itself | Uses Hibernate (or others like EclipseLink) under the hood |
| **Code Example for DAO** | Requires custom DAO classes | Just create interface and extend JpaRepository or CrudRepository |
| **Usage Scope** | Good for complex ORM mappings and custom optimization | Great for rapid development and clean architecture |

**Finding a Country using Country Code:**

**FileName: Country.java**

package com.library.model;

import javax.persistence.\*;

@Entity

@Table(name = "country")

public class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@Column(unique = true, nullable = false)

private String code; // like "IN", "US", "JP"

// Getters and Setters

public Long getId() { return id; }

public void setId(Long id) { this.id = id; }

public String getName() { return name; }

public void setName(String name) { this.name = name; }

public String getCode() { return code; }

public void setCode(String code) { this.code = code; }

}

FileName:CountryRepository.java

package com.library.repository;

import com.library.model.Country;

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

public interface CountryRepository extends JpaRepository<Country, Long> {

Country findByCode(String code);

}

FileName:CountryService.java

package com.library.service;

import com.library.model.Country;

import com.library.repository.CountryRepository;

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

import org.springframework.stereotype.Service;

@Service

public class CountryService {

@Autowired

private CountryRepository countryRepository;

public Country getCountryByCode(String code) {

return countryRepository.findByCode(code);

}

}

FileName: MainApp.java

package com.library;

import com.library.model.Country;

import com.library.service.CountryService;

import org.springframework.context.ApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainApp {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(AppConfig.class); // if you're using Java config

CountryService service = context.getBean(CountryService.class);

Country country = service.getCountryByCode("IN");

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

}

}