**Spring Data JPA vs Hibernate**

1. Overview

This section outlines the differences between Spring Data JPA and Hibernate through practical coding examples. While both are employed for managing data persistence, they cater to distinct needs within Java development.

2. Spring Data JPA Example

Configuration:

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

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Entity Class:

package com.example.jpademo.entity;

import jakarta.persistence.\*;

@Entity

public class StaffMember {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer employeeId;

private String fullName;

private String division;

}

Repository Interface:

package com.example.jpademo.repository;

import com.example.jpademo.entity.StaffMember;

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

public interface StaffRepository extends JpaRepository<StaffMember, Integer> {}

Service Layer:

package com.example.jpademo.service;

import com.example.jpademo.entity.StaffMember;

import com.example.jpademo.repository.StaffRepository;

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

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

@Service

public class StaffService {

@Autowired

private StaffRepository staffRepository;

@Transactional

public StaffMember registerStaff(StaffMember staff) {

return staffRepository.save(staff);

}

}

Controller:

package com.example.jpademo.controller;

import com.example.jpademo.entity.StaffMember;

import com.example.jpademo.service.StaffService;

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

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/api/staff")

public class StaffController {

@Autowired

private StaffService staffService;

@PostMapping

public StaffMember addStaffMember(@RequestBody StaffMember staff) {

return staffService.registerStaff(staff);

}

}

3. Hibernate Example

pom.xml Dependencies (summary):

<dependencies>

<!-- Hibernate Core -->

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.6.15.Final</version>

</dependency>

<!-- MySQL Connector -->

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.33</version>

</dependency>

<!-- JPA API -->

<dependency>

<groupId>jakarta.persistence</groupId>

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

<version>2.2.3</version>

</dependency>

<!-- Logging -->

<dependency>

<groupId>org.jboss.logging</groupId>

<artifactId>jboss-logging</artifactId>

<version>3.4.3.Final</version>

</dependency>

</dependencies>

Entity Class:

package com.example.entity;

import jakarta.persistence.\*;

@Entity

@Table(name = "staff")

public class StaffMember {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer staffId;

private String fullName;

private String division;

public StaffMember() {}

public StaffMember(String fullName, String division) {

this.fullName = fullName;

this.division = division;

}

public Integer getStaffId() { return staffId; }

public void setStaffId(Integer staffId) { this.staffId = staffId; }

public String getFullName() { return fullName; }

public void setFullName(String fullName) { this.fullName = fullName; }

public String getDivision() { return division; }

public void setDivision(String division) { this.division = division; }

}

hibernate.cfg.xml:

<hibernate-configuration>

<session-factory>

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

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

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

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

<property name="hibernate.dialect">org.hibernate.dialect.MySQL8Dialect</property>

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

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

<mapping class="com.example.entity.StaffMember"/>

</session-factory>

</hibernate-configuration>

HibernateUtil.java:

package com.example.util;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class HibernateUtil {

private static final SessionFactory sessionFactory;

static {

try {

sessionFactory = new Configuration()

.configure("hibernate.cfg.xml")

.buildSessionFactory();

} catch (Throwable ex) {

throw new ExceptionInInitializerError("Failed to initialize SessionFactory " + ex);

}

}

public static SessionFactory getSessionFactory() {

return sessionFactory;

}

}

App.java:

package com.example;

import com.example.entity.StaffMember;

import com.example.util.HibernateUtil;

import org.hibernate.Session;

import org.hibernate.Transaction;

public class App {

public static void main(String[] args) {

StaffMember staff = new StaffMember("Arjun", "IT");

Session session = HibernateUtil.getSessionFactory().openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

session.save(staff);

tx.commit();

System.out.println("Staff saved with ID: " + staff.getStaffId());

} catch (Exception e) {

if (tx != null) tx.rollback();

e.printStackTrace();

} finally {

session.close();

}

}

}

4. Comparison

- Hibernate acts as an Object Relational Mapping (ORM) tool, implementing the JPA specification.

- Spring Data JPA is a higher-level framework layered over JPA and Hibernate, designed to minimize repetitive code.

- Spring Data JPA eliminates the need for manual SQL or HQL for standard operations by leveraging repository interfaces.

- Hibernate offers greater customization and control but demands more effort in managing sessions and transactions.

- By default, Spring Data JPA relies on Hibernate, meaning JPA usage does not exclude Hibernate.