**Hands on 5**

**Implement services for managing Country**   
  
An application requires for features to be implemented with regards to country. These features needs to be supported by implementing them as service using Spring Data JPA.

* Find a country based on country code
* Add new country
* Update country
* Delete country
* Find list of countries matching a partial country name

Before starting the implementation of the above features, there are few configuration and data population that needs to be incorporated. Please refer each topic below and implement the same.   
  
**Explanation for Hibernate table creation configuration**

* Moreover the ddl-auto defines how hibernate behaves if a specific table or column is not present in the database.
  + create - drops existing tables data and structure, then creates new tables
  + validate - check if the table and columns exist or not, throws an exception if a matching table or column is not found
  + update - if a table does not exists, it creates a new table; if a column does not exists, it creates a new column
  + create-drop - creates the table, once all operations are completed, the table is dropped

# Hibernate ddl auto (create, create-drop, update, validate)

spring.jpa.hibernate.ddl-auto=validate

Populate country table

* Delete all the records in Country table and then use the below script to create the actual list of all countries in our world.

insert into country (co\_code, co\_name) values ("AF", "Afghanistan");

insert into country (co\_code, co\_name) values ("AL", "Albania");

insert into country (co\_code, co\_name) values ("DZ", "Algeria");

insert into country (co\_code, co\_name) values ("AS", "American Samoa");

insert into country (co\_code, co\_name) values ("AD", "Andorra");

insert into country (co\_code, co\_name) values ("AO", "Angola");

insert into country (co\_code, co\_name) values ("AI", "Anguilla");

insert into country (co\_code, co\_name) values ("AQ", "Antarctica");

insert into country (co\_code, co\_name) values ("AG", "Antigua and Barbuda");

insert into country (co\_code, co\_name) values ("AR", "Argentina");

insert into country (co\_code, co\_name) values ("AM", "Armenia");

insert into country (co\_code, co\_name) values ("AW", "Aruba");

insert into country (co\_code, co\_name) values ("AU", "Australia");

insert into country (co\_code, co\_name) values ("AT", "Austria");

insert into country (co\_code, co\_name) values ("AZ", "Azerbaijan");

insert into country (co\_code, co\_name) values ("BS", "Bahamas");

insert into country (co\_code, co\_name) values ("BH", "Bahrain");

insert into country (co\_code, co\_name) values ("BD", "Bangladesh");

insert into country (co\_code, co\_name) values ("BB", "Barbados");

insert into country (co\_code, co\_name) values ("BY", "Belarus");

insert into country (co\_code, co\_name) values ("BE", "Belgium");

insert into country (co\_code, co\_name) values ("BZ", "Belize");

insert into country (co\_code, co\_name) values ("BJ", "Benin");

insert into country (co\_code, co\_name) values ("BM", "Bermuda");

insert into country (co\_code, co\_name) values ("BT", "Bhutan");

insert into country (co\_code, co\_name) values ("BO", "Bolivia, Plurinational State of");

insert into country (co\_code, co\_name) values ("BQ", "Bonaire, Sint Eustatius and Saba");

insert into country (co\_code, co\_name) values ("BA", "Bosnia and Herzegovina");

insert into country (co\_code, co\_name) values ("BW", "Botswana");

insert into country (co\_code, co\_name) values ("BV", "Bouvet Island");

insert into country (co\_code, co\_name) values ("BR", "Brazil");

insert into country (co\_code, co\_name) values ("IO", "British Indian Ocean Territory");

insert into country (co\_code, co\_name) values ("BN", "Brunei Darussalam");

insert into country (co\_code, co\_name) values ("BG", "Bulgaria");

insert into country (co\_code, co\_name) values ("BF", "Burkina Faso");

insert into country (co\_code, co\_name) values ("BI", "Burundi");

insert into country (co\_code, co\_name) values ("KH", "Cambodia");

insert into country (co\_code, co\_name) values ("CM", "Cameroon");

insert into country (co\_code, co\_name) values ("CA", "Canada");

insert into country (co\_code, co\_name) values ("CV", "Cape Verde");

insert into country (co\_code, co\_name) values ("KY", "Cayman Islands");

insert into country (co\_code, co\_name) values ("CF", "Central African Republic");

insert into country (co\_code, co\_name) values ("TD", "Chad");

insert into country (co\_code, co\_name) values ("CL", "Chile");

insert into country (co\_code, co\_name) values ("CN", "China");

insert into country (co\_code, co\_name) values ("CX", "Christmas Island");

insert into country (co\_code, co\_name) values ("CC", "Cocos (Keeling) Islands");

insert into country (co\_code, co\_name) values ("CO", "Colombia");

insert into country (co\_code, co\_name) values ("KM", "Comoros");

insert into country (co\_code, co\_name) values ("CG", "Congo");

insert into country (co\_code, co\_name) values ("CD", "Congo, the Democratic Republic of the");

insert into country (co\_code, co\_name) values ("CK", "Cook Islands");

insert into country (co\_code, co\_name) values ("CR", "Costa Rica");

insert into country (co\_code, co\_name) values ("HR", "Croatia");

insert into country (co\_code, co\_name) values ("CU", "Cuba");

insert into country (co\_code, co\_name) values ("CW", "Curaçao");

insert into country (co\_code, co\_name) values ("CY", "Cyprus");

insert into country (co\_code, co\_name) values ("CZ", "Czech Republic");

insert into country (co\_code, co\_name) values ("CI", "Côte d'Ivoire");

insert into country (co\_code, co\_name) values ("DK", "Denmark");

insert into country (co\_code, co\_name) values ("DJ", "Djibouti");

insert into country (co\_code, co\_name) values ("DM", "Dominica");

insert into country (co\_code, co\_name) values ("DO", "Dominican Republic");

insert into country (co\_code, co\_name) values ("EC", "Ecuador");

insert into country (co\_code, co\_name) values ("EG", "Egypt");

insert into country (co\_code, co\_name) values ("SV", "El Salvador");

insert into country (co\_code, co\_name) values ("GQ", "Equatorial Guinea");

insert into country (co\_code, co\_name) values ("ER", "Eritrea");

insert into country (co\_code, co\_name) values ("EE", "Estonia");

insert into country (co\_code, co\_name) values ("ET", "Ethiopia");

insert into country (co\_code, co\_name) values ("FK", "Falkland Islands (Malvinas)");

insert into country (co\_code, co\_name) values ("FO", "Faroe Islands");

insert into country (co\_code, co\_name) values ("FJ", "Fiji");

insert into country (co\_code, co\_name) values ("FI", "Finland");

insert into country (co\_code, co\_name) values ("FR", "France");

insert into country (co\_code, co\_name) values ("GF", "French Guiana");

insert into country (co\_code, co\_name) values ("PF", "French Polynesia");

insert into country (co\_code, co\_name) values ("TF", "French Southern Territories");

insert into country (co\_code, co\_name) values ("GA", "Gabon");

insert into country (co\_code, co\_name) values ("GM", "Gambia");

insert into country (co\_code, co\_name) values ("GE", "Georgia");

insert into country (co\_code, co\_name) values ("DE", "Germany");

insert into country (co\_code, co\_name) values ("GH", "Ghana");

insert into country (co\_code, co\_name) values ("GI", "Gibraltar");

insert into country (co\_code, co\_name) values ("GR", "Greece");

insert into country (co\_code, co\_name) values ("GL", "Greenland");

insert into country (co\_code, co\_name) values ("GD", "Grenada");

insert into country (co\_code, co\_name) values ("GP", "Guadeloupe");

insert into country (co\_code, co\_name) values ("GU", "Guam");

insert into country (co\_code, co\_name) values ("GT", "Guatemala");

insert into country (co\_code, co\_name) values ("GG", "Guernsey");

insert into country (co\_code, co\_name) values ("GN", "Guinea");

insert into country (co\_code, co\_name) values ("GW", "Guinea-Bissau");

insert into country (co\_code, co\_name) values ("GY", "Guyana");

insert into country (co\_code, co\_name) values ("HT", "Haiti");

insert into country (co\_code, co\_name) values ("HM", "Heard Island and McDonald Islands");

insert into country (co\_code, co\_name) values ("VA", "Holy See (Vatican City State)");

insert into country (co\_code, co\_name) values ("HN", "Honduras");

insert into country (co\_code, co\_name) values ("HK", "Hong Kong");

insert into country (co\_code, co\_name) values ("HU", "Hungary");

insert into country (co\_code, co\_name) values ("IS", "Iceland");

insert into country (co\_code, co\_name) values ("IN", "India");

insert into country (co\_code, co\_name) values ("ID", "Indonesia");

insert into country (co\_code, co\_name) values ("IR", "Iran, Islamic Republic of");

insert into country (co\_code, co\_name) values ("IQ", "Iraq");

insert into country (co\_code, co\_name) values ("IE", "Ireland");

insert into country (co\_code, co\_name) values ("IM", "Isle of Man");

insert into country (co\_code, co\_name) values ("IL", "Israel");

insert into country (co\_code, co\_name) values ("IT", "Italy");

insert into country (co\_code, co\_name) values ("JM", "Jamaica");

insert into country (co\_code, co\_name) values ("JP", "Japan");

insert into country (co\_code, co\_name) values ("JE", "Jersey");

insert into country (co\_code, co\_name) values ("JO", "Jordan");

insert into country (co\_code, co\_name) values ("KZ", "Kazakhstan");

insert into country (co\_code, co\_name) values ("KE", "Kenya");

insert into country (co\_code, co\_name) values ("KI", "Kiribati");

insert into country (co\_code, co\_name) values ("KP", "Democratic People's Republic of Korea");

insert into country (co\_code, co\_name) values ("KR", "Republic of Korea");

insert into country (co\_code, co\_name) values ("KW", "Kuwait");

insert into country (co\_code, co\_name) values ("KG", "Kyrgyzstan");

insert into country (co\_code, co\_name) values ("LA", "Lao People's Democratic Republic");

insert into country (co\_code, co\_name) values ("LV", "Latvia");

insert into country (co\_code, co\_name) values ("LB", "Lebanon");

insert into country (co\_code, co\_name) values ("LS", "Lesotho");

insert into country (co\_code, co\_name) values ("LR", "Liberia");

insert into country (co\_code, co\_name) values ("LY", "Libya");

insert into country (co\_code, co\_name) values ("LI", "Liechtenstein");

insert into country (co\_code, co\_name) values ("LT", "Lithuania");

insert into country (co\_code, co\_name) values ("LU", "Luxembourg");

insert into country (co\_code, co\_name) values ("MO", "Macao");

insert into country (co\_code, co\_name) values ("MK", "Macedonia, the Former Yugoslav Republic of");

insert into country (co\_code, co\_name) values ("MG", "Madagascar");

insert into country (co\_code, co\_name) values ("MW", "Malawi");

insert into country (co\_code, co\_name) values ("MY", "Malaysia");

insert into country (co\_code, co\_name) values ("MV", "Maldives");

insert into country (co\_code, co\_name) values ("ML", "Mali");

insert into country (co\_code, co\_name) values ("MT", "Malta");

insert into country (co\_code, co\_name) values ("MH", "Marshall Islands");

insert into country (co\_code, co\_name) values ("MQ", "Martinique");

insert into country (co\_code, co\_name) values ("MR", "Mauritania");

insert into country (co\_code, co\_name) values ("MU", "Mauritius");

insert into country (co\_code, co\_name) values ("YT", "Mayotte");

insert into country (co\_code, co\_name) values ("MX", "Mexico");

insert into country (co\_code, co\_name) values ("FM", "Micronesia, Federated States of");

insert into country (co\_code, co\_name) values ("MD", "Moldova, Republic of");

insert into country (co\_code, co\_name) values ("MC", "Monaco");

insert into country (co\_code, co\_name) values ("MN", "Mongolia");

insert into country (co\_code, co\_name) values ("ME", "Montenegro");

insert into country (co\_code, co\_name) values ("MS", "Montserrat");

insert into country (co\_code, co\_name) values ("MA", "Morocco");

insert into country (co\_code, co\_name) values ("MZ", "Mozambique");

insert into country (co\_code, co\_name) values ("MM", "Myanmar");

insert into country (co\_code, co\_name) values ("NA", "Namibia");

insert into country (co\_code, co\_name) values ("NR", "Nauru");

insert into country (co\_code, co\_name) values ("NP", "Nepal");

insert into country (co\_code, co\_name) values ("NL", "Netherlands");

insert into country (co\_code, co\_name) values ("NC", "New Caledonia");

insert into country (co\_code, co\_name) values ("NZ", "New Zealand");

insert into country (co\_code, co\_name) values ("NI", "Nicaragua");

insert into country (co\_code, co\_name) values ("NE", "Niger");

insert into country (co\_code, co\_name) values ("NG", "Nigeria");

insert into country (co\_code, co\_name) values ("NU", "Niue");

insert into country (co\_code, co\_name) values ("NF", "Norfolk Island");

insert into country (co\_code, co\_name) values ("MP", "Northern Mariana Islands");

insert into country (co\_code, co\_name) values ("NO", "Norway");

insert into country (co\_code, co\_name) values ("OM", "Oman");

insert into country (co\_code, co\_name) values ("PK", "Pakistan");

insert into country (co\_code, co\_name) values ("PW", "Palau");

insert into country (co\_code, co\_name) values ("PS", "Palestine, State of");

insert into country (co\_code, co\_name) values ("PA", "Panama");

insert into country (co\_code, co\_name) values ("PG", "Papua New Guinea");

insert into country (co\_code, co\_name) values ("PY", "Paraguay");

insert into country (co\_code, co\_name) values ("PE", "Peru");

insert into country (co\_code, co\_name) values ("PH", "Philippines");

insert into country (co\_code, co\_name) values ("PN", "Pitcairn");

insert into country (co\_code, co\_name) values ("PL", "Poland");

insert into country (co\_code, co\_name) values ("PT", "Portugal");

insert into country (co\_code, co\_name) values ("PR", "Puerto Rico");

insert into country (co\_code, co\_name) values ("QA", "Qatar");

insert into country (co\_code, co\_name) values ("RO", "Romania");

insert into country (co\_code, co\_name) values ("RU", "Russian Federation");

insert into country (co\_code, co\_name) values ("RW", "Rwanda");

insert into country (co\_code, co\_name) values ("RE", "Réunion");

insert into country (co\_code, co\_name) values ("BL", "Saint Barthélemy");

insert into country (co\_code, co\_name) values ("SH", "Saint Helena, Ascension and Tristan da Cunha");

insert into country (co\_code, co\_name) values ("KN", "Saint Kitts and Nevis");

insert into country (co\_code, co\_name) values ("LC", "Saint Lucia");

insert into country (co\_code, co\_name) values ("MF", "Saint Martin (French part)");

insert into country (co\_code, co\_name) values ("PM", "Saint Pierre and Miquelon");

insert into country (co\_code, co\_name) values ("VC", "Saint Vincent and the Grenadines");

insert into country (co\_code, co\_name) values ("WS", "Samoa");

insert into country (co\_code, co\_name) values ("SM", "San Marino");

insert into country (co\_code, co\_name) values ("ST", "Sao Tome and Principe");

insert into country (co\_code, co\_name) values ("SA", "Saudi Arabia");

insert into country (co\_code, co\_name) values ("SN", "Senegal");

insert into country (co\_code, co\_name) values ("RS", "Serbia");

insert into country (co\_code, co\_name) values ("SC", "Seychelles");

insert into country (co\_code, co\_name) values ("SL", "Sierra Leone");

insert into country (co\_code, co\_name) values ("SG", "Singapore");

insert into country (co\_code, co\_name) values ("SX", "Sint Maarten (Dutch part)");

insert into country (co\_code, co\_name) values ("SK", "Slovakia");

insert into country (co\_code, co\_name) values ("SI", "Slovenia");

insert into country (co\_code, co\_name) values ("SB", "Solomon Islands");

insert into country (co\_code, co\_name) values ("SO", "Somalia");

insert into country (co\_code, co\_name) values ("ZA", "South Africa");

insert into country (co\_code, co\_name) values ("GS", "South Georgia and the South Sandwich Islands");

insert into country (co\_code, co\_name) values ("SS", "South Sudan");

insert into country (co\_code, co\_name) values ("ES", "Spain");

insert into country (co\_code, co\_name) values ("LK", "Sri Lanka");

insert into country (co\_code, co\_name) values ("SD", "Sudan");

insert into country (co\_code, co\_name) values ("SR", "Suriname");

insert into country (co\_code, co\_name) values ("SJ", "Svalbard and Jan Mayen");

insert into country (co\_code, co\_name) values ("SZ", "Swaziland");

insert into country (co\_code, co\_name) values ("SE", "Sweden");

insert into country (co\_code, co\_name) values ("CH", "Switzerland");

insert into country (co\_code, co\_name) values ("SY", "Syrian Arab Republic");

insert into country (co\_code, co\_name) values ("TW", "Taiwan, Province of China");

insert into country (co\_code, co\_name) values ("TJ", "Tajikistan");

insert into country (co\_code, co\_name) values ("TZ", "Tanzania, United Republic of");

insert into country (co\_code, co\_name) values ("TH", "Thailand");

insert into country (co\_code, co\_name) values ("TL", "Timor-Leste");

insert into country (co\_code, co\_name) values ("TG", "Togo");

insert into country (co\_code, co\_name) values ("TK", "Tokelau");

insert into country (co\_code, co\_name) values ("TO", "Tonga");

insert into country (co\_code, co\_name) values ("TT", "Trinidad and Tobago");

insert into country (co\_code, co\_name) values ("TN", "Tunisia");

insert into country (co\_code, co\_name) values ("TR", "Turkey");

insert into country (co\_code, co\_name) values ("TM", "Turkmenistan");

insert into country (co\_code, co\_name) values ("TC", "Turks and Caicos Islands");

insert into country (co\_code, co\_name) values ("TV", "Tuvalu");

insert into country (co\_code, co\_name) values ("UG", "Uganda");

insert into country (co\_code, co\_name) values ("UA", "Ukraine");

insert into country (co\_code, co\_name) values ("AE", "United Arab Emirates");

insert into country (co\_code, co\_name) values ("GB", "United Kingdom");

insert into country (co\_code, co\_name) values ("US", "United States");

insert into country (co\_code, co\_name) values ("UM", "United States Minor Outlying Islands");

insert into country (co\_code, co\_name) values ("UY", "Uruguay");

insert into country (co\_code, co\_name) values ("UZ", "Uzbekistan");

insert into country (co\_code, co\_name) values ("VU", "Vanuatu");

insert into country (co\_code, co\_name) values ("VE", "Venezuela, Bolivarian Republic of");

insert into country (co\_code, co\_name) values ("VN", "Viet Nam");

insert into country (co\_code, co\_name) values ("VG", "Virgin Islands, British");

insert into country (co\_code, co\_name) values ("VI", "Virgin Islands, U.S.");

insert into country (co\_code, co\_name) values ("WF", "Wallis and Futuna");

insert into country (co\_code, co\_name) values ("EH", "Western Sahara");

insert into country (co\_code, co\_name) values ("YE", "Yemen");

insert into country (co\_code, co\_name) values ("ZM", "Zambia");

insert into country (co\_code, co\_name) values ("ZW", "Zimbabwe");

insert into country (co\_code, co\_name) values ("AX", "Åland Islands");

Refer subsequent hands on exercises to implement the features related to country.

**Application.java**

package com.example.countrymanager;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication

public class Application {

    public static void main(String[] args) {

        SpringApplication.run(Application.class, args);

    }

}

**CountryController.java**

package com.example.countrymanager.controller;

import com.example.countrymanager.model.Country;

import com.example.countrymanager.service.CountryService;

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

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

import java.util.List;

@RestController

@RequestMapping("/countries")

public class CountryController {

    @Autowired

    private CountryService countryService;

    @GetMapping

    public List<Country> getAllCountries() {

        return countryService.getAllCountries();

    }

    @GetMapping("/{id}")

    public Country getCountryById(@PathVariable Long id) {

        return countryService.getCountryById(id);

    }

    @PostMapping

    public Country addCountry(@RequestBody Country country) {

        return countryService.addCountry(country);

    }

    @PutMapping("/{id}")

    public Country updateCountry(@PathVariable Long id, @RequestBody Country country) {

        return countryService.updateCountry(id, country);

    }

    @DeleteMapping("/{id}")

    public void deleteCountry(@PathVariable Long id) {

        countryService.deleteCountry(id);

    }

}

**Country.java**

package com.example.countrymanager.model;

public class Country {

    private Long id;

    private String name;

    private String capital;

    public Country() {}

    public Country(Long id, String name, String capital) {

        this.id = id;

        this.name = name;

        this.capital = capital;

    }

    public Long getId() {

        return id;

    }

    public void setId(Long id) {

        this.id = id;

    }    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public String getCapital() {

        return capital;

    }

    public void setCapital(String capital) {

        this.capital = capital;

    }

}

**CountryService.java**

package com.example.countrymanager.service;

import com.example.countrymanager.model.Country;

import java.util.List;

public interface CountryService {

    List<Country> getAllCountries();

    Country getCountryById(Long id);

    Country addCountry(Country country);

    Country updateCountry(Long id, Country country);

    void deleteCountry(Long id);

}

**CountryServiceImpl.java**

package com.example.countrymanager.service;

import com.example.countrymanager.model.Country;

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.concurrent.atomic.AtomicLong;

@Service

public class CountryServiceImpl implements CountryService {

    private List<Country> countries = new ArrayList<>();

    private AtomicLong idCounter = new AtomicLong();

    @Override

    public List<Country> getAllCountries() {

        return countries;

    }

    @Override

    public Country getCountryById(Long id) {

        return countries.stream().filter(c -> c.getId().equals(id)).findFirst().orElse(null);

    }

    @Override

    public Country addCountry(Country country) {

        country.setId(idCounter.incrementAndGet());

        countries.add(country);

        return country;

    }

    @Override

    public Country updateCountry(Long id, Country country) {

        Country existing = getCountryById(id);

        if (existing != null) {

            existing.setName(country.getName());

            existing.setCapital(country.getCapital());

        }

        return existing;

    }

    @Override

    public void deleteCountry(Long id) {

        countries.removeIf(c -> c.getId().equals(id));

    }

}



**Hands on 6**

**Find a country based on country code** 

* Create new exception class CountryNotFoundException in com.cognizant.spring-learn.service.exception
* Create new method findCountryByCode() in CountryService with @Transactional annotation
* In findCountryByCode() method, perform the following steps:
  + Method signature

@Transactional

public Country findCountryByCode(String countryCode) throws CountryNotFoundException

* Get the country based on findById() built in method

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

* From the result, check if a country is found. If not found, throw CountryNotFoundException

if (!result.isPresent())

* Use get() method to return the country fetched.

Country country = result.get();

* Include new test method in OrmLearnApplication to find a country based on country code and compare the country name to check if it is valid.

    private static void getAllCountriesTest() {

        LOGGER.info("Start");

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

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

        LOGGER.info("End");

    }

* Invoke the above method in main() method and test it.

**NOTE:** SME to explain the importance of @Transactional annotation. Spring takes care of creating the Hibernate session and manages the transactionality when executing the service method.

**FindCountryByCode.java**

package com.example.countryfinder;

import java.util.HashMap;

import java.util.Scanner;

public class FindCountryByCode {

    public static void main(String[] args) {

        HashMap<String, String> countryMap = new HashMap<>();

        countryMap.put("US", "United States");

        countryMap.put("IN", "India");

        countryMap.put("FR", "France");

        countryMap.put("DE", "Germany");

        countryMap.put("JP", "Japan");

        countryMap.put("CN", "China");

        countryMap.put("BR", "Brazil");

        countryMap.put("ZA", "South Africa");

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter country code (e.g., US): ");

        String code = scanner.nextLine().toUpperCase();

        String country = countryMap.get(code);

        if (country != null) {

            System.out.println("Country name: " + country);

        } else {

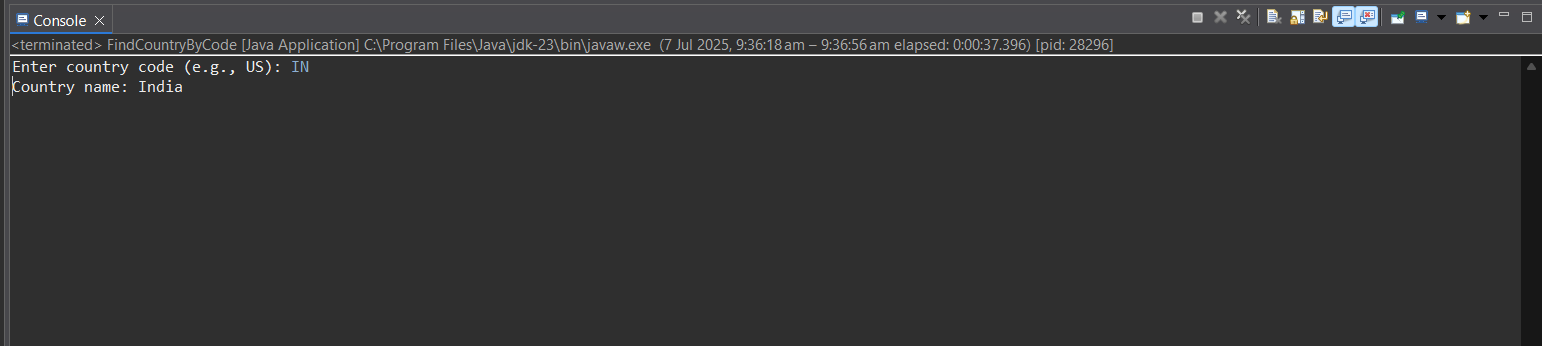
            System.out.println("Country code not found.");

        }

        scanner.close();

    }

}



**Hands on 7**

**Add a new country** 

* Create new method in CountryService.

@Transactional

public void addCountry(Country country)

* Invoke save() method of repository to get the country added.

countryRepository.save(country)

* Include new testAddCountry() method in OrmLearnApplication. Perform steps below:
  + Create new instance of country with a new code and name
  + Call countryService.addCountry() passing the country created in the previous step.
  + Invoke countryService.findCountryByCode() passing the same code used when adding a new country
  + Check in the database if the country is added

**Country.java**

* package com.addnewcountry;
* public class Country {
* private String code;
* private String name;
* public Country(String code, String name) {
* this.code = code;
* this.name = name;
* }
* public String getCode() {
* return code;
* }
* public String getName() {
* return name;
* }
* @Override
* public String toString() {
* return "Country [code=" + code + ", name=" + name + "]";
* }
* }

**CountryManager.java**

package com.addnewcountry;

import java.util.ArrayList;

import java.util.List;

public class CountryManager {

    private List<Country> countries = new ArrayList<>();

    public void addCountry(Country country) {

        countries.add(country);

        System.out.println("Added: " + country);

    }

    public List<Country> getCountries() {

        return countries;

    }

}

**Main.java**

package com.addnewcountry;

public class Main {

    public static void main(String[] args) {

        CountryManager manager = new CountryManager();

        manager.addCountry(new Country("IN", "India"));

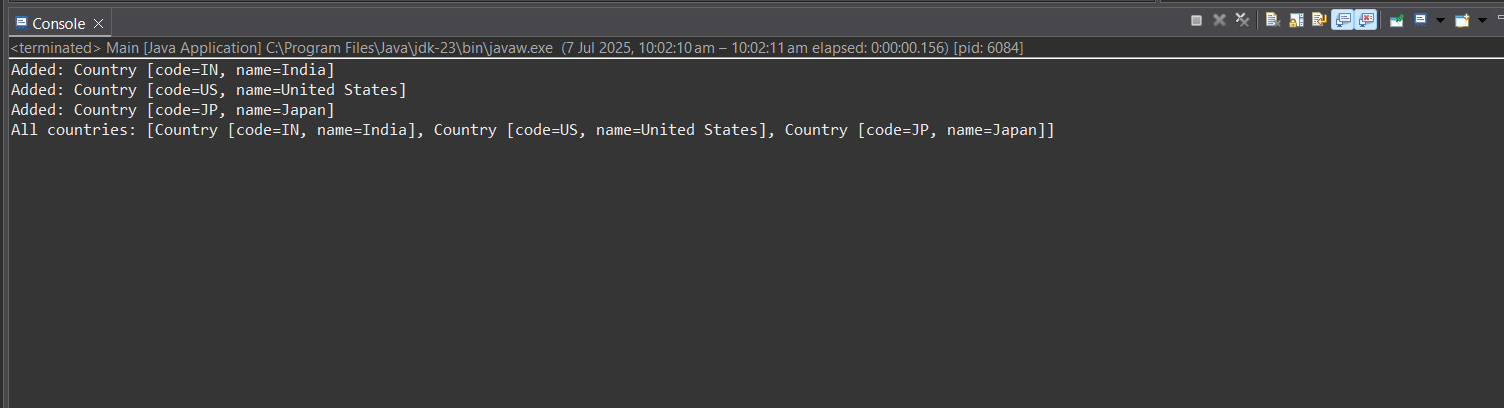
        manager.addCountry(new Country("US", "United States"));

        manager.addCountry(new Country("JP", "Japan"));

        System.out.println("All countries: " + manager.getCountries());

    }

}



**Demonstrate implementation of Query Methods feature of Spring Data JPA**

**Write queries on country table using Query Methods**

import jakarta.persistence.Table;

import jakarta.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;

}

}

package com.cognizant.orm\_learn;

import java.util.List;

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

import org.springframework.stereotype.Repository;

import com.cognizant.orm\_learn.Country;

*@Repository*

public interface CountryRepository extends JpaRepository<Country, String> {

List<Country> findByNameContaining(String keyword); // works if field is 'name'

List<Country> findByNameContainingOrderByNameAsc(String keyword);

List<Country> findByNameStartingWith(String prefix);

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Country;

import com.cognizant.orm\_learn.CountryRepository;

import com.cognizant.orm\_learn.CountryNotFoundException;

import jakarta.transaction.Transactional;

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

import org.springframework.stereotype.Service;

import java.util.Optional;

*@Service*

public class CountryService {

*@Autowired*

private CountryRepository countryRepository;

*@Transactional*

public Country findCountryByCode(String countryCode) throws CountryNotFoundException {

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

if (!result.isPresent()) {

throw new CountryNotFoundException("Country not found for code: " + countryCode);

}

return result.get();

}

*@Transactional*

public void addCountry(Country country) {

countryRepository.save(country);

}

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Country;

import com.cognizant.orm\_learn.CountryService;

import com.cognizant.orm\_learn.CountryNotFoundException;

import java.util.List;

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

import org.springframework.boot.CommandLineRunner;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

*@SpringBootApplication*

public class OrmLearnApplication implements CommandLineRunner {

*@Autowired*

private CountryRepository countryRepository;

public static void main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.class, args);

}

*@Override*

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

// Task 1: Search countries by keyword in name

System.***out***.println("\n[1] Countries containing 'ou':");

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

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

// Task 2: Search and sort countries by keyword in name

System.***out***.println("\n[2] Countries containing 'ou' (sorted ascending):");

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

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

// Task 3: Countries starting with 'Z'

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

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

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

}

}

OUTPUT:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-09T19:32:30.863+05:30 INFO 32660 --- [orm-learn] [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Starting OrmLearnApplication using Java 21.0.7 with PID 32660 (C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes started by admin in C:\Users\admin\Downloads\orm-learn\orm-learn)

2025-07-09T19:32:30.866+05:30 INFO 32660 --- [orm-learn] [ restartedMain] c.c.orm\_learn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-09T19:32:30.945+05:30 INFO 32660 --- [orm-learn] [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-09T19:32:31.909+05:30 INFO 32660 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-07-09T19:32:32.015+05:30 INFO 32660 --- [orm-learn] [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 87 ms. Found 1 JPA repository interface.

2025-07-09T19:32:32.755+05:30 INFO 32660 --- [orm-learn] [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-09T19:32:32.887+05:30 INFO 32660 --- [orm-learn] [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-09T19:32:32.961+05:30 INFO 32660 --- [orm-learn] [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-09T19:32:33.513+05:30 INFO 32660 --- [orm-learn] [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-09T19:32:33.551+05:30 INFO 32660 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-09T19:32:33.808+05:30 INFO 32660 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA

2025-07-09T19:32:33.811+05:30 INFO 32660 --- [orm-learn] [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-09T19:32:33.919+05:30 INFO 32660 --- [orm-learn] [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 2.3.232

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-09T19:32:35.250+05:30 INFO 32660 --- [orm-learn] [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

2025-07-09T19:32:35.254+05:30 INFO 32660 --- [orm-learn] [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-09T19:32:36.448+05:30 INFO 32660 --- [orm-learn] [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-09T19:32:36.486+05:30 INFO 32660 --- [orm-learn] [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Started OrmLearnApplication in 6.304 seconds (process running for 7.058)

[1] Countries containing 'ou':

2025-07-09T19:32:36.793+05:30 DEBUG 32660 --- [orm-learn] [ restartedMain] org.hibernate.SQL :

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

Hibernate:

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

BV - Bouvet Island

DJ - Djibouti

TF - French Southern Territories

GP - Guadeloupe

LU - Luxembourg

ZA - South Africa

GS - South Georgia and the South Sandwich Islands

SS - South Sudan

[2] Countries containing 'ou' (sorted ascending):

2025-07-09T19:32:36.884+05:30 DEBUG 32660 --- [orm-learn] [ restartedMain] org.hibernate.SQL :

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

order by

c1\_0.co\_name

Hibernate:

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

order by

c1\_0.co\_name

BV - Bouvet Island

DJ - Djibouti

TF - French Southern Territories

GP - Guadeloupe

LU - Luxembourg

ZA - South Africa

GS - South Georgia and the South Sandwich Islands

SS - South Sudan

[3] Countries starting with 'Z':

2025-07-09T19:32:36.891+05:30 DEBUG 32660 --- [orm-learn] [ restartedMain] org.hibernate.SQL :

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

Hibernate:

select

c1\_0.co\_code,

c1\_0.co\_name

from

country c1\_0

where

c1\_0.co\_name like ? escape '\'

ZM - Zambia

ZW - Zimbabwe

2025-07-09T19:32:36.909+05:30 INFO 32660 --- [orm-learn] [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'

2025-07-09T19:32:36.916+05:30 WARN 32660 --- [orm-learn] [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy method failed on bean with name 'inMemoryDatabaseShutdownExecutor': org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database is already closed (to disable automatic closing at VM shutdown, add ";DB\_CLOSE\_ON\_EXIT=FALSE" to the db URL) [90121-232]

2025-07-09T19:32:36.917+05:30 INFO 32660 --- [orm-learn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...

2025-07-09T19:32:36.919+05:30 INFO 32660 --- [orm-learn] [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

**Write queries on stock table using Query Methods** 

CREATE TABLE IF NOT EXISTS `stock` (

  `st\_id` INT NOT NULL AUTO\_INCREMENT,

  `st\_code` varchar(10),

  `st\_date` date,

  `st\_open` numeric(10,2),

  `st\_close` numeric(10,2),

  `st\_volume` numeric,

  PRIMARY KEY (`st\_id`)

);

INSERT INTO stock (st\_code, st\_date, st\_open, st\_close, st\_volume) VALUES

('FB', '2019-09-03', 184.00, 182.39, 9779400),

('FB', '2019-09-04', 184.65, 187.14, 11308000),

('GOOGL', '2019-04-22', 1236.67, 1253.76, 954200),

('GOOGL', '2019-04-23', 1256.64, 1270.59, 1593400),

('NFLX', '2018-12-21', 263.83, 246.39, 21397600),

('NFLX', '2018-12-24', 242.00, 233.88, 9547600),

('NFLX', '2018-12-26', 233.92, 253.67, 14402700),

('FB', '2019-01-31', 165.60, 166.69, 77233600);

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

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 Double stOpen;

private Double stClose;

private Long 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 Double getStOpen() { return stOpen; }

public void setStOpen(Double stOpen) { this.stOpen = stOpen; }

public Double getStClose() { return stClose; }

public void setStClose(Double stClose) { this.stClose = stClose; }

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Stock;

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

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

import java.time.LocalDate;

import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

List<Stock> findByStCodeAndStDateBetween(String stCode, LocalDate startDate, LocalDate endDate);

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

List<Stock> findTop3ByOrderByStVolumeDesc();

List<Stock> findTop3ByStCodeOrderByStCloseAsc(String stCode);

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Stock;

import com.cognizant.orm\_learn.StockRepository;

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 java.time.LocalDate;

import java.util.List;

*@SpringBootApplication*

public class OrmLearnApplication {

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

private static StockRepository *stockRepository*;

public static void main(String[] args) {

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

*stockRepository* = context.getBean(StockRepository.class);

*testFacebookSeptember2019*();

*testGooglePriceGreaterThan1250*();

*testTop3Volume*();

*testNetflixLowest3*();

}

private static void testFacebookSeptember2019() {

***LOGGER***.info("Fetching Facebook stocks in Sep 2019");

LocalDate start = LocalDate.*of*(2019, 9, 1);

LocalDate end = LocalDate.*of*(2019, 9, 30);

List<Stock> stocks = *stockRepository*.findByStCodeAndStDateBetween("FB", start, end);

stocks.forEach(stock -> ***LOGGER***.debug(stock.toString()));

}

private static void testGooglePriceGreaterThan1250() {

***LOGGER***.info("Fetching Google stocks > 1250");

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

stocks.forEach(stock -> ***LOGGER***.debug(stock.toString()));

}

private static void testTop3Volume() {

***LOGGER***.info("Fetching top 3 stocks by volume");

List<Stock> stocks = *stockRepository*.findTop3ByOrderByStVolumeDesc();

stocks.forEach(stock -> ***LOGGER***.debug(stock.toString()));

}

private static void testNetflixLowest3() {

***LOGGER***.info("Fetching 3 lowest Netflix stocks");

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

stocks.forEach(stock -> ***LOGGER***.debug(stock.toString()));

}

}

OUTPUT:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-10T18:39:37.990+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Starting OrmLearnApplication using Java 21.0.7 with PID 25184 (C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes started by admin in C:\Users\admin\Downloads\orm-learn\orm-learn)

2025-07-10T18:39:37.993+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-10T18:39:38.067+05:30 INFO 25184 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-10T18:39:38.971+05:30 INFO 25184 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-07-10T18:39:39.087+05:30 INFO 25184 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 103 ms. Found 4 JPA repository interfaces.

2025-07-10T18:39:39.611+05:30 INFO 25184 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-10T18:39:39.944+05:30 INFO 25184 --- [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA

2025-07-10T18:39:39.948+05:30 INFO 25184 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-10T18:39:40.205+05:30 INFO 25184 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-10T18:39:40.321+05:30 INFO 25184 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-10T18:39:40.378+05:30 INFO 25184 --- [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-10T18:39:40.889+05:30 INFO 25184 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-10T18:39:41.026+05:30 INFO 25184 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 2.3.232

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-10T18:39:42.472+05:30 INFO 25184 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

Hibernate: create table country (co\_code varchar(255) not null, co\_name varchar(255), primary key (co\_code))

Hibernate: create table department (id integer generated by default as identity, name varchar(255), primary key (id))

Hibernate: create table employee (id integer generated by default as identity, date\_of\_birth timestamp(6), name varchar(255), permanent boolean, salary float(53), department\_id integer, primary key (id))

Hibernate: alter table if exists stock alter column st\_close set data type float(53)

Hibernate: alter table if exists stock alter column st\_code set data type varchar(255)

Hibernate: alter table if exists stock alter column st\_open set data type float(53)

Hibernate: alter table if exists stock alter column st\_volume set data type bigint

Hibernate: alter table if exists employee add constraint FKbejtwvg9bxus2mffsm3swj3u9 foreign key (department\_id) references department

2025-07-10T18:39:42.586+05:30 INFO 25184 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T18:39:43.773+05:30 INFO 25184 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-10T18:39:43.826+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Started OrmLearnApplication in 6.458 seconds (process running for 7.215)

2025-07-10T18:39:43.836+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Fetching Facebook stocks in Sep 2019

Hibernate: select s1\_0.st\_id,s1\_0.st\_close,s1\_0.st\_code,s1\_0.st\_date,s1\_0.st\_open,s1\_0.st\_volume from stock s1\_0 where s1\_0.st\_code=? and s1\_0.st\_date between ? and ?

2025-07-10T18:39:44.147+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Fetching Google stocks > 1250

Hibernate: select s1\_0.st\_id,s1\_0.st\_close,s1\_0.st\_code,s1\_0.st\_date,s1\_0.st\_open,s1\_0.st\_volume from stock s1\_0 where s1\_0.st\_code=? and s1\_0.st\_close>?

2025-07-10T18:39:44.153+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Fetching top 3 stocks by volume

Hibernate: select s1\_0.st\_id,s1\_0.st\_close,s1\_0.st\_code,s1\_0.st\_date,s1\_0.st\_open,s1\_0.st\_volume from stock s1\_0 order by s1\_0.st\_volume desc fetch first ? rows only

2025-07-10T18:39:44.164+05:30 INFO 25184 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Fetching 3 lowest Netflix stocks

Hibernate: select s1\_0.st\_id,s1\_0.st\_close,s1\_0.st\_code,s1\_0.st\_date,s1\_0.st\_open,s1\_0.st\_volume from stock s1\_0 where s1\_0.st\_code=? order by s1\_0.st\_close fetch first ? rows only

2025-07-10T18:39:44.179+05:30 INFO 25184 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T18:39:44.188+05:30 WARN 25184 --- [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy method failed on bean with name 'inMemoryDatabaseShutdownExecutor': org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database is already closed (to disable automatic closing at VM shutdown, add ";DB\_CLOSE\_ON\_EXIT=FALSE" to the db URL) [90121-232]

2025-07-10T18:39:44.189+05:30 INFO 25184 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...

2025-07-10T18:39:44.193+05:30 INFO 25184 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

**Create payroll tables and bean mapping**

spring.datasource.url=jdbc:h2:mem:testdb

spring.datasource.driverClassName=org.h2.Driver

spring.datasource.username=sa

spring.datasource.password=

spring.jpa.hibernate.ddl-auto=update

spring.jpa.show-sql=true

spring.h2.console.enabled=true

spring.h2.console.path=/h2-console

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.List;

*@Entity*

*@Table*(name = "employee")

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 = "department\_id")

private Department department;

*@ManyToMany*(fetch = *FetchType*.***EAGER***)

*@JoinTable*(name = "employee\_skill",

joinColumns = *@JoinColumn*(name = "employee\_id"),

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

private List<Skill> skills;

*@Override*

public String toString() {

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

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

}

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> getSkills() {

return skills;

}

public void setSkills(List<Skill> skills) {

this.skills = skills;

}

// Getters, Setters, toString

}

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.List;

*@Entity*

*@Table*(name = "department")

public class Department {

*@Id*

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

private int id;

*@Column*(name = "name")

private String name;

*@OneToMany*(mappedBy = "department")

private List<Employee> employees;

*@Override*

public String toString() {

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

}

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> getEmployees() {

return employees;

}

public void setEmployees(List<Employee> employees) {

this.employees = employees;

}

}

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.List;

*@Entity*

*@Table*(name = "skill")

public class Skill {

*@Id*

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

private int id;

*@Column*(name = "name")

private String name;

*@ManyToMany*(mappedBy = "skills")

private List<Employee> employees;

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> getEmployees() {

return employees;

}

public void setEmployees(List<Employee> employees) {

this.employees = employees;

}

*@Override*

public String toString() {

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

}

// Getters, Setters, toString

}

package com.cognizant.orm\_learn;

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

import com.cognizant.orm\_learn.Department;

public interface DepartmentRepository extends JpaRepository<Department, Integer> {}

package com.cognizant.orm\_learn;

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

import com.cognizant.orm\_learn.Skill;

public interface SkillRepository extends JpaRepository<Skill, Integer> {}

package com.cognizant.orm\_learn;

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

import com.cognizant.orm\_learn.Employee;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Department;

import com.cognizant.orm\_learn.Employee;

import com.cognizant.orm\_learn.Skill;

import com.cognizant.orm\_learn.EmployeeRepository;

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 OrmLearnApplication implements CommandLineRunner {

*@Autowired*

private EmployeeRepository employeeRepository;

public static void main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.class, args);

}

*@Override*

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

System.***out***.println("Getting employee details with department and skills...");

Employee employee = employeeRepository.findById(1).orElse(null);

if (employee != null) {

System.***out***.println("Name: " + employee.getName());

System.***out***.println("Salary: " + employee.getSalary());

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

List<Skill> skills = employee.getSkills();

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

for (Skill skill : skills) {

System.***out***.println(" - " + skill.getName());

}

} else {

System.***out***.println("Employee not found!");

}

}

}

OUTPUT:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-10T08:32:48.799+05:30 INFO 33808 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Starting OrmLearnApplication using Java 21.0.7 with PID 33808 (C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes started by admin in C:\Users\admin\Downloads\orm-learn\orm-learn)

2025-07-10T08:32:48.801+05:30 INFO 33808 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-10T08:32:48.876+05:30 INFO 33808 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-10T08:32:49.735+05:30 INFO 33808 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-07-10T08:32:49.844+05:30 INFO 33808 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 90 ms. Found 4 JPA repository interfaces.

2025-07-10T08:32:50.353+05:30 INFO

2025-07-10T08:32:50.353+05:30 INFO restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-10T08:32:50.726+05:30 INFO 33808 --- [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA

2025-07-10T08:32:50.730+05:30 INFO 33808 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-10T08:32:50.975+05:30 INFO 33808 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-10T08:32:51.044+05:30 INFO 33808 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-10T08:32:51.086+05:30 INFO 33808 --- [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-10T08:32:51.465+05:30 INFO 33808 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-10T08:32:51.582+05:30 INFO 33808 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 2.3.232

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-10T08:32:52.863+05:30 INFO 33808 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

Hibernate: create table country (co\_code varchar(255) not null, co\_name varchar(255), primary key (co\_code))

Hibernate: alter table if exists employee alter column date\_of\_birth set data type timestamp(6)

2025-07-10T08:32:52.944+05:30 INFO 33808 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T08:32:54.018+05:30 INFO 33808 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-10T08:32:54.051+05:30 INFO 33808 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Started OrmLearnApplication in 5.91 seconds (process running for 6.738)

Getting employee details with department and skills...

Hibernate: select e1\_0.id,e1\_0.date\_of\_birth,d1\_0.id,d1\_0.name,e1\_0.name,e1\_0.permanent,e1\_0.salary,s1\_0.employee\_id,s1\_1.id,s1\_1.name from employee e1\_0 left join department d1\_0 on d1\_0.id=e1\_0.department\_id left join employee\_skill s1\_0 on e1\_0.id=s1\_0.employee\_id left join skill s1\_1 on s1\_1.id=s1\_0.skill\_id where e1\_0.id=?

Name: Ram

Salary: 50000.0

Department: Engineering

Skills:

- Java

- Python

2025-07-10T08:32:54.259+05:30 INFO 33808 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T08:32:54.268+05:30 WARN 33808 --- [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy method failed on bean with name 'inMemoryDatabaseShutdownExecutor': org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database is already closed (to disable automatic closing at VM shutdown, add ";DB\_CLOSE\_ON\_EXIT=FALSE" to the db URL) [90121-232]

2025-07-10T08:32:54.269+05:30 INFO 33808 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...

2025-07-10T08:32:54.274+05:30 INFO 33808 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

**Demonstrate writing Hibernate Query Language and Native Query**

DROP TABLE IF EXISTS employee\_skill;

DROP TABLE IF EXISTS employee;

DROP TABLE IF EXISTS skill;

DROP TABLE IF EXISTS department;

CREATE TABLE department (

id INT PRIMARY KEY,

name VARCHAR(255)

);

CREATE TABLE skill (

id INT PRIMARY KEY,

name VARCHAR(255)

);

CREATE TABLE employee (

id INT PRIMARY KEY,

name VARCHAR(255),

salary DECIMAL(10, 2),

permanent BOOLEAN,

date\_of\_birth DATE,

department\_id INT,

FOREIGN KEY (department\_id) REFERENCES department(id)

);

CREATE TABLE employee\_skill (

employee\_id INT NOT NULL,

skill\_id INT NOT NULL,

PRIMARY KEY (employee\_id, skill\_id),

CONSTRAINT fk\_emp FOREIGN KEY (employee\_id) REFERENCES employee(id),

CONSTRAINT fk\_skill FOREIGN KEY (skill\_id) REFERENCES skill(id)

);

INSERT INTO department (id, name) VALUES (1, 'Technology'), (2, 'HR');

INSERT INTO skill (id, name) VALUES (1, 'Java'), (2, 'SQL'), (3, 'Spring Boot');

INSERT INTO employee (id, name, salary, permanent, date\_of\_birth, department\_id)

VALUES (1, 'Alice', 60000, true, '1995-02-15', 1),

(2, 'Bob', 50000, false, '1996-07-23', 1),

(3, 'Carol', 70000, true, '1990-03-10', 2);

INSERT INTO employee\_skill (employee\_id, skill\_id) VALUES

(1, 1), (1, 2), (2, 2), (3, 1), (3, 3);

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

import java.util.Date;

import java.util.List;

*@Entity*

*@Table*(name = "employee")

public class Employee {

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

private Integer id;

private String name;

private double salary;

private boolean permanent;

*@Temporal*(*TemporalType*.***DATE***)

private Date dateOfBirth;

*@ManyToOne*(fetch = *FetchType*.***LAZY***)

*@JoinColumn*(name = "department\_id")

private Department department;

*@ManyToMany*(fetch = *FetchType*.***LAZY***)

*@JoinTable*(name = "employee\_skill",

joinColumns = *@JoinColumn*(name = "employee\_id"),

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

private List<Skill> skillList;

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 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 + ", skillList=" + skillList + "]";

}

}

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

*@Entity* *@Table*(name = "skill")

public class Skill {

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

private Integer id;

private String name;

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;

}

*@Override*

public String toString() {

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

}

}

package com.cognizant.orm\_learn;

import jakarta.persistence.\*;

*@Entity* *@Table*(name = "department")

public class Department {

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

private Integer id;

private String name;

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;

}

*@Override*

public String toString() {

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

}

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Department;

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

public interface DepartmentRepository extends JpaRepository<Department, Integer> { }

package com.cognizant.orm\_learn;

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

public interface SkillRepository extends JpaRepository<Skill, Integer> { }

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Employee;

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

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

import java.util.List;

public interface EmployeeRepository extends JpaRepository<Employee, Integer> {

// HQL / JPQL

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

List<Employee> findPermanent();

// HQL with JOIN FETCH to avoid N+1

*@Query*("SELECT DISTINCT e FROM Employee e " +

"JOIN FETCH e.department JOIN FETCH e.skillList " +

"WHERE e.permanent = true")

List<Employee> findPermanentWithDetails();

// HQL aggregate example: average salary per department

*@Query*("SELECT e.department.name, AVG(e.salary) FROM Employee e GROUP BY e.department.name")

List<Object[]> avgSalaryByDepartment();

// Native query: return all permanent employees

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

List<Employee> findPermanentNative();

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Employee;

import com.cognizant.orm\_learn.EmployeeRepository;

import org.springframework.stereotype.Service;

import jakarta.transaction.Transactional;

import java.util.List;

*@Service*

public class EmployeeService {

private final EmployeeRepository repo;

public EmployeeService(EmployeeRepository repo) { this.repo = repo; }

*@Transactional* public List<Employee> getPermanent() {

return repo.findPermanent();

}

*@Transactional* public List<Employee> getPermanentWithDetails() {

return repo.findPermanentWithDetails();

}

*@Transactional* public List<Object[]> getAvgSalaryByDept() {

return repo.avgSalaryByDepartment();

}

*@Transactional* public List<Employee> getPermanentNative() {

return repo.findPermanentNative();

}

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Department;

import com.cognizant.orm\_learn.DepartmentRepository;

import jakarta.transaction.Transactional;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.stereotype.Service;

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

*@Service*

public class DepartmentService {

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

*@Autowired*

private DepartmentRepository departmentRepository;

*@Transactional*

public Department get(int id) {

***LOGGER***.info("Start");

return departmentRepository.findById(id).get();

}

*@Transactional*

public void save(Department department) {

***LOGGER***.info("Start");

departmentRepository.save(department);

***LOGGER***.info("End");

}

}

package com.cognizant.orm\_learn;

import com.cognizant.orm\_learn.Employee;

import com.cognizant.orm\_learn.EmployeeService;

import org.slf4j.\*;

import org.springframework.boot.\*;

import org.springframework.boot.autoconfigure.SpringBootApplication;

import org.springframework.context.ApplicationContext;

import java.util.List;

*@SpringBootApplication*

public class OrmLearnApplication implements CommandLineRunner {

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

private final EmployeeService employeeService;

public OrmLearnApplication(EmployeeService employeeService) {

this.employeeService = employeeService;

}

public static void main(String[] args) {

SpringApplication.*run*(OrmLearnApplication.class, args);

}

*@Override*

public void run(String... args) {

***LOGGER***.info("=== HQL without fetch ===");

employeeService.getPermanent().forEach(e -> ***LOGGER***.debug("Emp: {}", e));

***LOGGER***.info("=== HQL with JOIN FETCH ===");

employeeService.getPermanentWithDetails()

.forEach(e -> ***LOGGER***.debug("Emp: {} Dept: {} Skills: {}", e.getName(), e.getDepartment(), e.getSkillList()));

***LOGGER***.info("=== HQL Aggregate (Avg salary per dept) ===");

employeeService.getAvgSalaryByDept()

.forEach(row -> ***LOGGER***.debug("Dept: {} Avg Salary: {}", row[0], row[1]));

***LOGGER***.info("=== Native SQL Query ===");

employeeService.getPermanentNative().forEach(e -> ***LOGGER***.debug("Emp (Native): {}", e));

}

}

OUTPUT:

. \_\_\_\_ \_ \_\_ \_ \_

/\\ / \_\_\_'\_ \_\_ \_ \_(\_)\_ \_\_ \_\_ \_ \ \ \ \

( ( )\\_\_\_ | '\_ | '\_| | '\_ \/ \_` | \ \ \ \

\\/ \_\_\_)| |\_)| | | | | || (\_| | ) ) ) )

' |\_\_\_\_| .\_\_|\_| |\_|\_| |\_\\_\_, | / / / /

=========|\_|==============|\_\_\_/=/\_/\_/\_/

:: Spring Boot :: (v3.5.3)

2025-07-10T19:54:26.803+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Starting OrmLearnApplication using Java 21.0.7 with PID 23448 (C:\Users\admin\Downloads\orm-learn\orm-learn\target\classes started by admin in C:\Users\admin\Downloads\orm-learn\orm-learn)

2025-07-10T19:54:26.806+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : No active profile set, falling back to 1 default profile: "default"

2025-07-10T19:54:26.886+05:30 INFO 23448 --- [ restartedMain] .e.DevToolsPropertyDefaultsPostProcessor : Devtools property defaults active! Set 'spring.devtools.add-properties' to 'false' to disable

2025-07-10T19:54:27.877+05:30 INFO 23448 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.

2025-07-10T19:54:27.986+05:30 INFO 23448 --- [ restartedMain] .s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 90 ms. Found 5 JPA repository interfaces.

2025-07-10T19:54:28.570+05:30 INFO 23448 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...

2025-07-10T19:54:28.960+05:30 INFO 23448 --- [ restartedMain] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection conn0: url=jdbc:h2:mem:testdb user=SA

2025-07-10T19:54:28.963+05:30 INFO 23448 --- [ restartedMain] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.

2025-07-10T19:54:29.267+05:30 INFO 23448 --- [ restartedMain] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]

2025-07-10T19:54:29.355+05:30 INFO 23448 --- [ restartedMain] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final

2025-07-10T19:54:29.417+05:30 INFO 23448 --- [ restartedMain] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled

2025-07-10T19:54:29.928+05:30 INFO 23448 --- [ restartedMain] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer

2025-07-10T19:54:30.078+05:30 INFO 23448 --- [ restartedMain] org.hibernate.orm.connections.pooling : HHH10001005: Database info:

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']

Database driver: undefined/unknown

Database version: 2.3.232

Autocommit mode: undefined/unknown

Isolation level: undefined/unknown

Minimum pool size: undefined/unknown

Maximum pool size: undefined/unknown

2025-07-10T19:54:31.892+05:30 INFO 23448 --- [ restartedMain] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform integration)

Hibernate: create table country (co\_code varchar(255) not null, co\_name varchar(255), primary key (co\_code))

Hibernate: alter table if exists employee alter column salary set data type float(53)

Hibernate: create table stock (st\_id integer generated by default as identity, st\_close float(53), st\_code varchar(255), st\_date date, st\_open float(53), st\_volume bigint, primary key (st\_id))

2025-07-10T19:54:32.008+05:30 INFO 23448 --- [ restartedMain] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T19:54:32.670+05:30 INFO 23448 --- [ restartedMain] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.

2025-07-10T19:54:34.262+05:30 INFO 23448 --- [ restartedMain] o.s.b.d.a.OptionalLiveReloadServer : LiveReload server is running on port 35729

2025-07-10T19:54:34.290+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : Started OrmLearnApplication in 8.133 seconds (process running for 8.866)

2025-07-10T19:54:34.302+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : === HQL without fetch ===

Hibernate: select e1\_0.id,e1\_0.date\_of\_birth,e1\_0.department\_id,e1\_0.name,e1\_0.permanent,e1\_0.salary from employee e1\_0 where e1\_0.permanent=true

2025-07-10T19:54:34.566+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : === HQL with JOIN FETCH ===

Hibernate: select distinct e1\_0.id,e1\_0.date\_of\_birth,d1\_0.id,d1\_0.name,e1\_0.name,e1\_0.permanent,e1\_0.salary,sl1\_0.employee\_id,sl1\_1.id,sl1\_1.name from employee e1\_0 join department d1\_0 on d1\_0.id=e1\_0.department\_id join employee\_skill sl1\_0 on e1\_0.id=sl1\_0.employee\_id join skill sl1\_1 on sl1\_1.id=sl1\_0.skill\_id where e1\_0.permanent=true

2025-07-10T19:54:34.605+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : === HQL Aggregate (Avg salary per dept) ===

Hibernate: select d1\_0.name,avg(e1\_0.salary) from employee e1\_0 join department d1\_0 on d1\_0.id=e1\_0.department\_id group by d1\_0.name

2025-07-10T19:54:34.629+05:30 INFO 23448 --- [ restartedMain] c.c.orm\_learn.OrmLearnApplication : === Native SQL Query ===

Hibernate: SELECT \* FROM employee WHERE permanent = true

2025-07-10T19:54:34.688+05:30 INFO 23448 --- [ionShutdownHook] j.LocalContainerEntityManagerFactoryBean : Closing JPA EntityManagerFactory for persistence unit 'default'

2025-07-10T19:54:34.695+05:30 WARN 23448 --- [ionShutdownHook] o.s.b.f.support.DisposableBeanAdapter : Invocation of destroy method failed on bean with name 'inMemoryDatabaseShutdownExecutor': org.h2.jdbc.JdbcSQLNonTransientConnectionException: Database is already closed (to disable automatic closing at VM shutdown, add ";DB\_CLOSE\_ON\_EXIT=FALSE" to the db URL) [90121-232]

2025-07-10T19:54:34.696+05:30 INFO 23448 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown initiated...

2025-07-10T19:54:34.699+05:30 INFO 23448 --- [ionShutdownHook] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Shutdown completed.

**Demonstrate implementation of O/R Mapping**

**App.java**

package com.orm.demo;

import org.hibernate.Session;

import org.hibernate.SessionFactory;

import org.hibernate.cfg.Configuration;

public class App {

    public static void main(String[] args) {

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

        Session session = factory.openSession();

        session.beginTransaction();

        Student student = new Student("John Doe", "Computer Science");

        session.save(student);

        session.getTransaction().commit();

        session.close();

        System.out.println("Student saved successfully!");

    }

}

**Student.java**

package com.orm.demo;

public class Student {

    private int id;

    private String name;

    private String department;

    public Student() {}

    public Student(String name, String department) {

        this.name = name;

        this.department = department;

    }

    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 String getDepartment() { return department; }

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

}

**Hibernate.xml**

<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/testdb</property>

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

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

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

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

<mapping resource="student.hbm.xml"/>

</session-factory>

</hibernate-configuration>

**Student.xml**

<hibernate-mapping>

<class name="com.orm.demo.Student" table="student">

<id name="id" column="id" type="int">

<generator class="increment"/>

</id>

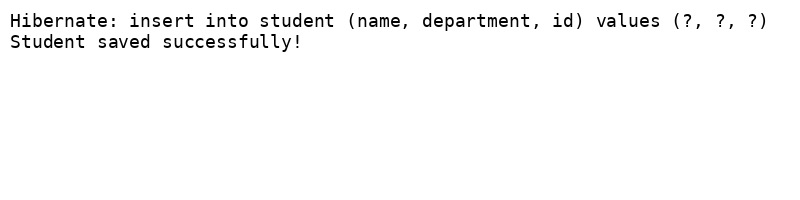
<property name="name" column="name" type="string"/>

<property name="department" column="department" type="string"/>

</class>

</hibernate-mapping>

**Output**



**Demonstrate writing Hibernate Query Language and Native Query**

**Employee.java**

package com.demo.model;

import javax.persistence.\*;

@Entity

@Table(name = "employee")

public class Employee {

    @Id

    @GeneratedValue(strategy = GenerationType.IDENTITY)

    private int id;

    @Column(name = "name")

    private String name;

    @Column(name = "salary")

    private double salary;

    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; }

}

