

HANDSON EXERCISES - WEEK 3

Skill : Spring Data JPA with Spring Boot,Hibernate

Hands On 1:

Spring Data JPA - Quick Example

application.properties

```
spring.application.name=orm-learn
server.port=9091
logging.level.org.springframework=info
logging.level.com.cognizant=debug
logging.level.org.hibernate.SQL=trace
logging.level.org.hibernate.type.descriptor.sql=trace
logging.pattern.console=%d{dd-MM-yy} %d{HH:mm:ss.SSS} %-
20.20thread %5p %-25.25logger{25} %25M %4L %m%n
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=Siddhik@2005
spring.jpa.hibernate.ddl-auto=validate
#spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

Country.java

```
package com.cognizant.orm_learn.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 + "];  
}
```

CountryRepository.java

```
package com.cognizant.orm_learn.repository;
```

```
import com.cognizant.orm_learn.model.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
import org.springframework.stereotype.Repository;
```

```
@Repository  
public interface CountryRepository extends JpaRepository<Country, String> {  
}
```

CountryService.java

```
package com.cognizant.orm_learn.service;
```

```
import com.cognizant.orm_learn.model.Country;  
import com.cognizant.orm_learn.repository.CountryRepository;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import org.springframework.transaction.annotation.Transactional;
```

```
import java.util.List;
```

```
@Service  
public class CountryService {
```

```
@Autowired  
private CountryRepository countryRepository;
```

```
@Transactional  
public List<Country> getAllCountries() {  
    return countryRepository.findAll();  
}  
}
```

OrmLearnApplication.java

```
package com.cognizant.orm_learn;
```

```
import com.cognizant.orm_learn.model.Country;  
import com.cognizant.orm_learn.service.CountryService;  
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.util.List;
```

```
@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);  
        LOGGER.info("Inside main");  
        testGetAllCountries();  
    }
```

```

private static void testGetAllCountries() {
    LOGGER.info("Start");
    List<Country> countries = countryService.getAllCountries();
    LOGGER.debug("countries={}", countries);
    LOGGER.info("End");
}
}

```

OUTPUT:

```

C:\Program Files\Java\jdk-21\bin\java.exe" ...

  ____ _
 / __ \| | | | _ \| | | |
( ( _ \| | | | | | | | | | |
 \___/|_| |_| |_| |_| |_| |_|
=====|_|=====|_|_|_|_|_|

:: Spring Boot ::      (v3.5.3)

06-07-25 22:39:08.824 restartedMain INFO c.c.o.OmLearnApplication logStarting 53 Starting OmLearnApplication using Java 21.0.2 with PID 22616 (C:\Us
06-07-25 22:39:08.827 restartedMain DEBUG c.c.o.OmLearnApplication logStarting 54 Running with Spring Boot v3.5.3, Spring v6.2.8
06-07-25 22:39:08.829 restartedMain INFO c.c.o.OmLearnApplication logStartupProfileInfo 652 No active profile set, falling back to 1 default profile: "default"
06-07-25 22:39:08.890 restartedMain INFO ertyDefaultsPostProcessor logTo 252 Devtools property defaults active! Set 'spring.devtools.add-propertie
06-07-25 22:39:09.953 restartedMain INFO toryConfigurationDelegate registerRepositoriesIn 145 Bootstrapping Spring Data JPA repositories in DEFAULT mode.
06-07-25 22:39:10.027 restartedMain INFO toryConfigurationDelegate registerRepositoriesIn 213 Finished Spring Data repository scanning in 63 ms. Found 1 JPA reposi
06-07-25 22:39:10.480 restartedMain INFO o.h.j.i.util.LogHelper logPersistenceUnitInformation 31 HHM000204: Processing PersistenceUnitInfo [name: default]
06-07-25 22:39:10.541 restartedMain INFO org.hibernate.Version logVersion 44 HHM000412: Hibernate ORM core version 6.6.18.Final
06-07-25 22:39:10.588 restartedMain INFO .i.RegionFactoryInitiator initiateService 50 HHM000026: Second-level cache disabled
06-07-25 22:39:10.991 restartedMain INFO SpringPersistenceUnitInfo addTransformer 87 No LoadTimeWeaver setup; ignoring JPA class transformer
06-07-25 22:39:11.024 restartedMain INFO c.z.h.HikariDataSource getConnection 109 HikariPool-1 - Starting...
06-07-25 22:39:11.559 restartedMain INFO c.z.h.pool.HikariPool checkFailFast 575 HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@fe2a
06-07-25 22:39:11.561 restartedMain INFO c.z.h.HikariDataSource getConnection 122 HikariPool-1 - Start completed.
06-07-25 22:39:11.651 restartedMain WARN o.h.orm.deprecation constructDialect 153 HHM0000025: MySQLDialect does not need to be specified explicitly us
06-07-25 22:39:11.677 restartedMain INFO o.h.o.connections.pooling logConnectionInfo 163 HHM10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0.36
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
06-07-25 22:39:11.702 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(12, org.hibernate.type.descriptor.sql.internal.Capacit
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(-9, org.hibernate.type.descriptor.sql.internal.Capacit
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(-3, org.hibernate.type.descriptor.sql.internal.Capacit
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(4003, org.hibernate.type.descriptor.sql.internal.DdlTy
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(4001, org.hibernate.type.descriptor.sql.internal.DdlTy
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(4002, org.hibernate.type.descriptor.sql.internal.DdlTy
06-07-25 22:39:11.704 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2004, org.hibernate.type.descriptor.sql.internal.Capac
06-07-25 22:39:11.705 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2005, org.hibernate.type.descriptor.sql.internal.Capac
06-07-25 22:39:11.705 restartedMain DEBUG h.t.d.s.s.DdlTypeRegistry addDescriptor 64 addDescriptor(2011, org.hibernate.type.descriptor.sql.internal.Capac
06-07-25 22:39:12.735 restartedMain INFO .p.i.JtaPlatformInitiator initiateService 59 HHM000489: No JTA platform available (set 'hibernate.transaction.jta
06-07-25 22:39:12.836 restartedMain INFO rEntityManagerFactoryBean buildNativeEntityManagerFactory 447 Initialized JPA EntityManagerFactory for persistence unit 'def
06-07-25 22:39:13.330 restartedMain INFO .OptionalLiveReloadServer startServer 59 LiveReload server is running on port 35729
06-07-25 22:39:13.350 restartedMain INFO c.c.o.OmLearnApplication logStarted 59 Started OmLearnApplication in 5.066 seconds (process running for 5.
06-07-25 22:39:13.355 restartedMain INFO c.c.o.OmLearnApplication main 35 Inside main
06-07-25 22:39:13.355 restartedMain INFO c.c.o.OmLearnApplication testGetAllCountries 40 Start
06-07-25 22:39:13.533 restartedMain DEBUG org.hibernate.SQL logStatement 135 select c1_0.co_code,c1_0.co_name from country c1_0
06-07-25 22:39:13.602 restartedMain DEBUG c.c.o.OmLearnApplication testGetAllCountries 42 countries=[Country [code=IN, name=India], Country [code=US, name=Uni
06-07-25 22:39:13.602 restartedMain INFO c.c.o.OmLearnApplication testGetAllCountries 43 End
06-07-25 22:39:13.607 licationShutdownHook INFO rEntityManagerFactoryBean destroy 660 Closing JPA EntityManagerFactory for persistence unit 'default'
06-07-25 22:39:13.610 licationShutdownHook INFO c.z.h.HikariDataSource close 349 HikariPool-1 - Shutdown initiated...
06-07-25 22:39:13.620 licationShutdownHook INFO c.z.h.HikariDataSource close 351 HikariPool-1 - Shutdown completed.

Process finished with exit code 0

```

Hands on 5

Implement services for managing Country

Country.java

```
package com.example.CountryApp.entity;
```

```
import jakarta.persistence.Entity;  
import jakarta.persistence.Id;  
import jakarta.persistence.Table;
```

```
@Entity  
@Table(name = "country")  
public class Country {
```

```
    @Id  
    private String coCode;
```

```
    private String coName;  
    public String getCoCode() {  
        return coCode;  
    }
```

```
    public void setCoCode(String coCode) {  
        this.coCode = coCode;  
    }
```

```
    public String getCoName() {  
        return coName;  
    }
```

```
    public void setCoName(String coName) {  
        this.coName = coName;  
    }
```

```
}
```

CountryRepository.java

```
package com.example.CountryApp.repository;
```

```
import com.example.CountryApp.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;  
import java.util.List;
```

```
public interface CountryRepository extends JpaRepository<Country, String> {  
    List<Country> findByCoNameContainingIgnoreCase(String coName);  
}
```

CountryController.java

```
package com.example.CountryApp.controller;
```

```
import com.example.CountryApp.entity.Country;  
import com.example.CountryApp.service.CountryService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.*;  
import java.util.List;  
import java.util.Optional;
```

```
@RestController  
@RequestMapping("/countries")  
public class CountryController {
```

```
    @Autowired  
    private CountryService service;
```

```
    @GetMapping("/{code}")  
    public Optional<Country> getCountry(@PathVariable String code) {  
        return service.getCountryByCode(code);  
    }
```

```
    @PostMapping  
    public Country addCountry(@RequestBody Country country) {  
        return service.addCountry(country);  
    }
```

```
    @PutMapping  
    public Country updateCountry(@RequestBody Country country) {  
        return service.updateCountry(country);  
    }
```

```
@DeleteMapping("/{code}")
public String deleteCountry(@PathVariable String code) {
    service.deleteCountry(code);
    return "Country deleted successfully";
}
```

```
@GetMapping("/search")
public List<Country> searchCountries(@RequestParam String name) {
    return service.searchCountries(name);
}
}
```

CountryService.java

```
package com.example.CountryApp.service;
```

```
import com.example.CountryApp.entity.Country;
import com.example.CountryApp.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import java.util.List;
import java.util.Optional;
```

```
@Service
public class CountryService {
```

```
    @Autowired
    private CountryRepository countryRepo;
```

```
    public Country addCountry(Country country) {
        return countryRepo.save(country);
    }
```

```
    public Optional<Country> getCountryByCode(String code) {
        return countryRepo.findById(code);
    }
```

```
    public Country updateCountry(Country country) {
        return countryRepo.save(country);
    }
```

```
public void deleteCountry(String code) {  
    countryRepo.deleteById(code);  
}
```

```
public List<Country> searchCountries(String name) {  
    return countryRepo.findByCoNameContainingIgnoreCase(name);  
}
```

CountryAppApplication.java

```
package com.example.CountryApp;
```

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication  
public class CountryAppApplication {
```

```
    public static void main(String[] args) {  
        SpringApplication.run(CountryAppApplication.class, args);  
    }
```

```
}
```


[illegible]

Hands on 4**Difference between JPA, Hibernate and Spring Data JPA****Java Persistence API (JPA)**

- JPA is a Java Specification used for persisting, reading, and managing data from Java objects to relational databases.
- It provides interfaces and annotations, but no implementation.
- It allows developers to map Java classes to database tables using annotations like @Entity, @Table, @Id, etc.
- Hibernate is one of the most commonly used implementations of JPA.

Hibernate

- Hibernate is an Object-Relational Mapping framework that implements the JPA specification.
- It provides all features defined in JPA and adds its own powerful features like caching, lazy loading, criteria queries, etc.
- When used directly we need to manage:
 - Sessions
 - Transactions
 - Error handling
- Hibernate reduces manual SQL and allows developers to work with Java objects.

Spring Data JPA

- Spring Data JPA is not a JPA implementation.
- It is a high-level abstraction built on top of JPA and a JPA provider (like Hibernate).
- Its main purpose is to simplify repository access and reduce boilerplate code.
- Automatically generates common queries like save(), findById(), delete() by extending JpaRepository.
- Integrates seamlessly with Spring's dependency injection and transaction management.

1) JPA with Hibernate**pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
```

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.5.3</version>
  <relativePath/> <!-- lookup parent from repository -->
</parent>
<groupId>com.example</groupId>
<artifactId>employeejpa</artifactId>
<version>0.0.1-SNAPSHOT</version>
<name>employeejpa</name>
<description>Demo project for Spring Boot</description>
<url/>
<licenses>
  <license/>
</licenses>
<developers>
  <developer/>
</developers>
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>21</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>

  <dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>

  <dependency>
```

```
<groupId>com.mysql</groupId>
<artifactId>mysql-connector-j</artifactId>
<scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
</project>
```

Application.properties

```
spring.application.name=employeejpa
spring.datasource.url=jdbc:mysql://localhost:3306/employee_db
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

```
server.port=9094
```

Employee.java

```
package com.example.employeejpa;
```

```
import jakarta.persistence.*;
```

```
@Entity
public class Employee {
```

```
@Id  
@GeneratedValue(strategy = GenerationType.IDENTITY)  
private Integer id;
```

```
private String name;  
private String department;
```

```
public Integer getId() {  
    return id;  
}
```

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

```
public String getName() {  
    return name;  
}
```

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

```
public String getDepartment() {  
    return department;  
}
```

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

EmployeeService.java

```
package com.example.employeejpa;
```

```
import jakarta.persistence.EntityManager;  
import jakarta.persistence.PersistenceContext;  
import jakarta.transaction.Transactional;  
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

```
@Service
public class EmployeeService {

    @PersistenceContext
    private EntityManager entityManager;

    @Transactional
    public void addEmployee(Employee employee) {
        entityManager.persist(employee);
    }

    public Employee getEmployeeById(Integer id) {
        return entityManager.find(Employee.class, id);
    }

    public List<Employee> getAllEmployees() {
        return entityManager.createQuery("FROM Employee",
Employee.class).getResultList();
    }

    @Transactional
    public void deleteEmployee(Integer id) {
        Employee emp = entityManager.find(Employee.class, id);
        if (emp != null) {
            entityManager.remove(emp);
        }
    }
}
```

EmployeeController.java

```
package com.example.employeejpa;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

import java.util.List;

@RestController
@RequestMapping("/employees")
public class EmployeeController {

    @Autowired
    private EmployeeService service;
```

```
@PostMapping
public String add(@RequestBody Employee employee) {
    service.addEmployee(employee);
    return "Employee added.";
}
```

```
@GetMapping("/{id}")
public Employee get(@PathVariable Integer id) {
    return service.getEmployeeById(id);
}
```

```
@GetMapping
public List<Employee> getAll() {
    return service.getAllEmployees();
}
```

```
@DeleteMapping("/{id}")
public String delete(@PathVariable Integer id) {
    service.deleteEmployee(id);
    return "Deleted.";
}
```

EmployeejpaApplication.java

```
package com.example.employeejpa;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

@SpringBootApplication
public class EmployeejpaApplication {

    public static void main(String[] args) {
        SpringApplication.run(EmployeejpaApplication.class, args);
    }
}
```


[illegible]

pom.xml

```
<parent>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-parent</artifactId>
  <version>3.5.3</version>
  <relativePath/>
</parent>
```

```
<groupId>com.example</groupId>  
<artifactId>employeesdjava</artifactId>  
<version>0.0.1-SNAPSHOT</version>  
<name>employeesdjava</name>  
<description>Demo project using Spring Data JPA and MySQL</description>
```



```
<properties>
  <java.version>21</java.version>
</properties>
```

```
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
```

```
    <dependency>
      <groupId>com.mysql</groupId>
      <artifactId>mysql-connector-j</artifactId>
      <scope>runtime</scope>
    </dependency>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-web</artifactId>
    </dependency>
```

```
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-test</artifactId>
      <scope>test</scope>
    </dependency>
  </dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

application.properties

```
spring.application.name=employeesdjpa
spring.datasource.url=jdbc:mysql://localhost:3306/employee_db
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver

spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect

server.port=9095
```

Employee.java

```
package com.example.employeesdjpa;

import jakarta.persistence.*;

@Entity
public class Employee {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Integer id;

    private String name;
    private String department;

    public Integer getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }
}
```

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

```
public String getDepartment() {  
    return department;  
}
```

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

EmployeeRepository.java

```
package com.example.employeesdmp;
```

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

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

EmployeeService.java

```
package com.example.employeesdmp;
```

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;
```

```
import java.util.List;
```

```
@Service  
public class EmployeeService {
```

```
@Autowired  
private EmployeeRepository repo;
```

```
public void addEmployee(Employee employee) {  
    repo.save(employee);  
}
```

```
public Employee getEmployeeById(Integer id) {  
    return repo.findById(id).orElse(null);  
}
```

```
}
```

```
public List<Employee> getAllEmployees() {  
    return repo.findAll();  
}
```

```
public void deleteEmployee(Integer id) {  
    repo.deleteById(id);  
}  
}
```

EmployeeController.java

```
package com.example.employeesdjpa;
```

```
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.*;
```

```
import java.util.List;
```

```
@RestController  
@RequestMapping("/employees")  
public class EmployeeController {
```

```
    @Autowired  
    private EmployeeService service;
```

```
    @PostMapping  
    public String add(@RequestBody Employee employee) {  
        service.addEmployee(employee);  
        return "Employee added.";  
    }
```

```
    @GetMapping("/{id}")  
    public Employee get(@PathVariable Integer id) {  
        return service.getEmployeeById(id);  
    }
```

```
    @GetMapping  
    public List<Employee> getAll() {  
        return service.getAllEmployees();  
    }
```

```
    @DeleteMapping("/{id}")
```

SuperSet ID : 6314238

BAPANAPALLI PREM SAI SIDDHIK

```
public String delete(@PathVariable Integer id) {  
    service.deleteEmployee(id);  
    return "Deleted.";  
}
```

EmployeesJpaApplication.java

```
package com.example.employeesdjava;
```

```
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class EmployeesdjpaApplication {
    public static void main(String[] args) {
        SpringApplication.run(EmployeesdjpaApplication.class, args);
    }
}
```

OUTPUT

[illegible]

Hands on 6

Find a country based on country code

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>countryapp</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>countryapp</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
```

```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

```
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
```

```
</project>
```

applicationContext.xml

```
spring.application.name=countryapp
server.port=9092
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/country_db
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.jpa.hibernate.ddl-auto=validate
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL8Dialect
```

Country.java

```
package com.example.countryapp.entity;
```

```
import jakarta.persistence.Entity;
```

```
import jakarta.persistence.Id;
```

```
import jakarta.persistence.Table;
```

```
@Entity
```

```
@Table(name = "country")
```

```
public class Country {
```

```
    @Id
```

```
    private String coCode;
```

```
    private String coName;
```

```
    public String getCoCode() {
```

```
        return coCode;
```

```
    }
```

```
    public void setCoCode(String coCode) {
```

```
        this.coCode = coCode;
```

```
    }
```

```
    public String getCoName() {
```

```
        return coName;
```

```
    }
```

```
    public void setCoName(String coName) {
```

```
        this.coName = coName;
```

```
    }
```

```
    @Override
```

```
    public String toString() {
```

```
        return "Country{" +
```

```
            "coCode=" + coCode + "\n" +
```

```
            ", coName=" + coName + "\n" +
```

```
            "}";
```

```
    }
```

```
}
```


CountryRepository.java

```
package com.example.countryapp.repository;

import com.example.countryapp.entity.Country;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CountryRepository extends JpaRepository<Country, String> {
}
```

CountryNotFoundException.java

```
package com.example.countryapp.exception;

public class CountryNotFoundException extends Exception {
    public CountryNotFoundException(String message) {
        super(message);
    }
}
```

CountryService.java

```
package com.example.countryapp.service;

import com.example.countryapp.entity.Country;
import com.example.countryapp.exception.CountryNotFoundException;
import com.example.countryapp.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public Country findCountryByCode(String code) throws
CountryNotFoundException {
```

```
        return countryRepository.findById(code)
            .orElseThrow(() -> new CountryNotFoundException("Country not
found with code: " + code));
    }
}
```

CountryController.java

```
package com.example.countryapp.controller;
```

```
import com.example.countryapp.entity.Country;
import com.example.countryapp.exception.CountryNotFoundException;
import com.example.countryapp.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;
```

```
@RestController
@RequestMapping("/countries")
public class CountryController {
```

```
    @Autowired
    private CountryService countryService;
```

```
    @GetMapping("/{code}")
    public Country getCountry(@PathVariable String code) throws
CountryNotFoundException {
        return countryService.findCountryByCode(code);
    }
}
```

CountryAppApplication.java

```
package com.example.countryapp;
```

```
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
@SpringBootApplication
public class CountryAppApplication {
    public static void main(String[] args) {
        SpringApplication.run(CountryAppApplication.class, args);
    }
}
```

```

C:\Program Files\Java\jdk-21\bin\java.exe" ...

  ____
 /  _ \ / ___/ ___/ ___/ ___/ ___/
( (    /  _ \ / ___/ ___/ ___/ ___/
 \  __/  / ___/ ___/ ___/ ___/ ___/
  \___/  \___/ ___/ ___/ ___/ ___/

=====|_|=====|_|_|_|_|_|_|_|_|

:: Spring Boot ::                (v3.5.3)

2025-07-08T23:55:11.964+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:11.969+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:13.016+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:13.088+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:13.909+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:13.930+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:13.930+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:14.099+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:14.101+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:14.483+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:14.631+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:14.714+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:15.186+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:15.222+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:16.010+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:16.012+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:16.107+05:30 WARN 9780 --- [countryapp] [
2025-07-08T23:55:16.108+05:30 WARN 9780 --- [countryapp] [
2025-07-08T23:55:16.130+05:30 INFO 9780 --- [countryapp] [

Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown

2025-07-08T23:55:16.973+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:17.051+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:17.429+05:30 WARN 9780 --- [countryapp] [
2025-07-08T23:55:17.858+05:30 INFO 9780 --- [countryapp] [
2025-07-08T23:55:17.872+05:30 INFO 9780 --- [countryapp] [

main c.e.countryapp.CountryAppApplication : Starting CountryAppApplication using Java 21.0.2 with P
main c.e.countryapp.CountryAppApplication : No active profile set, falling back to 1 default profile
main s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT m
main s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 61 ms. Foun
main o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9092 (http)
main o.apache.catalina.core.StandardService : Starting service [Tomcat]
main o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
main o.e.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
main w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in
main o.hibernate.jpa.internal.util.LogHelper : HHN000204: Processing PersistenceUnitInfo [name: default
main org.hibernate.Version : HHN000412: Hibernate ORM core version 6.6.18.Final
main o.h.c.internal.RegionFactoryInitiator : HHN000026: Second-level cache disabled
main o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
main com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
main com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.Connec
main com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
main org.hibernate.orm.deprecation : HHN9000025: MySQL8Dialect does not need to be specified
main org.hibernate.orm.deprecation : HHN9000026: MySQL8Dialect has been deprecated; use org
main org.hibernate.orm.connections.pooling : HHN10001005: Database info:

main o.h.e.t.j.p.i.JtaPlatformInitiator : HHN000489: No JTA platform available (set 'hibernate.te
main j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence un
main JpaBaseConfigurationJpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefor
main o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9092 (http) with context path '/'
main c.e.countryapp.CountryAppApplication : Started CountryAppApplication in 6.381 seconds (process

```

Hands on 7**Add a new country****application.properties**

```
spring.application.name=countryapp  
server.port=9092
```

```
spring.datasource.url=jdbc:mysql://localhost:3306/country_db  
spring.datasource.username=root  
spring.datasource.password=Siddhik@2005  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
```

```
spring.jpa.hibernate.ddl-auto=update  
spring.jpa.show-sql=true  
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQLDialect
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>  
<project xmlns="http://maven.apache.org/POM/4.0.0"  
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0  
    https://maven.apache.org/xsd/maven-4.0.0.xsd">  
  <modelVersion>4.0.0</modelVersion>  
  <parent>  
    <groupId>org.springframework.boot</groupId>  
    <artifactId>spring-boot-starter-parent</artifactId>  
    <version>3.5.3</version>  
    <relativePath/> <!-- lookup parent from repository -->  
  </parent>  
  <groupId>com.example</groupId>  
  <artifactId>countryapp</artifactId>  
  <version>0.0.1-SNAPSHOT</version>  
  <name>countryapp</name>  
  <description>Demo project for Spring Boot</description>  
  <url/>  
  <licenses>  
    <license/>  
  </licenses>  
  <developers>  
    <developer/>  
  </developers>
```

```
<scm>
  <connection/>
  <developerConnection/>
  <tag/>
  <url/>
</scm>
<properties>
  <java.version>21</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
<build>
  <plugins>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
    </plugin>
  </plugins>
</build>
</project>
```

Country.java

```
package com.example.countryapp.entity;
```

```
import jakarta.persistence.Entity;
```

```
import jakarta.persistence.Id;
```

```
import jakarta.persistence.Table;
```

```
@Entity
```

```
@Table(name = "country")
```

```
public class Country {
```

```
    @Id
```

```
    private String coCode;
```

```
    private String coName;
```

```
    public String getCoCode() {
```

```
        return coCode;
```

```
    }
```

```
    public void setCoCode(String coCode) {
```

```
        this.coCode = coCode;
```

```
    }
```

```
    public String getCoName() {
```

```
        return coName;
```

```
    }
```

```
    public void setCoName(String coName) {
```

```
        this.coName = coName;
```

```
    }
```

CountryRepository.java

```
package com.example.countryapp.repository;
```

```
import com.example.countryapp.entity.Country;
```

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

```
public interface CountryRepository extends JpaRepository<Country, String> {
```

```
}
```

CountryService.java

```
package com.example.countryapp.service;

import com.example.countryapp.entity.Country;
import com.example.countryapp.repository.CountryRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

@Service
public class CountryService {

    @Autowired
    private CountryRepository countryRepository;

    @Transactional
    public void addCountry(Country country) {
        countryRepository.save(country);
    }

    @Transactional
    public Country getCountryByCode(String code) {
        return countryRepository.findById(code)
            .orElseThrow(() -> new RuntimeException("Country not found"));
    }
}
```

CountryController.java

```
package com.example.countryapp.controller;

import com.example.countryapp.entity.Country;
import com.example.countryapp.service.CountryService;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.*;

@RestController
@RequestMapping("/countries")
public class CountryController {

    @Autowired
    private CountryService service;
```

```
@PostMapping
public String addCountry(@RequestBody Country country) {
    service.addCountry(country);
    return "Country added successfully";
}
```

```
@GetMapping("/{code}")
public Country getCountry(@PathVariable String code) {
    return service.getCountryByCode(code);
}
}
```

CountryappApplication

```
package com.example.countryapp;

import com.example.countryapp.entity.Country;
import com.example.countryapp.service.CountryService;
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 CountryappApplication implements CommandLineRunner {

    @Autowired
    private CountryService countryService;

    public static void main(String[] args) {
        SpringApplication.run(CountryappApplication.class, args);
    }

    @Override
    public void run(String... args) {
        // Add new country
        Country c = new Country();
        c.setCoCode("JP");
        c.setCoName("Japan");
        countryService.addCountry(c);

        Country result = countryService.getCountryByCode("JP");
        System.out.println("Fetched: " + result.getCoName());
    }
}
```


applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```
<bean id="bookRepository" class="com.library.repository.BookRepository"/>
```

```
<bean id="bookService" class="com.library.service.BookService">
    <property name="bookRepository" ref="bookRepository"/>
</bean>
```

```
</beans>
```

OUTPUT

```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...

:: Spring Boot ::                (v3.5.3)

2025-07-09T00:15:16.566+05:30 INFO 28324 --- [countryapp] [main] c.e.countryapp.CountryappApplication : Starting CountryappApplication using Java 21.0.2 with
2025-07-09T00:15:16.570+05:30 INFO 28324 --- [countryapp] [main] c.e.countryapp.CountryappApplication : No active profile set, falling back to 1 default profil
2025-07-09T00:15:17.749+05:30 INFO 28324 --- [countryapp] [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT m
2025-07-09T00:15:17.832+05:30 INFO 28324 --- [countryapp] [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 70 ms. Four
2025-07-09T00:15:18.448+05:30 INFO 28324 --- [countryapp] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9092 (http)
2025-07-09T00:15:18.464+05:30 INFO 28324 --- [countryapp] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-09T00:15:18.465+05:30 INFO 28324 --- [countryapp] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-09T00:15:18.528+05:30 INFO 28324 --- [countryapp] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-09T00:15:18.529+05:30 INFO 28324 --- [countryapp] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1
2025-07-09T00:15:18.779+05:30 INFO 28324 --- [countryapp] [main] o.hibernate.jpa.internal.util.LogHelper : HHH0000204: Processing PersistenceUnitInfo [name: default]
2025-07-09T00:15:18.858+05:30 INFO 28324 --- [countryapp] [main] org.hibernate.Version : HHH0000412: Hibernate ORM core version 6.6.18.Final
2025-07-09T00:15:18.904+05:30 INFO 28324 --- [countryapp] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-07-09T00:15:19.440+05:30 INFO 28324 --- [countryapp] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-09T00:15:19.473+05:30 INFO 28324 --- [countryapp] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-09T00:15:20.014+05:30 INFO 28324 --- [countryapp] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.Conne
2025-07-09T00:15:20.018+05:30 INFO 28324 --- [countryapp] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-09T00:15:20.104+05:30 WARN 28324 --- [countryapp] [main] org.hibernate.orm.deprecation : HHH99000025: MySQLDialect does not need to be specified

2025-07-09T00:15:20.131+05:30 INFO 28324 --- [countryapp] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0.36
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-09T00:15:21.390+05:30 INFO 28324 --- [countryapp] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.t
2025-07-09T00:15:21.450+05:30 INFO 28324 --- [countryapp] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence u
2025-07-09T00:15:21.928+05:30 WARN 28324 --- [countryapp] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
2025-07-09T00:15:22.557+05:30 INFO 28324 --- [countryapp] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9092 (http) with context path '/'
2025-07-09T00:15:22.567+05:30 INFO 28324 --- [countryapp] [main] c.e.countryapp.CountryappApplication : Started CountryappApplication in 6.737 seconds (proces
Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
Hibernate: select c1_0.co_code,c1_0.co_name from country c1_0 where c1_0.co_code=?
Fetched: Japan
```

Demonstrate implementation of Query Methods feature of Spring Data JPA**application.properties**

```
spring.application.name=querymethods
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
server.port=9099
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
    https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>querymethods</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>querymethods</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
  </scm>
</project>
```

```
<url/>
</scm>
<properties>
  <java.version>21</java.version>
</properties>
<dependencies>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
  </dependency>
```

```
  <dependency>
    <groupId>com.mysql</groupId>
    <artifactId>mysql-connector-j</artifactId>
    <scope>runtime</scope>
  </dependency>
  <dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
  </dependency>
  <dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
  </dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <annotationProcessorPaths>
          <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </path>
```

```
        </annotationProcessorPaths>
    </configuration>
</plugin>
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <configuration>
        <excludes>
            <exclude>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
            </exclude>
        </excludes>
    </configuration>
</plugin>
</plugins>
</build>
```

```
</project>
```

Country.java

```
package com.example.querymethods.entity;
```

```
import jakarta.persistence.*;
```

```
@Entity
@Table(name = "country")
public class Country {
```

```
    @Id
    @Column(name = "co_code")
    private String code;
```

```
    @Column(name = "co_name")
    private String name;
```

```
    public Country() {}
```

```
    public Country(String code, String name) {
        this.code = code;
        this.name = 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 + "\" +  
        '}'  
    }  
}
```

CountryRepository.java

```
package com.example.querymethods.repository;
```

```
import com.example.querymethods.entity.Country;  
import org.springframework.data.jpa.repository.JpaRepository;
```

```
import java.util.List;
```

```
public interface CountryRepository extends JpaRepository<Country, String> {  
    List<Country> findByNameContainingIgnoreCase(String keyword);  
    List<Country> findByNameContainingIgnoreCaseOrderByNameAsc(String  
keyword);  
    List<Country> findByNameStartingWithIgnoreCase(String prefix);  
}
```

QuerymethodsApplication.java

```
package com.example.querymethods;

import com.example.querymethods.entity.Country;
import com.example.querymethods.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 QuerymethodsApplication implements CommandLineRunner {

    @Autowired
    private CountryRepository countryRepository;

    public static void main(String[] args) {
        SpringApplication.run(QuerymethodsApplication.class, args);
    }

    @Override
    public void run(String... args) {
        System.out.println("\n--- Countries containing 'ou' ---");
        List<Country> list1 =
countryRepository.findByNameContainingIgnoreCase("ou");
        list1.forEach(System.out::println);

        System.out.println("\n--- Countries containing 'ou' sorted ascending ---");
        List<Country> list2 =
countryRepository.findByNameContainingIgnoreCaseOrderByNameAsc("ou");
        list2.forEach(System.out::println);

        System.out.println("\n--- Countries starting with 'Z' ---");
        List<Country> list3 =
countryRepository.findByNameStartingWithIgnoreCase("Z");
        list3.forEach(System.out::println);
    }
}
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
```

```

:: Spring Boot ::                (v3.5.3)

```

```
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown

2025-07-09T01:14:29.651+05:30 INFO 23272 --- [querymethods] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable)
2025-07-09T01:14:29.707+05:30 INFO 23272 --- [querymethods] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-09T01:14:30.150+05:30 WARN 23272 --- [querymethods] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, data will be available to web views during rendering. If you wish to avoid this, please set spring.jpa.open-in-view to false.
2025-07-09T01:14:30.568+05:30 INFO 23272 --- [querymethods] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9099 (http with context path '/')
2025-07-09T01:14:30.577+05:30 INFO 23272 --- [querymethods] [main] c.e.q.QueryMethodsApplication : Started QueryMethodsApplication in 5.767 seconds (process running for 10.007)
```

```
--- Countries containing 'ou' sorted ascending ---
Hibernate:
```

```
--- Countries starting with 'Z' ---
Hibernate:
```

```

        c1_0.co_code,
        c1_0.co_name
    from
        country c1_0
    where
        upper(c1_0.co_name) like upper(?) escape '\\'
Country{code='ZH', name='Zambia'}
Country{code='ZW', name='Zimbabwe'}
2025-07-09T01:24:29.152+05:30 WARN 23272 --- [querymethods] [l-1:housekeeper] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Thread starvation or clock leap detected (housek

```


Demonstrate implementation of O/R Mapping**pom.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>ormmapping</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>ormmapping</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
    <dependency>
```



```
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

```
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <annotationProcessorPaths>
          <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </path>
        </annotationProcessorPaths>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </exclude>
        </excludes>
      </configuration>
    </plugin>
  </plugins>
</build>
```

```
        </excludes>
    </configuration>
</plugin>
</plugins>
</build>
```

```
</project>
```

application.properties

```
spring.application.name=ormmapping
spring.datasource.url=jdbc:mysql://localhost:3306/ormlearn
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
server.port=9097
spring.jpa.hibernate.ddl-auto=update
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Department.java

```
package com.example.ormmapping.entity;
```

```
import jakarta.persistence.*;
import java.util.Set;
```

```
@Entity
public class Department {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private int id;
```

```
    private String name;
```

```
    @OneToMany(mappedBy = "department", fetch = FetchType.EAGER)
    private Set<Employee> employeeList;
```

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

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

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

```
@Override  
public String toString() {  
    return "Department{id=" + id + ", name=" + name + "}";  
}  
}
```

Employee.java

```
package com.example.ormmapping.entity;
```

```
import jakarta.persistence.*;  
import java.util.Date;  
import java.util.Set;
```

```
@Entity  
public class Employee {  
    @Id  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private int id;
```

```
    private String name;  
    private double salary;  
    private boolean permanent;
```

```
@Temporal(TemporalType.DATE)  
private Date dateOfBirth;
```

```
@ManyToOne  
@JoinColumn(name = "em_dp_id")  
private Department department;
```

```
@ManyToMany(fetch = FetchType.EAGER)  
@JoinTable(name = "employee_skill",  
    joinColumns = @JoinColumn(name = "es_em_id"),  
    inverseJoinColumns = @JoinColumn(name = "es_sk_id"))  
private Set<Skill> skillList;
```

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

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

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

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

```
public Date getDateOfBirth() { return dateOfBirth; }  
public void setDateOfBirth(Date dateOfBirth) { this.dateOfBirth =  
dateOfBirth; }
```

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

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

```
@Override  
public String toString() {  
    return "Employee{" +  
        "id=" + id +  
        ", name=" + name + "\" +  
        ", salary=" + salary +  
        ", permanent=" + permanent +  
        ", dateOfBirth=" + dateOfBirth +  
        "'";  
}  
}
```

Skill.java

```
package com.example.ormmapping.entity;
```

```
import jakarta.persistence.*;  
import java.util.Set;
```

```
@Entity  
public class Skill {  
    @Id
```

```
@GeneratedValue(strategy = GenerationType.IDENTITY)
private int id;
```

```
private String name;
```

```
@ManyToMany(mappedBy = "skillList")
private Set<Employee> employeeList;

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

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

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

```
@Override
public String toString() {
    return "Skill{id=" + id + ", name=" + name + "}";
}
}
```

DepartmentRepository.java

```
package com.example.ormmapping.repository;
import com.example.ormmapping.entity.Department;
import org.springframework.data.jpa.repository.JpaRepository;
```

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

EmployeeRepository.java

```
package com.example.ormmapping.repository;
import com.example.ormmapping.entity.Employee;
import org.springframework.data.jpa.repository.JpaRepository;
```

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

SkillRepository.java

```
package com.example.ormmapping.repository;  
import com.example.ormmapping.entity.Skill;  
import org.springframework.data.jpa.repository.JpaRepository;  
  
public interface SkillRepository extends JpaRepository<Skill, Integer> {}
```

OrmmappingApplication.java

```
package com.example.ormmapping;  
  
import com.example.ormmapping.entity.Department;  
import com.example.ormmapping.entity.Employee;  
import com.example.ormmapping.entity.Skill;  
import com.example.ormmapping.repository.DepartmentRepository;  
import com.example.ormmapping.repository.EmployeeRepository;  
import com.example.ormmapping.repository.SkillRepository;  
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.text.SimpleDateFormat;  
import java.util.*;  
  
@SpringBootApplication  
public class OrmmappingApplication implements CommandLineRunner {  
  
    @Autowired  
    private EmployeeRepository employeeRepository;  
  
    @Autowired  
    private DepartmentRepository departmentRepository;  
  
    @Autowired  
    private SkillRepository skillRepository;  
  
    public static void main(String[] args) {  
        SpringApplication.run(OrmmappingApplication.class, args);  
    }  
  
    @Override
```

```
public void run(String... args) throws Exception {  
    Department dept = new Department();  
    dept.setName("Technology");  
    departmentRepository.save(dept);  
  
    Skill skill1 = new Skill();  
    skill1.setName("Java");  
    skillRepository.save(skill1);  
  
    Skill skill2 = new Skill();  
    skill2.setName("SQL");  
    skillRepository.save(skill2);  
  
    Employee emp = new Employee();  
    emp.setName("John Doe");  
    emp.setSalary(75000);  
    emp.setPermanent(true);  
    emp.setDateOfBirth(new SimpleDateFormat("yyyy-MM-dd").parse("1990-  
05-20"));  
    emp.setDepartment(dept);  
  
    Set<Skill> skills = new HashSet<>();  
    skills.add(skill1);  
    skills.add(skill2);  
    emp.setSkillList(skills);  
  
    employeeRepository.save(emp);  
  
    // Fetch employee with department and skills  
    Employee employee = employeeRepository.findById(emp.getId()).get();  
    System.out.println("Employee: " + employee);  
    System.out.println("Department: " + employee.getDepartment());  
    System.out.println("Skills: " + employee.getSkillList());  
}
```

```
"C:\Program Files\Java\jdk-21\bin\java.exe" ...
```

```
:: Spring Boot ::                (v3.5.3)
```

```
Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown

2025-07-09T01:38:50.252+05:30 INFO 25372 --- [ormmapping] [ main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jta.platform' to enable JTA platform)
2025-07-09T01:38:50.476+05:30 INFO 25372 --- [ormmapping] [ main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-09T01:38:50.984+05:30 WARN 25372 --- [ormmapping] [ main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database access may be performed during view rendering.
2025-07-09T01:38:51.533+05:30 INFO 25372 --- [ormmapping] [ main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9097 (http with context path '/')
2025-07-09T01:38:51.543+05:30 INFO 25372 --- [ormmapping] [ main] c.e.OrmMapping.OrmMappingApplication : Started OrmMappingApplication in 6.923 seconds (process running for 10.001 seconds)
```

```

Hibernate:
insert
into
    department
    (name)
values
    (?)
Hibernate:
insert
into
    skill
    (name)
values
    (?)
Hibernate:
insert
into
    skill
    (name)
values
    (?)
Hibernate:

```

```

insert
into
    employee
(date_of_birth, em_dp_id, name, permanent, salary)
values
    (?, ?, ?, ?, ?)

Hibernate:
insert
into
    employee_skill
(es_em_id, es_sk_id)
values
    (?, ?)

Hibernate:
insert
into
    employee_skill
(es_em_id, es_sk_id)
values
    (?, ?)

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,
    s11.0.es_em_id,
    s11.1.id,

```



```

    sl1_1.name
from
  employee e1_0
left join
  department d1_0
    on d1_0.id=e1_0.em_dp_id
left join
  employee_skill sl1_0
    on e1_0.id=sl1_0.es_em_id
left join
  skill sl1_1
    on sl1_1.id=sl1_0.es_sk_id
where
  e1_0.id=?
Hibernate:
select
  e1_0.em_dp_id,
  e1_0.id,
  e1_0.date_of_birth,
  e1_0.name,
  e1_0.permanent,
  e1_0.salary,
  sl1_0.es_em_id,
  sl1_1.id,
  sl1_1.name
from
  employee e1_0 |
left join
  employee_skill sl1_0
    on e1_0.id=sl1_0.es_em_id
left join
```

```

    skill sl1_1
      on sl1_1.id=sl1_0.es_sk_id
where
  e1_0.em_dp_id=?
Employee: Employee{id=1, name='John Doe', salary=75000.0, permanent=true, dateOfBirth=1990-05-20}
Department: Department{id=1, name='Technology'}
Skills: [Skill{id=1, name='Java'}, Skill{id=2, name='SQL'}]
```

Demonstrate writing Hibernate Query Language and Native Query

pom.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.5.3</version>
    <relativePath/> <!-- lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>hqlnative</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>hqlnative</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
    <developerConnection/>
    <tag/>
    <url/>
  </scm>
  <properties>
    <java.version>21</java.version>
  </properties>
  <dependencies>
    <dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-starter-data-jpa</artifactId>
    </dependency>
```

```
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

```
<dependency>
  <groupId>com.mysql</groupId>
  <artifactId>mysql-connector-j</artifactId>
  <scope>runtime</scope>
</dependency>
<dependency>
  <groupId>org.projectlombok</groupId>
  <artifactId>lombok</artifactId>
  <optional>true</optional>
</dependency>
<dependency>
  <groupId>org.springframework.boot</groupId>
  <artifactId>spring-boot-starter-test</artifactId>
  <scope>test</scope>
</dependency>
</dependencies>
```

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <configuration>
        <annotationProcessorPaths>
          <path>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
          </path>
        </annotationProcessorPaths>
      </configuration>
    </plugin>
    <plugin>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-maven-plugin</artifactId>
      <configuration>
        <excludes>
          <exclude>
            <groupId>org.projectlombok</groupId>
            <artifactId>lombok</artifactId>
```

```
        </exclude>
    </excludes>
</configuration>
</plugin>
</plugins>
</build>
```

```
</project>
```

application.properties

```
spring.application.name=hqlnative
spring.datasource.url=jdbc:mysql://localhost:3306/country_db
spring.datasource.username=root
spring.datasource.password=Siddhik@2005
server.port=9095
spring.jpa.hibernate.ddl-auto=none
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Stock.java

```
package com.example.hqlnative.entity;
```

```
import jakarta.persistence.*;
```

```
import java.math.BigDecimal;
import java.util.Date;
```

```
@Entity
@Table(name = "stock")
public class Stock {
```

```
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "st_id")
    private int id;
```

```
    @Column(name = "st_code")
    private String code;
```

```
    @Column(name = "st_date")
    @Temporal(TemporalType.DATE)
    private Date date;
```

```
@Column(name = "st_open")
private BigDecimal open;
```

```
@Column(name = "st_close")
private BigDecimal close;
```

```
@Column(name = "st_volume")
private long volume;
```

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

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

```
public Date getDate() { return date; }
public void setDate(Date date) { this.date = date; }
```

```
public BigDecimal getOpen() { return open; }
public void setOpen(BigDecimal open) { this.open = open; }
```

```
public BigDecimal getClose() { return close; }
public void setClose(BigDecimal close) { this.close = close; }
```

```
public long getVolume() { return volume; }
public void setVolume(long volume) { this.volume = volume; }
```

```
@Override
public String toString() {
    return "Stock{" +
        "id=" + id +
        ", code=" + code + '\n' +
        ", date=" + date +
        ", open=" + open +
        ", close=" + close +
        ", volume=" + volume +
        '}';
}
```

StockRepository.java

```
package com.example.hqlnative.repository;

import com.example.hqlnative.entity.Stock;
import org.springframework.data.jpa.repository.JpaRepository;
import org.springframework.data.jpa.repository.Query;
import org.springframework.data.repository.query.Param;

import java.util.Date;
import java.util.List;

public interface StockRepository extends JpaRepository<Stock, Integer> {

    @Query("SELECT s FROM Stock s WHERE s.code = 'FB' AND s.date BETWEEN :startDate AND :endDate")
    List<Stock> getFacebookStocksInSep2019(@Param("startDate") Date start, @Param("endDate") Date end);

    @Query("SELECT s FROM Stock s WHERE s.code = 'GOOGL' AND s.close > 1250")
    List<Stock> getGoogleStocksAbove1250();

    @Query(value = "SELECT * FROM stock ORDER BY st_volume DESC LIMIT 3", nativeQuery = true)
    List<Stock> getTop3VolumeStocks();

    @Query(value = "SELECT * FROM stock WHERE st_code = 'NFLX' ORDER BY st_close ASC LIMIT 3", nativeQuery = true)
    List<Stock> getLowestNetflixStocks();
}
```

HqlNativeApplication.java

```
package com.example.hqlnative;

import com.example.hqlnative.entity.Stock;
import com.example.hqlnative.repository.StockRepository;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
```

```
import org.springframework.boot.CommandLineRunner;
```

```
import java.text.SimpleDateFormat;
```

```
import java.util.Date;
```

```
import java.util.List;
```

```
@SpringBootApplication
```

```
public class HqlNativeApplication implements CommandLineRunner {
```

```
    @Autowired
```

```
    private StockRepository stockRepository;
```

```
    public static void main(String[] args) {
```

```
        SpringApplication.run(HqlNativeApplication.class, args);
```

```
    }
```

```
    @Override
```

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

```
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
```

```
        Date start = sdf.parse("2019-09-01");
```

```
        Date end = sdf.parse("2019-09-30");
```

```
        System.out.println("\n--- Facebook Stocks in Sep 2019 (HQL) ---");
```

```
        List<Stock> fbStocks =
```

```
        stockRepository.getFacebookStocksInSep2019(start, end);
```

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

```
        System.out.println("\n--- Google Stocks > 1250 (HQL) ---");
```

```
        List<Stock> googleStocks = stockRepository.getGoogleStocksAbove1250();
```

```
        googleStocks.forEach(System.out::println);
```

```
        System.out.println("\n--- Top 3 Volume Stocks (Native SQL) ---");
```

```
        List<Stock> topVolumes = stockRepository.getTop3VolumeStocks();
```

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

```
        System.out.println("\n--- Lowest Netflix Stocks (Native SQL) ---");
```

```
        List<Stock> lowNetflix = stockRepository.getLowestNetflixStocks();
```

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

```
    }
```

```
}
```

OUTPUT

```
C:\Program Files\Java\jdk-21\bin\java.exe" ...

  ____
 / \  / ____ \ ____ \ ____ \ ____ \
(  ) / ____ \ ____ \ ____ \ ____ \
 \ /  / ____ \ ____ \ ____ \ ____ \
  '   / ____ \ ____ \ ____ \ ____ \
=====|_|=====|_|_/|_|_/|_|_/

:: Spring Boot ::                (v3.5.3)

2025-07-09T02:05:05.070+05:30 INFO 29952 --- [hqlnative] [main] c.e.hqlnative.HqlNativeApplication : Starting HqlNativeApplication using Java 21.0.2 with PID 29952 (C:\
2025-07-09T02:05:05.077+05:30 INFO 29952 --- [hqlnative] [main] c.e.hqlnative.HqlNativeApplication : No active profile set, falling back to 1 default profile: "default"
2025-07-09T02:05:06.025+05:30 INFO 29952 --- [hqlnative] [main] s.d.r.c.RepositoryConfigurationDelegate : Bootstrapping Spring Data JPA repositories in DEFAULT mode.
2025-07-09T02:05:06.112+05:30 INFO 29952 --- [hqlnative] [main] s.d.r.c.RepositoryConfigurationDelegate : Finished Spring Data repository scanning in 72 ms. Found 1 JPA repo
2025-07-09T02:05:06.687+05:30 INFO 29952 --- [hqlnative] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 9095 (http)
2025-07-09T02:05:06.705+05:30 INFO 29952 --- [hqlnative] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-07-09T02:05:06.795+05:30 INFO 29952 --- [hqlnative] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.42]
2025-07-09T02:05:06.791+05:30 INFO 29952 --- [hqlnative] [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2025-07-09T02:05:06.795+05:30 INFO 29952 --- [hqlnative] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 1606 ms
2025-07-09T02:05:07.076+05:30 INFO 29952 --- [hqlnative] [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2025-07-09T02:05:07.161+05:30 INFO 29952 --- [hqlnative] [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 6.6.18.Final
2025-07-09T02:05:07.212+05:30 INFO 29952 --- [hqlnative] [main] o.h.c.internal.RegionFactoryInitiator : HHH000026: Second-level cache disabled
2025-07-09T02:05:07.447+05:30 INFO 29952 --- [hqlnative] [main] o.s.o.j.p.SpringPersistenceUnitInfo : No LoadTimeWeaver setup: ignoring JPA class transformer
2025-07-09T02:05:07.495+05:30 INFO 29952 --- [hqlnative] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2025-07-09T02:05:08.354+05:30 INFO 29952 --- [hqlnative] [main] com.zaxxer.hikari.pool.HikariPool : HikariPool-1 - Added connection com.mysql.cj.jdbc.ConnectionImpl@41
2025-07-09T02:05:08.370+05:30 INFO 29952 --- [hqlnative] [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2025-07-09T02:05:08.605+05:30 INFO 29952 --- [hqlnative] [main] org.hibernate.orm.connections.pooling : HHH10001005: Database info:
Database JDBC URL [Connecting through datasource 'HikariDataSource (HikariPool-1)']
Database driver: undefined/unknown
Database version: 8.0.36

Autocommit mode: undefined/unknown
Isolation level: undefined/unknown
Minimum pool size: undefined/unknown
Maximum pool size: undefined/unknown
2025-07-09T02:05:09.583+05:30 INFO 29952 --- [hqlnative] [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000489: No JTA platform available (set 'hibernate.transaction.jt
2025-07-09T02:05:09.587+05:30 INFO 29952 --- [hqlnative] [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2025-07-09T02:05:10.341+05:30 INFO 29952 --- [hqlnative] [main] o.s.d.j.r.query.QueryEnhancerFactory : Hibernate is in classpath; If applicable, HQL parser will be used.
2025-07-09T02:05:11.083+05:30 WARN 29952 --- [hqlnative] [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database
2025-07-09T02:05:11.678+05:30 INFO 29952 --- [hqlnative] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 9095 (http) with context path '/'
2025-07-09T02:05:11.697+05:30 INFO 29952 --- [hqlnative] [main] c.e.hqlnative.HqlNativeApplication : Started HqlNativeApplication in 7.458 seconds (process running for

--- Facebook Stocks in Sep 2019 (HQL) ---
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='FB'
  and s1_0.st_date between ? and ?
Stock{id=1, code='FB', date=2019-09-03, open=184.00, close=182.39, volume=9779400}
Stock{id=2, code='FB', date=2019-09-04, open=184.65, close=187.14, volume=11388000}
Stock{id=3, code='FB', date=2019-09-05, open=188.53, close=190.90, volume=13876700}
Stock{id=4, code='FB', date=2019-09-06, open=190.21, close=187.49, volume=15226800}
Stock{id=5, code='FB', date=2019-09-09, open=187.73, close=188.76, volume=14722400}
Stock{id=6, code='FB', date=2019-09-10, open=187.44, close=186.17, volume=15455900}

Stock{id=7, code='FB', date=2019-09-11, open=186.46, close=188.49, volume=11761700}
Stock{id=8, code='FB', date=2019-09-12, open=189.86, close=187.47, volume=11419800}
Stock{id=9, code='FB', date=2019-09-13, open=187.33, close=187.19, volume=11441100}
Stock{id=10, code='FB', date=2019-09-16, open=186.93, close=186.22, volume=8444800}
Stock{id=11, code='FB', date=2019-09-17, open=186.66, close=188.08, volume=9671100}

--- Google Stocks > 1250 (HQL) ---
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='GOOGL'
  and s1_0.st_close>1250
Stock{id=12, code='GOOGL', date=2019-04-22, open=1236.67, close=1253.76, volume=954200}
Stock{id=13, code='GOOGL', date=2019-04-23, open=1256.64, close=1270.59, volume=1593400}
Stock{id=14, code='GOOGL', date=2019-04-24, open=1270.59, close=1260.05, volume=1169800}
Stock{id=15, code='GOOGL', date=2019-04-25, open=1270.30, close=1267.34, volume=1567200}
Stock{id=16, code='GOOGL', date=2019-04-26, open=1273.38, close=1277.42, volume=1361400}
Stock{id=17, code='GOOGL', date=2019-04-29, open=1280.51, close=1296.20, volume=3618400}
Stock{id=18, code='GOOGL', date=2019-10-17, open=1251.40, close=1252.80, volume=1047900}

--- Top 3 Volume Stocks (Native SQL) ---
Hibernate:
SELECT
```



```

    *
FROM
    stock
ORDER BY
    st_volume DESC
LIMIT
    3
Stock{id=20, code='NFLX', date=2018-12-21, open=263.83, close=246.39, volume=21397600}
Stock{id=6, code='FB', date=2019-09-10, open=187.44, close=186.17, volume=15455900}
Stock{id=4, code='FB', date=2019-09-06, open=190.21, close=187.49, volume=15226800}

--- Lowest Netflix Stocks (Native SQL) ---
Hibernate:
SELECT
    *
FROM
    stock
WHERE
    st_code = 'NFLX'
ORDER BY
    st_close ASC
LIMIT
    3
Stock{id=19, code='NFLX', date=2018-12-24, open=242.00, close=233.88, volume=9547600}
Stock{id=20, code='NFLX', date=2018-12-21, open=263.83, close=246.39, volume=21397600}
Stock{id=21, code='NFLX', date=2018-12-26, open=233.92, close=253.67, volume=14402700}
```