EMPLOYEE RESTFUL API SERVICES

Repository:

package com.verizon.adb.dao;

import java.time.LocalDate;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.verizon.adb.model.Employee;

@Repository

public interface EmployeeRepository extends JpaRepository<Employee, Long> {

boolean existsByEmpMobile(String mobileNumber);

boolean existsByEmpEmail(String emailId);

Employee findByEmpMobile(String empMobile);

Employee findByEmpEmail(String empEmail);

List<Employee> findAllByEmpDept(String empDept);

List<Employee> findAllByDateOfBirth(LocalDate dateOfBirth);

}

Employee Class

package com.verizon.adb.model;

import java.time.LocalDate;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.EnumType;

import javax.persistence.Enumerated;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.validation.constraints.Email;

import javax.validation.constraints.NotEmpty;

import javax.validation.constraints.NotNull;

import javax.validation.constraints.Pattern;

import javax.validation.constraints.Size;

import org.springframework.format.annotation.DateTimeFormat;

import org.springframework.format.annotation.DateTimeFormat.ISO;

@Entity

@Table(name = "employees")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private long empId;

@NotEmpty(message = "Name cannot be empty")

@Size(min = 5, max = 20, message = "Name length must be between 5 to 20 chars")

private String empName;

@Column(name = "basic")

private double empBasic;

@NotEmpty(message = "Email ID cannot be null")

@Email(message = "Invalid Email Id")

private String empEmail;

@NotEmpty(message = "Mobile number cannot be empty")

@Pattern(regexp = "\\d{10}", message = "mobile number can be only 10 digits")

@Column(name = "mob")

private String empMobile;

@NotEmpty(message = "Dept cannot be null")

private String empDept;

@DateTimeFormat(iso = ISO.DATE)

@NotNull(message = "Date of Birth cannot be left blank")

@Column(name = "dob")

private LocalDate dateOfBirth;

@Enumerated(EnumType.STRING)

private Gender gender;

public long getEmpId() {

return empId;

}

public void setEmpId(long empId) {

this.empId = empId;

}

public String getEmpName() {

return empName;

}

public void setEmpName(String empName) {

this.empName = empName;

}

public double getEmpBasic() {

return empBasic;

}

public void setEmpBasic(double empBasic) {

this.empBasic = empBasic;

}

public String getEmpEmail() {

return empEmail;

}

public void setEmpEmail(String empEmail) {

this.empEmail = empEmail;

}

public String getEmpMobile() {

return empMobile;

}

public void setEmpMobile(String empMobile) {

this.empMobile = empMobile;

}

public String getEmpDept() {

return empDept;

}

public void setEmpDept(String empDept) {

this.empDept = empDept;

}

public LocalDate getDateOfBirth() {

return dateOfBirth;

}

public void setDateOfBirth(LocalDate dateOfBirth) {

this.dateOfBirth = dateOfBirth;

}

public Gender getGender() {

return gender;

}

public void setGender(Gender gender) {

this.gender = gender;

}

}

**EmployeeServiceImpl**

package com.verizon.adb.service;

import java.time.LocalDate;

import java.util.List;

import java.util.Optional;

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

import org.springframework.stereotype.Service;

import com.verizon.adb.dao.EmployeeRepository;

import com.verizon.adb.model.Employee;

@Service

public class EmployeeServiceImpl implements EmployeeService {

@Autowired

private EmployeeRepository empRepo;

@Override

public Employee getEmployeeById(long empId) {

Employee emp = null;

Optional<Employee> optEmp = empRepo.findById(empId);

if (optEmp.isPresent()) {

emp = optEmp.get();

}

return emp;

}

@Override

public List<Employee> getAllEmployees() {

return empRepo.findAll();

}

@Override

public Employee addEmployees(Employee employee) {

return empRepo.save(employee);

}

@Override

public Employee updateEmployee(Employee employee) {

return empRepo.save(employee);

}

@Override

public boolean deleteEmployee(long empId) {

boolean isDeleted = false;

if (empRepo.existsById(empId)) {

empRepo.deleteById(empId);

isDeleted = true;

}

return isDeleted;

}

@Override

public boolean existsByEmpMobile(String mobileNumber) {

return empRepo.existsByEmpMobile(mobileNumber);

}

@Override

public boolean existsByEmpEmail(String emailId) {

return empRepo.existsByEmpEmail(emailId);

}

@Override

public Employee findByEmpMobile(String empMobile) {

return empRepo.findByEmpMobile(empMobile);

}

@Override

public Employee findByEmpEmail(String empEmail) {

return empRepo.findByEmpEmail(empEmail);

}

@Override

public List<Employee> findAllByEmpDept(String empDept) {

return empRepo.findAllByEmpDept(empDept);

}

@Override

public List<Employee> findAllByDateOfBirth(LocalDate dateOfBirth) {

return empRepo.findAllByDateOfBirth(dateOfBirth);

}

}

**EMPLOYEE API**

package com.verizon.adb.restapi;

import java.time.LocalDate;

import java.util.Collections;

import java.util.List;

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

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

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

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.verizon.adb.model.Employee;

import com.verizon.adb.service.EmployeeService;

@RestController

@CrossOrigin

@RequestMapping("/employees")

public class EmployeeApi {

@Autowired

public EmployeeService service;

@GetMapping

public ResponseEntity<List<Employee>> getAllEmployees() {

return new ResponseEntity<>(service.getAllEmployees(), HttpStatus.OK);

}

@GetMapping("/{id}")

public ResponseEntity<Employee> getEmployeeById(@PathVariable("id") long empId) {

ResponseEntity<Employee> resp;

Employee e = service.getEmployeeById(empId);

if (e == null) {

resp = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

} else {

return new ResponseEntity<>(e, HttpStatus.OK);

}

return resp;

}

@GetMapping("/{field}/{srhValue}")

public ResponseEntity<List<Employee>> getAllEmployees(@PathVariable("field") String fieldBy,

@PathVariable("srhValue") String searchValue) {

ResponseEntity<List<Employee>> resp = null;

switch (fieldBy) {

case "empMobile":

Employee eByMobNum = service.findByEmpMobile(searchValue);

if (eByMobNum != null) {

resp = new ResponseEntity<>(Collections.singletonList(eByMobNum), HttpStatus.OK);

} else {

resp = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

break;

case "empEmail":

Employee eByEmail = service.findByEmpEmail(searchValue);

if (eByEmail != null) {

resp = new ResponseEntity<>(Collections.singletonList(eByEmail), HttpStatus.OK);

} else {

resp = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

break;

case "empDept":

List<Employee> results = service.findAllByEmpDept(searchValue);

if (results != null && results.size() != 0) {

resp = new ResponseEntity<>(results, HttpStatus.OK);

} else {

resp = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

break;

case "dob":

List<Employee> results1 = service.findAllByDateOfBirth(LocalDate.parse(searchValue));

if (results1 != null && results1.size() != 0) {

resp = new ResponseEntity<>(results1, HttpStatus.OK);

} else {

resp = new ResponseEntity<>(HttpStatus.NOT\_FOUND);

}

break;

default:

break;

}

return resp;

}

@PostMapping

public ResponseEntity<Employee> addEmployee(@RequestBody Employee employee) {

ResponseEntity<Employee> resp = null;

if (service.existsByEmpEmail(employee.getEmpEmail())) {

resp = new ResponseEntity<Employee>(HttpStatus.ALREADY\_REPORTED);

}

if (service.existsByEmpMobile(employee.getEmpMobile())) {

resp = new ResponseEntity<Employee>(HttpStatus.ALREADY\_REPORTED);

}

if (resp == null) {

Employee e = service.addEmployees(employee);

if (e == null) {

resp = new ResponseEntity<Employee>(HttpStatus.BAD\_REQUEST);

} else {

resp = new ResponseEntity<Employee>(e, HttpStatus.OK);

}

}

return resp;

}

@PutMapping

public ResponseEntity<Employee> updateContact(@RequestBody Employee employee) {

ResponseEntity<Employee> resp = null;

Employee e = service.getEmployeeById(employee.getEmpId());

if (!employee.getEmpEmail().equals(e.getEmpEmail())) {

if (service.existsByEmpEmail(employee.getEmpEmail())) {

resp = new ResponseEntity<Employee>(HttpStatus.ALREADY\_REPORTED);

}

}

if (!employee.getEmpMobile().equals(e.getEmpMobile())) {

if (service.existsByEmpMobile(employee.getEmpMobile())) {

resp = new ResponseEntity<Employee>(HttpStatus.ALREADY\_REPORTED);

}

}

if (resp == null) {

Employee e1 = service.updateEmployee(employee);

if (e1 == null) {

resp = new ResponseEntity<Employee>(HttpStatus.BAD\_REQUEST);

} else {

resp = new ResponseEntity<Employee>(e1, HttpStatus.OK);

}

}

return resp;

}

@DeleteMapping("/{id}")

public ResponseEntity<Void> deleteEmployee(@PathVariable("id") long empId) {

ResponseEntity<Void> resp = null;

if (service.deleteEmployee(empId)) {

resp = new ResponseEntity<Void>(HttpStatus.OK);

} else {

resp = new ResponseEntity<Void>(HttpStatus.NOT\_FOUND);

}

return resp;

}

}

**APPLICATION PROPERTIES**

server.port = 7070

spring.datasource.url=jdbc:mysql://localhost:3306/adb2

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

spring.datasource.username = root

spring.datasource.password = admin

spring.jpa.show-sql = true

spring.jpa.hibernate.ddl-auto = update

spring.jpa.properties.hibernate.dialect = org.hibernate.dialect.MySQL5Dialect

**SCREEN SHOTS:**





