**Hands on 4**

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

**For Hibernate :  
  
Code :  
  
application.properties :**spring.application.name=hibernate

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

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

spring.datasource.username=root

spring.datasource.password=Aathira@14

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

debug=true

**AssetVendor.java :**package com.hibernate.hibernate.entity;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.GeneratedValue;

import jakarta.persistence.GenerationType;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

import jakarta.validation.constraints.Size;

@Entity

@Table(name = "assetvendor")

public class AssetVendor {

@Id

@GeneratedValue(strategy = GenerationType.UUID)

@Column(name = "vendor\_Id", nullable = false, updatable = false)

private java.util.UUID id;

@Column(name = "vendor\_Name")

private String vendorName;

@Column(name = "vendor\_Number")

@Size(min = 10, max = 10)

private String vendorNumber;

@Column(name = "vendor\_Brand")

private String vendorBrand;

@Column(name = "vendor\_Address")

private String vendorAddress;

public AssetVendor() {

}

public java.util.UUID getId() {

return id;

}

public void setId(java.util.UUID id) {

this.id = id;

}

public String getVendorName() {

return vendorName;

}

public void setVendorName(String vendorName) {

this.vendorName = vendorName;

}

public String getVendorNumber() {

return vendorNumber;

}

public void setVendorNumber(String vendorNumber) {

this.vendorNumber = vendorNumber;

}

public String getVendorBrand() {

return vendorBrand;

}

public void setVendorBrand(String vendorBrand) {

this.vendorBrand = vendorBrand;

}

public String getVendorAddress() {

return vendorAddress;

}

public void setVendorAddress(String vendorAddress) {

this.vendorAddress = vendorAddress;

}

}  
  
**AssetVendorConfig.java :**package com.hibernate.hibernate.config;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;

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

import org.springframework.context.annotation.\*;

import org.springframework.core.env.Environment;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

@PropertySource("classpath:application.properties")

public class AssetVendorConfig {

@Autowired

private Environment env;

@Bean

public DataSource dataSource() {

DriverManagerDataSource ds = new DriverManagerDataSource();

ds.setDriverClassName(env.getRequiredProperty("spring.datasource.driver-class-name"));

ds.setUrl(env.getRequiredProperty("spring.datasource.url"));

ds.setUsername(env.getRequiredProperty("spring.datasource.username"));

ds.setPassword(env.getRequiredProperty("spring.datasource.password"));

return ds;

}

@Bean

public LocalSessionFactoryBean sessionFactory() {

LocalSessionFactoryBean factoryBean = new LocalSessionFactoryBean();

factoryBean.setDataSource(dataSource());

factoryBean.setPackagesToScan("com.hibernate.hibernate.entity");

factoryBean.setHibernateProperties(hibernateProperties());

return factoryBean;

}

private Properties hibernateProperties() {

Properties props = new Properties();

props.put("hibernate.dialect", "org.hibernate.dialect.PostgreSQLDialect");

props.put("hibernate.hbm2ddl.auto", "update");

props.put("hibernate.show\_sql", "true");

return props;

}

@Bean

public HibernateTransactionManager transactionManager(SessionFactory sessionFactory) {

HibernateTransactionManager txManager = new HibernateTransactionManager();

txManager.setSessionFactory(sessionFactory);

return txManager;

}

}  
  
**AssetVendorDAOImpl :**package com.hibernate.hibernate.dao.daoimpl;

import java.util.List;

import java.util.UUID;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Repository;

import com.hibernate.hibernate.dao.AssetVendorDAO;

import com.hibernate.hibernate.entity.AssetVendor;

import jakarta.persistence.criteria.CriteriaBuilder;

import jakarta.persistence.criteria.CriteriaQuery;

import jakarta.persistence.criteria.Root;

import jakarta.transaction.Transactional;

@Repository

@Transactional

public class AssetVendorDAOImpl implements AssetVendorDAO {

private static final Logger logger = LoggerFactory.getLogger(AssetVendorDAO.class);

@Autowired

private SessionFactory sessionFactory;

public Session getSession() {

return sessionFactory.getCurrentSession();

}

@Override

public AssetVendor saveVendor(AssetVendor assetVendor) {

getSession().persist(assetVendor);

return assetVendor;

}

@Override

public List<AssetVendor> getAllVendor(AssetVendor assetVendor) {

Session session = getSession();

CriteriaBuilder cb = session.getCriteriaBuilder();

CriteriaQuery<AssetVendor> cq = cb.createQuery(AssetVendor.class);

Root<AssetVendor> root = cq.from(AssetVendor.class);

cq.orderBy(cb.asc(root.get("vendorName")));

return session.createQuery(cq).getResultList();

}

@Override

public void deleteAssetVendor(AssetVendor assetVendor, UUID vendorId) {

Session session = getSession();

AssetVendor vendor = session.get(AssetVendor.class, vendorId);

if (vendor != null) {

getSession().remove(vendor);

}

}

@Override

public AssetVendor updateVendor(AssetVendor assetVendor, UUID vendorId) {

Session session = getSession();

AssetVendor updateVendor = session.get(AssetVendor.class, vendorId);

if (updateVendor != null) {

if (assetVendor.getVendorName() != null) {

updateVendor.setVendorName(assetVendor.getVendorName());

}

if (assetVendor.getVendorNumber() != null) {

updateVendor.setVendorNumber(assetVendor.getVendorNumber());

}

if (assetVendor.getVendorBrand() != null) {

updateVendor.setVendorBrand(assetVendor.getVendorBrand());

}

if (assetVendor.getVendorAddress() != null) {

updateVendor.setVendorAddress(assetVendor.getVendorAddress());

}

session.merge(updateVendor);

logger.info("Vendor Updated Successfully");

} else {

logger.warn("Vendor is not found");

}

return updateVendor;

}

}  
  
**AssetVendorController :**package com.hibernate.hibernate.controller;

import java.util.List;

import java.util.UUID;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.web.bind.annotation.DeleteMapping;

import org.springframework.web.bind.annotation.GetMapping;

import org.springframework.web.bind.annotation.PathVariable;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.PutMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.hibernate.hibernate.dao.AssetVendorDAO;

import com.hibernate.hibernate.entity.AssetVendor;

import com.hibernate.hibernate.service.AssetVendorService;

@RestController

@RequestMapping("/assetvendor")

public class AssetVendorController {

private static final Logger logger = LoggerFactory.getLogger(AssetVendorController.class);

@Autowired

private AssetVendorService assetVendorService;

@Autowired

private AssetVendorDAO assetVendorDAO;

// @PostMapping("/savevendor")

// public String saveVendor(@Valid AssetVendor vendor) {

// assetVendorService.saveVendor(vendor);

// return "Vendor Saved Successfully";

// }

@PostMapping("/persistvendor")

public String saveVendor(@RequestBody AssetVendor assetVendor) {

AssetVendor assetVendorObj = null;

AssetVendor saveVendor = convertToEntity(assetVendor);

try {

logger.debug("Created Vendor Details");

assetVendorObj = assetVendorDAO.saveVendor(assetVendor);

logger.debug("Created Vendor Successfully");

} catch (Exception e) {

logger.debug("Error while creating Vendor : " + e.getMessage());

}

return "Vendor Successfully Created";

}

private AssetVendor convertToEntity(AssetVendor assetVendor) {

AssetVendor entity = new AssetVendor();

entity.setVendorName(assetVendor.getVendorName());

entity.setVendorBrand(assetVendor.getVendorBrand());

entity.setVendorNumber(assetVendor.getVendorNumber());

entity.setVendorAddress(assetVendor.getVendorAddress());

return entity;

}

@GetMapping("/getallvendor")

public List<AssetVendor> getAllVendor(AssetVendor assetVendor) {

List<AssetVendor> assetVendorList = null;

try {

logger.debug("Successfully got the Vendors");

assetVendorList = assetVendorDAO.getAllVendor(assetVendor);

} catch (Exception e) {

logger.debug("Error while getting All Vendor : " + e.getMessage());

}

return assetVendorList;

}

@DeleteMapping("/deletevendor/{vendorId}")

public String deleteVendor(AssetVendor assetVendor, @PathVariable UUID vendorId) {

try {

assetVendorDAO.deleteAssetVendor(assetVendor, vendorId);

return "Vendor Deleted Successfully";

} catch (Exception e) {

logger.debug("Cannot find the Vendor : " + e.getMessage());

return "Cannot Delete the Vendor";

}

}

@PutMapping("/updatevendor/{vendorId}")

public AssetVendor updateVendor(@RequestBody AssetVendor assetVendor, @PathVariable UUID vendorId) {

try {

AssetVendor updatedVendor = assetVendorDAO.updateVendor(assetVendor, vendorId);

return updatedVendor;

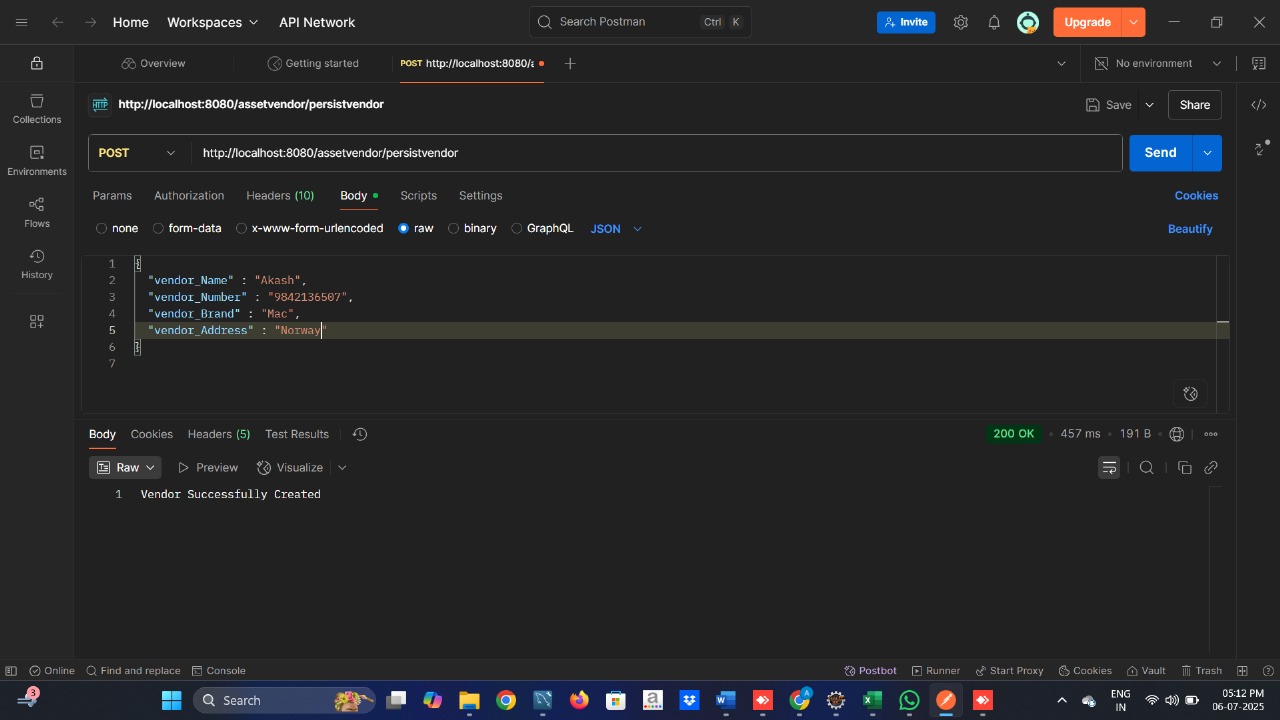
} catch (Exception e) {

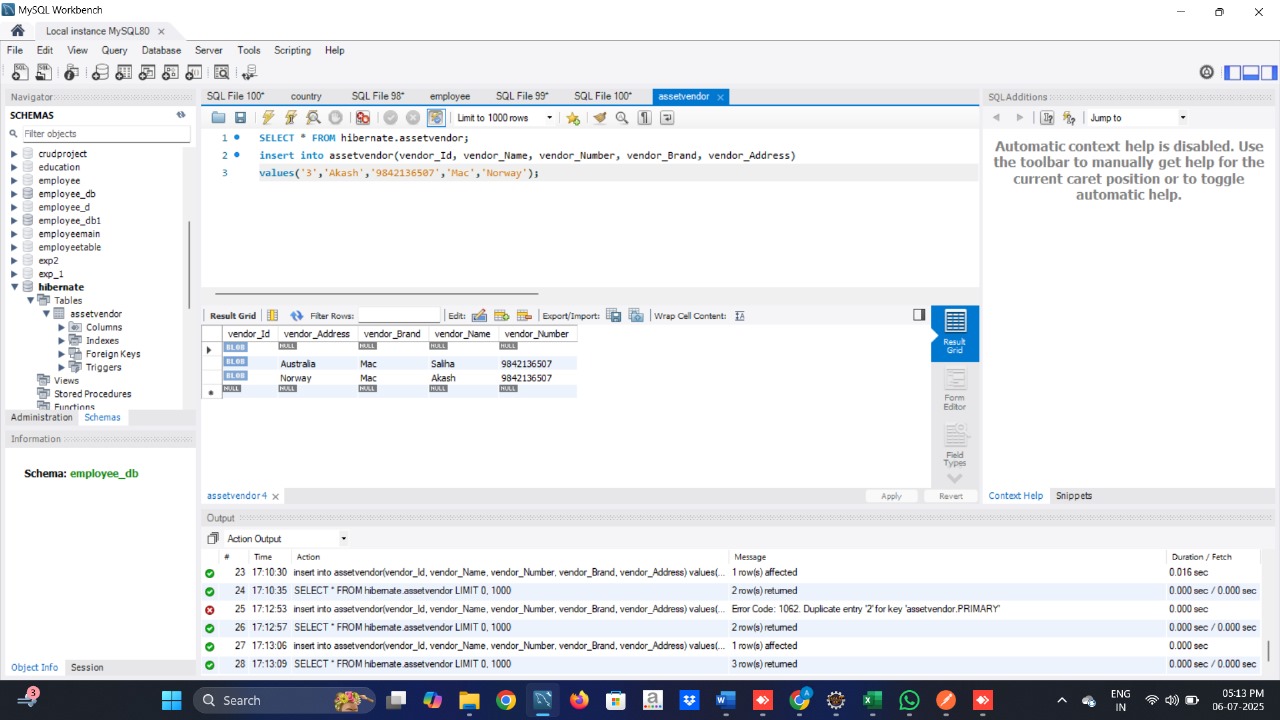
logger.debug("Cannot Update the Vendor");

return assetVendor;

}

}

}  
  
  
**Output :**



**For Spring Data JPA :  
  
Code :  
  
application.properties :**spring.application.name=SpringDataJPA

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

spring.datasource.username=root

spring.datasource.password=Aathira@14

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

**Employee.java :**package com.SpringDataJPA.SpringDataJPA.model;

import jakarta.persistence.\*;

*@Entity*

*@Table*(name = "employee")

public class Employee {

*@Id*

*@GeneratedValue*(strategy = *GenerationType*.***IDENTITY***)

private Integer id;

private String name;

private String department;

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 getDepartment() {

return department;

}

public void setDepartment(String department) {

this.department = department;

}

}

**EmployeeRepository.java :**package com.SpringDataJPA.SpringDataJPA.repository;

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

import com.SpringDataJPA.SpringDataJPA.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer>{

}  
  
**EmployeeService.java :**package com.SpringDataJPA.SpringDataJPA.service;

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

import org.springframework.stereotype.Service;

import com.SpringDataJPA.SpringDataJPA.model.Employee;

import com.SpringDataJPA.SpringDataJPA.repository.EmployeeRepository;

import jakarta.transaction.Transactional;

@Service

public class EmployeeService {

@Autowired

private EmployeeRepository employeeRepository;

@Transactional

public void addEmployee(Employee employee) {

employeeRepository.save(employee);

}

}  
  
**EmployeeController.java :**package com.SpringDataJPA.SpringDataJPA.controller;

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

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import com.SpringDataJPA.SpringDataJPA.model.Employee;

import com.SpringDataJPA.SpringDataJPA.service.EmployeeService;

@RestController

@RequestMapping("employee")

public class EmployeeController {

@Autowired

private EmployeeService employeeService;

@PostMapping("/addemployee")

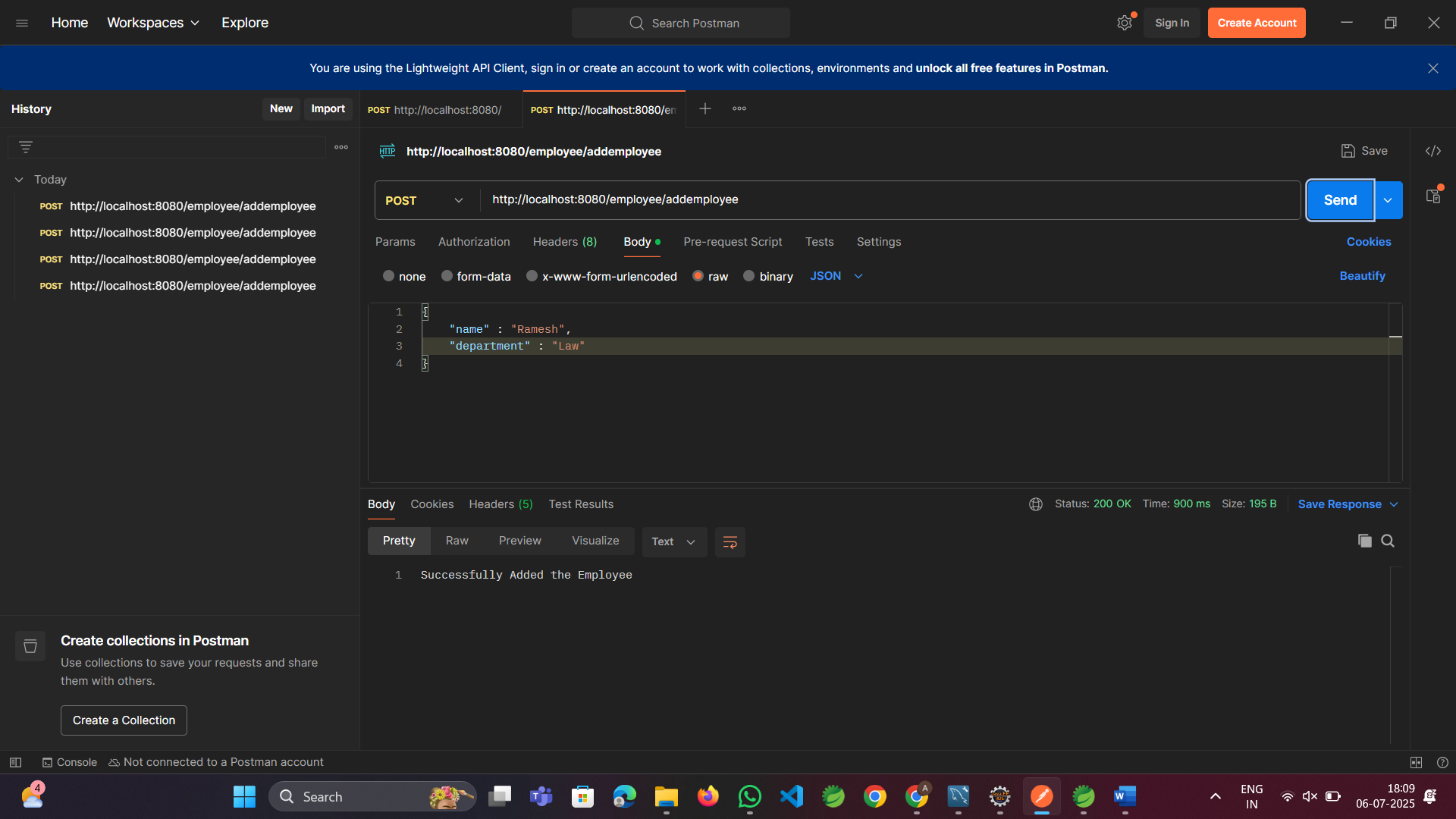
public String addEmployee(@RequestBody Employee employee) {

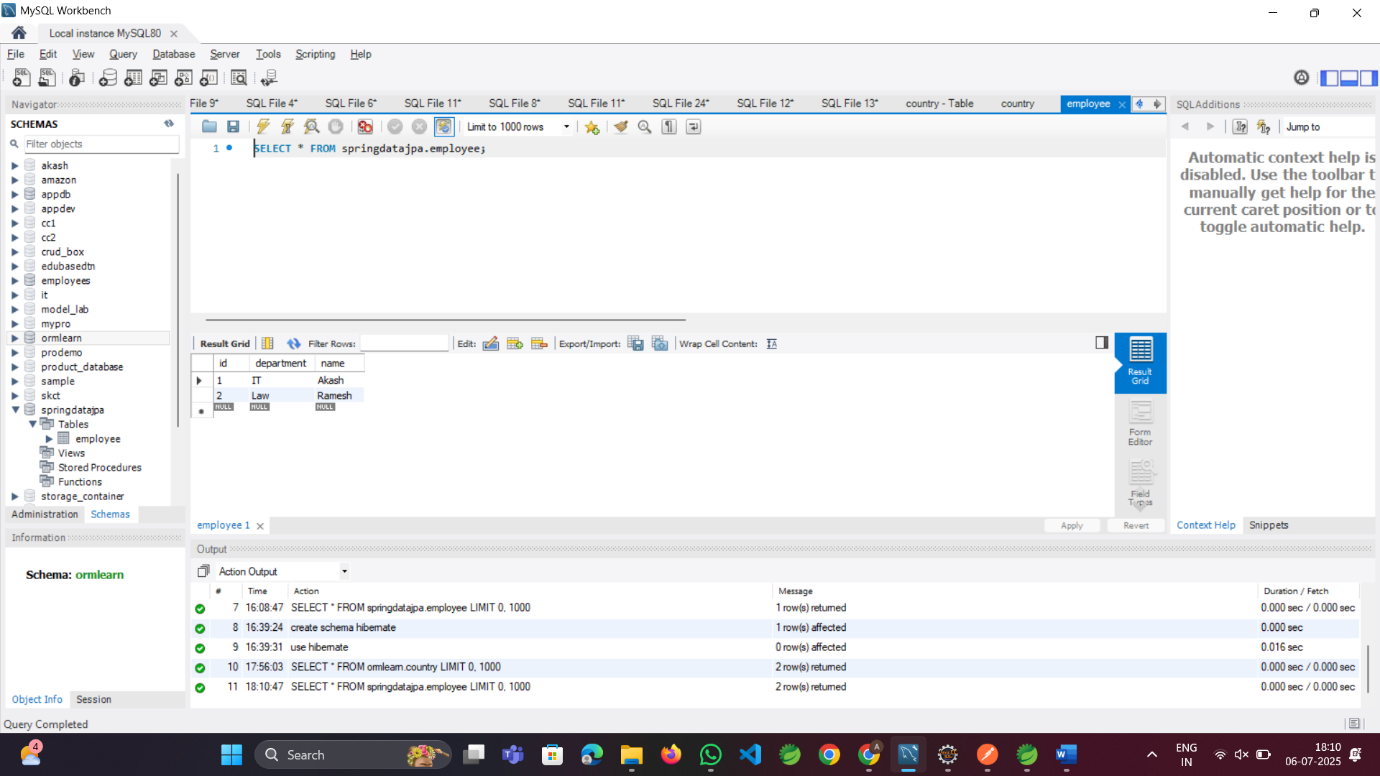
employeeService.addEmployee(employee);

return "Successfully Added the Employee";

}

}  
  
  
**Output :**

****

****