**Additional Exercises:  
  
1.Find a Country Based on Country Code**@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

Optional<Country> result = countryRepository.findById(countryCode);

if (!result.isPresent()) {

throw new CountryNotFoundException("Country Not Found: " + countryCode);

}

return result.get();

}  
  
---------------------------------------------------------------------------------------------------------------------------------------------------  
package com.cognizant.springlearn.service.exception;

public class CountryNotFoundException extends Exception {

public CountryNotFoundException(String message) {

super(message);

}

}  
--------------------------------------------------------------------------------------------------------------------------------------------------  
Add a New Country  
@Transactional

public void addCountry(Country country) {

countryRepository.save(country);

}  
  
private static void testFindCountryByCode() throws CountryNotFoundException {

LOGGER.info("Start");

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

LOGGER.info("End");

}

private static void testAddCountry() {

LOGGER.info("Start");

Country country = new Country();

country.setCode("ZZ");

country.setName("Testland");

countryService.addCountry(country);

LOGGER.debug("Added Country: {}", country);

LOGGER.info("End");

}  
  
--------------------------------------------------------------------------------------------------------------------------------------------------  
  
package com.cognizant.ormlearn;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

import com.cognizant.springlearn.service.exception.CountryNotFoundException;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import java.util.List;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryService countryService;

public static void main(String[] args) throws CountryNotFoundException {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryService = context.getBean(CountryService.class);

testGetAllCountries();

testFindCountryByCode();

testAddCountry();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

List<Country> countries = countryService.getAllCountries();

LOGGER.debug("Countries: {}", countries);

LOGGER.info("End");

}

private static void testFindCountryByCode() throws CountryNotFoundException {

LOGGER.info("Start");

Country country = countryService.findCountryByCode("IN");

LOGGER.debug("Country: {}", country);

LOGGER.info("End");

}

private static void testAddCountry() {

LOGGER.info("Start");

Country country = new Country();

country.setCode("ZZ");

country.setName("Testland");

countryService.addCountry(country);

LOGGER.debug("Added Country: {}", country);

LOGGER.info("End");

}

}

-------------------------------------------------------------------------------------------------------------------------------------------------  
**Query Methods in Spring Data JPA,** **Demonstrate implementation of Query Methods feature of Spring Data JPA, Demonstrate implementation of O/R Mapping, Demonstrate writing Hibernate Query Language and Native Query**package com.cognizant.ormlearn;

import java.util.\*;

import javax.persistence.\*;

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

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.transaction.annotation.Transactional;

@SpringBootApplication

public class OrmLearnApplication {

private static final Logger LOGGER = LoggerFactory.getLogger(OrmLearnApplication.class);

private static CountryRepository countryRepository;

private static EmployeeRepository employeeRepository;

public static void main(String[] args) {

ApplicationContext context = SpringApplication.run(OrmLearnApplication.class, args);

countryRepository = context.getBean(CountryRepository.class);

employeeRepository = context.getBean(EmployeeRepository.class);

testQueryMethods();

testHqlAndNativeQueries();

}

private static void testQueryMethods() {

LOGGER.info("---- Query Methods ----");

List<Country> containing = countryRepository.findByNameContaining("ou");

List<Country> containingSorted = countryRepository.findByNameContainingOrderByNameAsc("ou");

List<Country> startingWith = countryRepository.findByNameStartingWith("Z");

LOGGER.debug("Containing 'ou': {}", containing);

LOGGER.debug("Containing 'ou' sorted: {}", containingSorted);

LOGGER.debug("Starting with 'Z': {}", startingWith);

}

private static void testHqlAndNativeQueries() {

LOGGER.info("---- HQL and Native Query ----");

List<Employee> permanentEmployees = employeeRepository.getAllPermanentEmployees();

List<Employee> detailedEmployees = employeeRepository.getAllPermanentEmployeesWithDeptAndSkills();

List<Employee> allEmployees = employeeRepository.getAllEmployeesNative();

LOGGER.debug("Permanent Employees: {}", permanentEmployees);

LOGGER.debug("Permanent with Details: {}", detailedEmployees);

LOGGER.debug("All Employees (Native): {}", allEmployees);

}

}

@Entity

@Table(name = "country")

class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

public String getCode() { return code; }

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

public String getName() { return name; }

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

public String toString() { return "Country [code=" + code + ", name=" + name + "]"; }

}

@Entity

class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@OneToMany(mappedBy = "department", fetch = FetchType.EAGER)

private Set<Employee> employeeList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Set<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }

public String toString() { return "Department [id=" + id + ", name=" + name + "]"; }

}

@Entity

class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public Set<Employee> getEmployeeList() { return employeeList; }

public void setEmployeeList(Set<Employee> employeeList) { this.employeeList = employeeList; }

public String toString() { return "Skill [id=" + id + ", name=" + name + "]"; }

}

@Entity

class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

private String name;

private double salary;

private boolean permanent;

@Temporal(TemporalType.DATE)

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "em\_dp\_id")

private Department department;

@ManyToMany

@JoinTable(name = "employee\_skill",

joinColumns = @JoinColumn(name = "es\_em\_id"),

inverseJoinColumns = @JoinColumn(name = "es\_sk\_id"))

private Set<Skill> skillList;

public int getId() { return id; }

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

public String getName() { return name; }

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

public double getSalary() { return salary; }

public void setSalary(double salary) { this.salary = salary; }

public boolean isPermanent() { return permanent; }

public void setPermanent(boolean permanent) { this.permanent = permanent; }

public Date getDateOfBirth() { return dateOfBirth; }

public void setDateOfBirth(Date dateOfBirth) {

public Department getDepartment() { return department; }

public void setDepartment(Department department) { this.department = department; }

public Set<Skill> getSkillList() { return skillList; }

public void setSkillList(Set<Skill> skillList) { this.skillList = skillList; }

public String toString() { return "Employee [id=" + id + ", name=" + name + "]"; }

}

@Repository

interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword);

List<Country> findByNameContainingOrderByNameAsc(String keyword);

List<Country> findByNameStartingWith(String prefix);

}

@Repository

interface EmployeeRepository extends JpaRepository<Employee, Integer> {

@Query("SELECT e FROM Employee e WHERE e.permanent = 1")

List<Employee> getAllPermanentEmployees();

@Query("SELECT e FROM Employee e LEFT JOIN FETCH e.department d LEFT JOIN FETCH e.skillList WHERE e.permanent = 1")

List<Employee> getAllPermanentEmployeesWithDeptAndSkills();

@Query(value = "SELECT \* FROM employee", nativeQuery = true)

List<Employee> getAllEmployeesNative();

}

@Service

class CountryService {

@Autowired

private CountryRepository countryRepository;

@Transactional

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

}