**MODULE 6 Spring Data JPA with Spring Boot, Hibernate**

**Hands on 1**

**Spring Data JPA - Quick Example**

package com.cognizant.ormlearn.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_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; }

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

package com.cognizant.ormlearn.repository;

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

import org.springframework.stereotype.Repository;

import com.cognizant.ormlearn.model.Country;

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

}

package com.cognizant.ormlearn;

import java.util.List;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import com.cognizant.ormlearn.model.Country;

import com.cognizant.ormlearn.service.CountryService;

@SpringBootApplication

public class OrmLearnApplication {

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

private static CountryService countryService;

public static void main(String[] args) {

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

countryService = context.getBean(CountryService.class);

testGetAllCountries();

}

private static void testGetAllCountries() {

LOGGER.info("Start");

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

System.out.println("✅ Output from DB:");

countries.forEach(System.out::println);

LOGGER.info("End");

}

}

spring.application.name=orm-learn

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

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

spring.datasource.username=root

spring.datasource.password=Lalithsai@14

spring.jpa.hibernate.ddl-auto=update

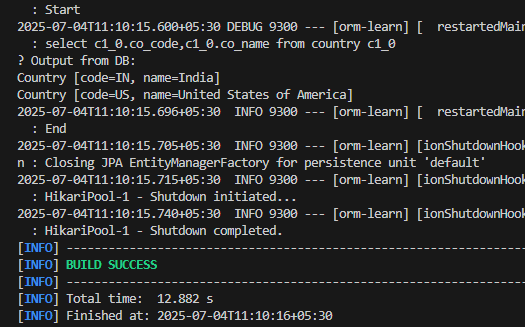
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect

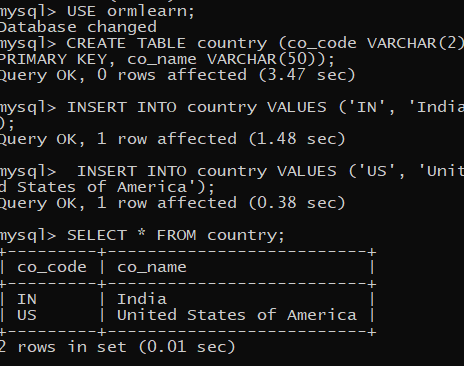
logging.level.org.springframework=info

logging.level.com.cognizant=debug

logging.level.org.hibernate.SQL=trace

logging.level.org.hibernate.type.descriptor.sql=trace





**Hands on 2**

**Hibernate XML Config implementation walk through**

package com.cognizant.hibernate.model;

public class Country {

private String code;

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;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

package com.cognizant.hibernate;

import java.util.List;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import com.cognizant.hibernate.model.Country;

public class App {

public static void main(String[] args) {

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

Session session = factory.openSession();

try {

session.beginTransaction();

List<Country> countries = session.createQuery("from Country", Country.class).list();

for (Country country : countries) {

System.out.println(country);

}

session.getTransaction().commit();

} catch (Exception e) {

session.getTransaction().rollback();

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

<!DOCTYPE hibernate-mapping PUBLIC

"-//Hibernate/Hibernate Mapping DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-mapping-3.0.dtd">

<hibernate-mapping>

<class name="com.cognizant.hibernate.model.Country" table="country">

<id name="code" column="co\_code"/>

<property name="name" column="co\_name"/>

</class>

</hibernate-mapping>

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<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/ormlearn</property>

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

<property name="hibernate.connection.password">Lalithsai@14</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

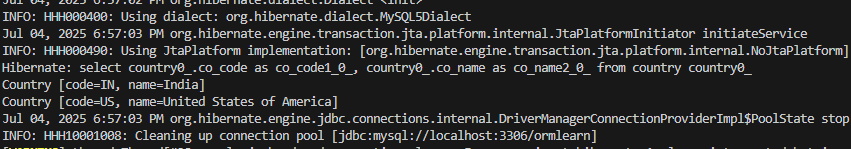
<property name="show\_sql">true</property>

<!-- Mapping file -->

<mapping resource="country.hbm.xml"/>

</session-factory>

</hibernate-configuration>



**Hands on 3**

**Hibernate Annotation Config implementation walk through**    
package com.cognizant.hibernate.model;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

import javax.persistence.Column;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_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;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

package com.cognizant.hibernate;

import com.cognizant.hibernate.model.Country;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

import java.util.List;

public class App {

public static void main(String[] args) {

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

Session session = factory.openSession();

try {

session.beginTransaction();

List<Country> countries = session.createQuery("from Country", Country.class).list();

for (Country country : countries) {

System.out.println(country);

}

session.getTransaction().commit();

} catch (Exception e) {

session.getTransaction().rollback();

e.printStackTrace();

} finally {

session.close();

factory.close();

}

}

}

<!DOCTYPE hibernate-configuration PUBLIC

"-//Hibernate/Hibernate Configuration DTD 3.0//EN"

"http://hibernate.sourceforge.net/hibernate-configuration-3.0.dtd">

<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/ormlearn</property>

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

<property name="hibernate.connection.password">Lalithsai@14</property>

<property name="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</property>

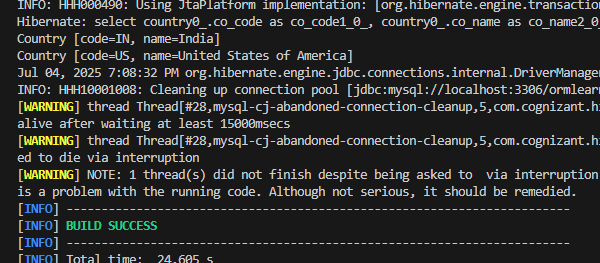
<property name="show\_sql">true</property>

<!-- No XML mapping needed -->

<mapping class="com.cognizant.hibernate.model.Country"/>

</session-factory>

</hibernate-configuration>



**2**

**Hands on 1**

**Write queries on country table using Query Methods**    
package com.cognizant.springdatajpa.model;

import jakarta.persistence.Column;

import jakarta.persistence.Entity;

import jakarta.persistence.Id;

import jakarta.persistence.Table;

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "co\_code")

private String code;

@Column(name = "co\_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;

}

@Override

public String toString() {

return "Country [code=" + code + ", name=" + name + "]";

}

}

package com.cognizant.springdatajpa.repository;

import com.cognizant.springdatajpa.model.Country;

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

import java.util.List;

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String name);

List<Country> findByNameContainingOrderByNameAsc(String name);

List<Country> findByNameStartingWith(String prefix);

}

package com.cognizant.springdatajpa;

import com.cognizant.springdatajpa.model.Country;

import com.cognizant.springdatajpa.repository.CountryRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.util.List;

@SpringBootApplication

public class SpringdataCountryDemoApplication implements CommandLineRunner {

@Autowired

private CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.run(SpringdataCountryDemoApplication.class, args);

}

@Override

public void run(String... args) {

System.out.println(" 1. Countries containing 'ou'");

List<Country> list1 = countryRepository.findByNameContaining("ou");

list1.forEach(c -> System.out.println(c.getCode() + " " + c.getName()));

System.out.println("\n2. Countries containing 'ou' sorted by name ascending");

List<Country> list2 = countryRepository.findByNameContainingOrderByNameAsc("ou");

list2.forEach(c -> System.out.println(c.getCode() + " " + c.getName()));

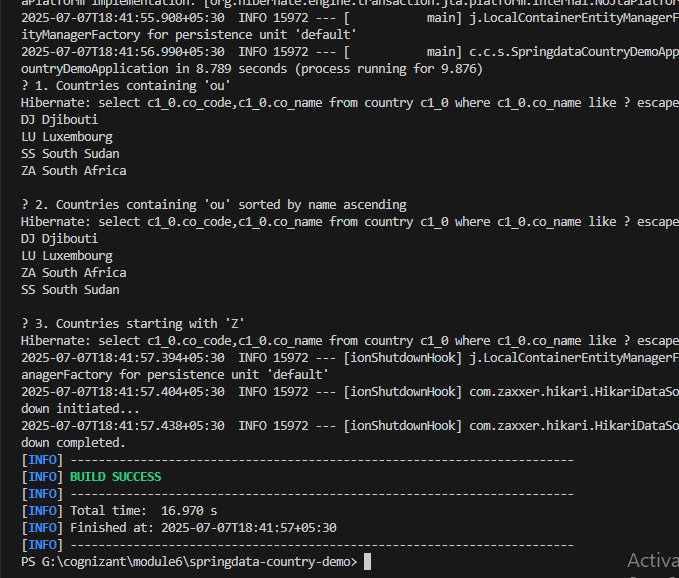
System.out.println("\n 3. Countries starting with 'Z'");

List<Country> list3 = countryRepository.findByNameStartingWith("Z");

list3.forEach(c -> System.out.println(c.getCode() + " " + c.getName()));

}

}



**Hands on 2**

**Write queries on stock table using Query Methods**    
   
With one year stock data of Facebook, Google and Netflix, we need to implement Spring Data JPA Query Methods for the following scenarios:   
   
**Sample Data**   
Sample data for implementing this hands on is provided to you in the platform

package com.cognizant.springdatajpa.model;

import jakarta.persistence.\*;

import java.math.BigDecimal;

import java.time.LocalDate;

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int stId;

private String stCode;

private LocalDate stDate;

private BigDecimal stOpen;

private BigDecimal stClose;

private BigDecimal stVolume;

// Getters and Setters

public int getStId() {

return stId;

}

public void setStId(int stId) {

this.stId = stId;

}

public String getStCode() {

return stCode;

}

public void setStCode(String stCode) {

this.stCode = stCode;

}

public LocalDate getStDate() {

return stDate;

}

public void setStDate(LocalDate stDate) {

this.stDate = stDate;

}

public BigDecimal getStOpen() {

return stOpen;

}

public void setStOpen(BigDecimal stOpen) {

this.stOpen = stOpen;

}

public BigDecimal getStClose() {

return stClose;

}

public void setStClose(BigDecimal stClose) {

this.stClose = stClose;

}

public BigDecimal getStVolume() {

return stVolume;

}

public void setStVolume(BigDecimal stVolume) {

this.stVolume = stVolume;

}

}

package com.cognizant.springdatajpa.repository;

import com.cognizant.springdatajpa.model.Stock;

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

import org.springframework.stereotype.Repository;

import java.time.LocalDate;

import java.util.List;

@Repository

public interface StockRepository extends JpaRepository<Stock, Integer> {

List<Stock> findByStCodeAndStDateBetween(String stCode, LocalDate start, LocalDate end);

List<Stock> findByStCodeAndStCloseGreaterThan(String stCode, double price);

List<Stock> findTop3ByOrderByStVolumeDesc();

List<Stock> findTop3ByStCodeOrderByStCloseAsc(String stCode);

}

package com.cognizant.springdatajpa;

import com.cognizant.springdatajpa.model.Stock;

import com.cognizant.springdatajpa.repository.StockRepository;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import java.time.LocalDate;

import java.util.List;

@SpringBootApplication

public class SpringdataCountryDemoApplication implements CommandLineRunner {

@Autowired

private StockRepository stockRepository;

public static void main(String[] args) {

SpringApplication.run(SpringdataCountryDemoApplication.class, args);

}

@Override

public void run(String... args) throws Exception {

System.out.println("FB stocks in Sep 2019:");

List<Stock> fbStocks = stockRepository.findByStCodeAndStDateBetween(

"FB",

LocalDate.of(2019, 9, 1),

LocalDate.of(2019, 9, 30)

);

fbStocks.forEach(System.out::println);

System.out.println("\nGOOGL stocks with close > 1250:");

List<Stock> googlStocks = stockRepository.findByStCodeAndStCloseGreaterThan("GOOGL", 1250);

googlStocks.forEach(System.out::println);

System.out.println("\nTop 3 stock volumes:");

List<Stock> topVolumes = stockRepository.findTop3ByOrderByStVolumeDesc();

topVolumes.forEach(System.out::println);

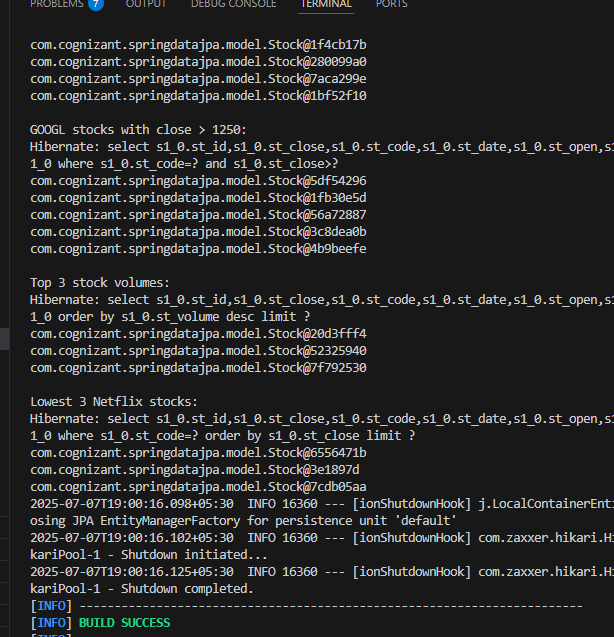
System.out.println("\nLowest 3 Netflix stocks:");

List<Stock> lowNetflix = stockRepository.findTop3ByStCodeOrderByStCloseAsc("NFLX");

lowNetflix.forEach(System.out::println);

}

}



**Hands on 3**

**Create payroll tables and bean mapping**    
   
To demonstrate one to many, many to one and many to many relationships in Hibernate, a schema with entities employee, department and skill will be used. In this hands on we will setup the tables and data, which forms the basis for learning the mappings in Hibernate.   
   
**Schema Structure**   
   
   
   
Follow steps below to create necessary tables:

package com.cognizant.springdatajpa.model;

import jakarta.persistence.\*;

import java.util.List;

@Entity

@Table(name = "dept\_demo")

public class Department {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@OneToMany(mappedBy = "department")

private List<Employee> employeeList;

// Getters, Setters, toString

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 List<Employee> getEmployeeList() {

return employeeList;

}

public void setEmployeeList(List<Employee> employeeList) {

this.employeeList = employeeList;

}

@Override

public String toString() {

return "Department [id=" + id + ", name=" + name + "]";

}

}

package com.cognizant.springdatajpa.model;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.List;

@Entity

@Table(name = "emp\_demo")

public class Employee {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@Column(name = "salary")

private double salary;

@Column(name = "permanent")

private boolean permanent;

@Column(name = "date\_of\_birth")

private Date dateOfBirth;

@ManyToOne

@JoinColumn(name = "dept\_id")

private Department department;

@ManyToMany

@JoinTable(

name = "emp\_skill\_demo",

joinColumns = @JoinColumn(name = "emp\_id"),

inverseJoinColumns = @JoinColumn(name = "skill\_id")

)

private List<Skill> skillList;

// Getters, Setters, toString

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) {

this.dateOfBirth = dateOfBirth;

}

public Department getDepartment() {

return department;

}

public void setDepartment(Department department) {

this.department = department;

}

public List<Skill> getSkillList() {

return skillList;

}

public void setSkillList(List<Skill> skillList) {

this.skillList = skillList;

}

@Override

public String toString() {

return "Employee [id=" + id + ", name=" + name + ", salary=" + salary + ", permanent=" + permanent

+ ", dateOfBirth=" + dateOfBirth + ", department=" + department + "]";

}

}

package com.cognizant.springdatajpa.model;

import jakarta.persistence.\*;

import java.util.Set;

@Entity

@Table(name = "skill\_demo")

public class Skill {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "name")

private String name;

@ManyToMany(mappedBy = "skillList")

private Set<Employee> employeeList;

// Getters, setters, toString()

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;

}

@Override

public String toString() {

return "Skill [id=" + id + ", name=" + name + "]";

}

}

package com.cognizant.springdatajpa.repository;

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

import com.cognizant.springdatajpa.model.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {

}

package com.cognizant.springdatajpa.repository;

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

import com.cognizant.springdatajpa.model.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

}

package com.cognizant.springdatajpa.repository;

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

import com.cognizant.springdatajpa.model.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {

}

package com.cognizant.springdatajpa;

import java.util.Optional;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.transaction.annotation.Transactional;

import com.cognizant.springdatajpa.model.Employee;

import com.cognizant.springdatajpa.repository.EmployeeRepository;

@SpringBootApplication

public class SpringdataCountryDemoApplication implements CommandLineRunner {

@Autowired

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.run(SpringdataCountryDemoApplication.class, args);

}

@Override

@Transactional

public void run(String... args) throws Exception {

Optional<Employee> employeeOptional = employeeRepository.findById(1);

if (employeeOptional.isPresent()) {

Employee employee = employeeOptional.get();

System.out.println("Employee: " + employee);

System.out.println("Department: " + employee.getDepartment());

System.out.println("Skills:");

employee.getSkillList().forEach(System.out::println);

} else {

System.out.println("Employee with ID 1 not found.");

}

}

}

