Difference between JPA, Hibernate and Spring Data JPA

PART 1: Using Spring Data JPA (STS + MySQL)

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.example</groupId>

<artifactId>employee-management</artifactId>

<version>1.0.0</version>

<packaging>jar</packaging>

<properties>

<java.version>17</java.version>

</properties>

<dependencies>

<dependency>

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

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

</dependency>

<dependency>

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

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

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<scope>runtime</scope>

</dependency>

</dependencies>

<build>

<plugins>

<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/employee\_db

spring.datasource.username=root

spring.datasource.password=yourpassword

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

Employee.java (Model)

package com.example.employeemanagement.model;

import jakarta.persistence.\*;

@Entity

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String role;

public Employee() {}

public Employee(String name, String role) {

this.name = name;

this.role = role;

}

// Getters & Setters

public Integer getId() { return id; }

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

public String getName() { return name; }

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

public String getRole() { return role; }

public void setRole(String role) { this.role = role; }

}

EmployeeRepository.java (Repository)

package com.example.employeemanagement.repository;

import com.example.employeemanagement.model.Employee;

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

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

EmployeeService.java (Service)

package com.example.employeemanagement.service;

import com.example.employeemanagement.model.Employee;

import com.example.employeemanagement.repository.EmployeeRepository;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.List;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

public List<Employee> getAllEmployees() {

return employeeRepository.findAll();

}

public void deleteEmployee(Integer id) {

employeeRepository.deleteById(id);

}

}

EmployeeController.java (Controller)

package com.example.employeemanagement.controller;

import com.example.employeemanagement.model.Employee;

import com.example.employeemanagement.service.EmployeeService;

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

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

import java.util.List;

@RestController

@RequestMapping("/api/employees")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping("/add")

public String addEmployee(@RequestBody Employee employee) {

employeeService.addEmployee(employee);

return "Employee added successfully!";

}

@GetMapping("/all")

public List<Employee> getAllEmployees() {

return employeeService.getAllEmployees();

}

@DeleteMapping("/delete/{id}")

public String deleteEmployee(@PathVariable Integer id) {

employeeService.deleteEmployee(id);

return "Employee deleted successfully!";

}

}

EmployeeManagementApplication.java

package com.example.employeemanagement;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class EmployeeManagementApplication {

public static void main(String[] args) {

SpringApplication.run(EmployeeManagementApplication.class, args);

}

}

PART 2: Using Pure Hibernate (Without Spring Boot)

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.example</groupId>

<artifactId>hibernate-demo</artifactId>

<version>1.0-SNAPSHOT</version>

<properties>

<maven.compiler.source>17</maven.compiler.source>

<maven.compiler.target>17</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.hibernate.orm</groupId>

<artifactId>hibernate-core</artifactId>

<version>6.1.7.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

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

<version>8.0.33</version>

</dependency>

<dependency>

<groupId>jakarta.persistence</groupId>

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

<version>3.1.0</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-api</artifactId>

<version>2.0.9</version>

</dependency>

<dependency>

<groupId>org.slf4j</groupId>

<artifactId>slf4j-simple</artifactId>

<version>2.0.9</version>

</dependency>

</dependencies>

</project>

hibernate.cfg.xml

Place inside: src/main/resources

<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/employee\_db</property>

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

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

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

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

<property name="format\_sql">true</property>

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

<mapping class="com.example.model.Employee"/>

</session-factory>

</hibernate-configuration>

Employee.java

com.example.model

package com.example.model;

import jakarta.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Integer id;

private String name;

private String role;

public Employee() {}

public Employee(String name, String role) {

this.name = name;

this.role = role;

}

// Getters and setters

public Integer getId() { return id; }

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

public String getName() { return name; }

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

public String getRole() { return role; }

public void setRole(String role) { this.role = role; }

}

App.java

com.example

package com.example;

import com.example.model.Employee;

import org.hibernate.\*;

import org.hibernate.cfg.Configuration;

public class App {

public static void main(String[] args) {

SessionFactory factory = new Configuration().configure().buildSessionFactory();

Session session = factory.openSession();

Transaction tx = null;

try {

tx = session.beginTransaction();

Employee employee = new Employee("Alice", "Developer");

session.save(employee);

tx.commit();

System.out.println("Employee inserted!");

} catch (Exception e) {

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

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

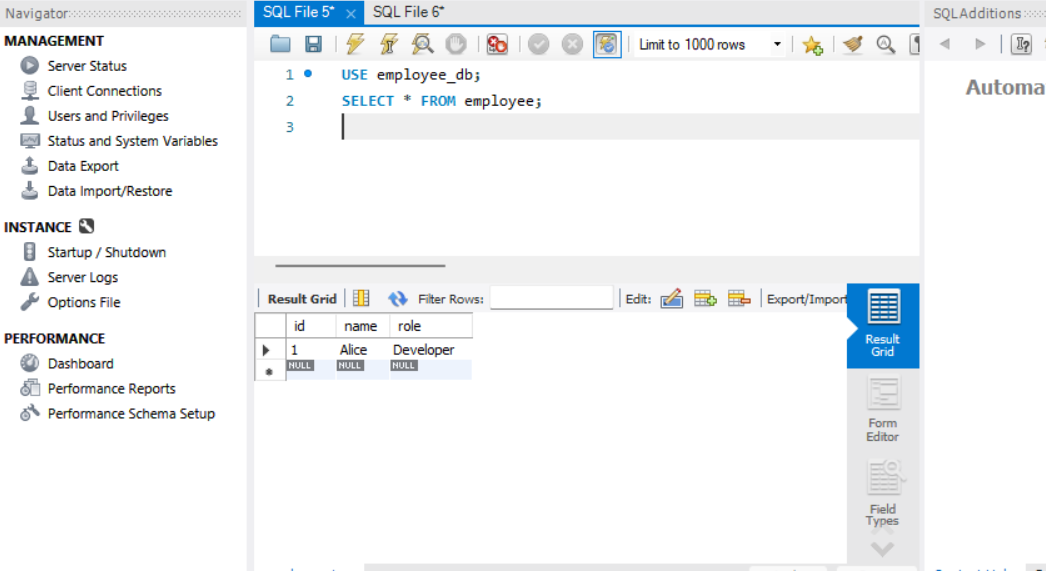
}

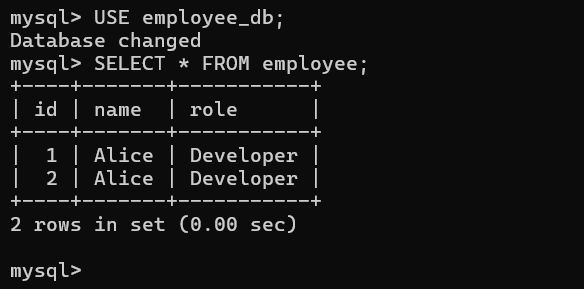
Check Output in MySQL

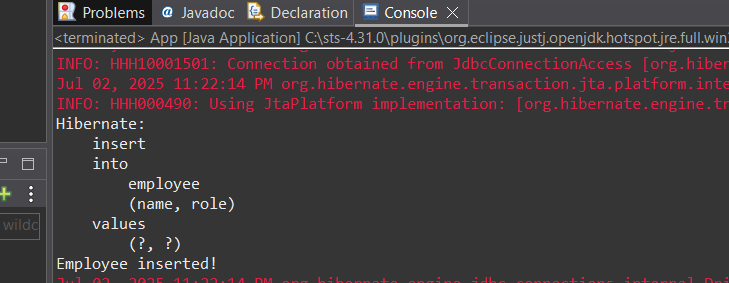
USE employee\_db;

SELECT \* FROM employee;

Output:







G Kullayi Reddy

[Kullayireddy374@gmail.com](mailto:Kullayireddy374@gmail.com)

SupersetID: 6371952