**SUPERSET ID: 6416829**

**1.Spring Data JPA - Quick Example**

SpringDataJpaExample.java:

package com.example.springdatajpa;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import org.springframework.stereotype.Repository;

import jakarta.persistence.\*;

import java.util.List;

@SpringBootApplication

@RestController

@RequestMapping("/users")

public class SpringDataJpaExample {

private final UserRepository userRepository;

public SpringDataJpaExample(UserRepository userRepository) {

this.userRepository = userRepository;

}

public static void main(String[] args) {

SpringApplication.run(SpringDataJpaExample.class, args);

}

@GetMapping

public List<User> getAllUsers() {

return userRepository.findAll();

}

@PostMapping

public User createUser(@RequestBody User user) {

return userRepository.save(user);

}

@DeleteMapping("/{id}")

public void deleteUser(@PathVariable Long id) {

userRepository.deleteById(id);

}

@Entity

public static class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String email;

public User() {}

public User(String name, String email) {

this.name = name;

this.email = email;

}

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 getEmail() { return email; }

public void setEmail(String email) { this.email = email; }

}

@Repository

interface UserRepository extends JpaRepository<User, Long> {}

}

**Body (JSON):**

{

"name": "sornambika",

"email": "sorna@example.com"

}

Output:

[

{

"id": 1,

"name": "sornambika",

"email": "sorna@example.com"

}

]

**2.** **Difference between JPA, Hibernate and Spring Data JPA**

1. JPA (Java Persistence API)

| **Feature** | **Description** |
| --- | --- |
| **Type** | Specification (not a tool or implementation) |
| **Provider?** | No (just an API specification) |
| **Purpose** | Defines a standard for ORM (Object Relational Mapping) in Java |
| **Key Uses** | Annotations like @Entity, @Id, @OneToMany, etc. |
| **Example Use** | You use JPA to write entity classes and query logic using JPQL |
| **Need an Implementation?** | Yes, like Hibernate or EclipseLink |

2.Hibernate

| **Feature** | **Description** |
| --- | --- |
| **Type** | JPA Implementation (ORM Tool) |
| **Provider?** | Yes (most widely used implementation of JPA) |
| **Purpose** | Implements JPA; provides extra features beyond JPA |
| **Key Uses** | SessionFactory, HQL, Criteria API, lazy/eager loading, caching |
| **Extra Features** | Dirty checking, batch processing, second-level caching, native SQL support |

3. Spring Data JPA

| **Feature** | **Description** |
| --- | --- |
| **Type** | Abstraction layer over JPA (Spring project) |
| **Provider?** | No, uses Hibernate under the hood (or any JPA provider) |
| **Purpose** | Simplifies repository and data access layer using interfaces |
| **Key Uses** | JpaRepository, method queries like findByEmail, paging, sorting |
| **Boilerplate Code?** | Almost none — no need to write implementations for most CRUD methods |

**ADDITIONAL HANDS- ON**

**1.Implement services for managing Country**

CountryApp.java

package com.example.countryapp;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import jakarta.persistence.\*;

import java.util.List;

import java.util.Optional;

@SpringBootApplication

@RestController

@RequestMapping("/countries")

public class CountryApp {

private final CountryService countryService;

public CountryApp(CountryService countryService) {

this.countryService = countryService;

}

public static void main(String[] args) {

SpringApplication.run(CountryApp.class, args);

}

@GetMapping

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

@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);

}

@Entity

public static class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String capital;

public Country() {}

public Country(String name, String capital) {

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; }

}

@Repository

interface CountryRepository extends JpaRepository<Country, Long> {}

@Service

public static class CountryService {

private final CountryRepository countryRepository;

public CountryService(CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public Country addCountry(Country country) {

return countryRepository.save(country);

}

public Country updateCountry(Long id, Country updatedCountry) {

Optional<Country> optionalCountry = countryRepository.findById(id);

if (optionalCountry.isPresent()) {

Country existing = optionalCountry.get();

existing.setName(updatedCountry.getName());

existing.setCapital(updatedCountry.getCapital());

return countryRepository.save(existing);

}

return null;

}

public void deleteCountry(Long id) {

countryRepository.deleteById(id);

}

}

}

**POST** /countries

{

"name": "India",

"capital": "New Delhi"

}

**GET** /countries

[

{

"id": 1,

"name": "India",

"capital": "New Delhi"

}

]

**PUT** /countries/1

{

"name": "Bharat",

"capital": "New Delhi"

}

**DELETE /countries/1**

DELETE <http://localhost:8080/countries/1>

. **GET /countries** — After deletion

GET <http://localhost:8080/countries>

**2.** **Find a country based on country code**

CountryApp.java

package com.example.countryapp;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import jakarta.persistence.\*;

import java.util.List;

import java.util.Optional;

@SpringBootApplication

@RestController

@RequestMapping("/countries")

public class CountryApp {

private final CountryService countryService;

public CountryApp(CountryService countryService) {

this.countryService = countryService;

}

public static void main(String[] args) {

SpringApplication.run(CountryApp.class, args);

}

@GetMapping

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

@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);

}

@GetMapping("/code/{code}")

public Country getCountryByCode(@PathVariable String code) {

return countryService.getCountryByCode(code);

}

@Entity

public static class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String capital;

private String code;

public Country() {}

public Country(String name, String capital, String code) {

this.name = name;

this.capital = capital;

this.code = code;

}

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; }

public String getCode() { return code; }

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

}

@Repository

interface CountryRepository extends JpaRepository<Country, Long> {

Country findByCode(String code);

}

@Service

public static class CountryService {

private final CountryRepository countryRepository;

public CountryService(CountryRepository countryRepository) {

this.countryRepository = countryRepository;

}

public List<Country> getAllCountries() {

return countryRepository.findAll();

}

public Country addCountry(Country country) {

return countryRepository.save(country);

}

public Country updateCountry(Long id, Country updatedCountry) {

Optional<Country> optionalCountry = countryRepository.findById(id);

if (optionalCountry.isPresent()) {

Country existing = optionalCountry.get();

existing.setName(updatedCountry.getName());

existing.setCapital(updatedCountry.getCapital());

existing.setCode(updatedCountry.getCode());

return countryRepository.save(existing);

}

return null;

}

public void deleteCountry(Long id) {

countryRepository.deleteById(id);

}

public Country getCountryByCode(String code) {

return countryRepository.findByCode(code);

}

}

}

**POST /countries – Add Country**

**Request:**

{

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

**Response:**

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

**GET /countries/code/IN – Find by Code**

**Response:**

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

**Response:**

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

### GET /countries – All Countries

**Response:**

[

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

]

**3.Add a new Country**

1. **POST Request to /countries**

Request URL:

<http://localhost:8080/countries>

HTTP Method:

POST

Headers:

Content-Type: application/json

Request Body:

{

"name": "France",

"capital": "Paris",

"code": "FR"

}

**2.Verify by GET /countries**

GET Request:

<http://localhost:8080/countries>

Response:

[

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

},

{

"id": 2,

"name": "France",

"capital": "Paris",

"code": "FR"

}

]

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

CountryApp.java

package com.example.countryapp;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import jakarta.persistence.\*;

import java.util.List;

import java.util.Optional;

@SpringBootApplication

@RestController

@RequestMapping("/countries")

public class CountryApp {

private final CountryService countryService;

public CountryApp(CountryService countryService) {

this.countryService = countryService;

}

public static void main(String[] args) {

SpringApplication.run(CountryApp.class, args);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@GetMapping("/name/{name}")

public Country getByName(@PathVariable String name) {

return countryService.getByName(name);

}

@GetMapping("/capital/contains/{text}")

public List<Country> getByCapitalContaining(@PathVariable String text) {

return countryService.getByCapitalContaining(text);

}

@GetMapping("/code/ignorecase/{code}")

public Country getByCodeIgnoreCase(@PathVariable String code) {

return countryService.getByCodeIgnoreCase(code);

}

@GetMapping

public List<Country> getAll() {

return countryService.getAllCountries();

}

@Entity

public static class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String capital;

private String code;

public Country() {}

public Country(String name, String capital, String code) {

this.name = name;

this.capital = capital;

this.code = code;

}

public Long getId() { return id; }

public String getName() { return name; }

public String getCapital() { return capital; }

public String getCode() { return code; }

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

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

public void setCapital(String capital) { this.capital = capital; }

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

}

@Repository

interface CountryRepository extends JpaRepository<Country, Long> {

Country findByName(String name);

List<Country> findByCapitalContaining(String keyword);

Country findByCodeIgnoreCase(String code);

}

@Service

public static class CountryService {

private final CountryRepository repo;

public CountryService(CountryRepository repo) {

this.repo = repo;

}

public Country addCountry(Country country) {

return repo.save(country);

}

public List<Country> getAllCountries() {

return repo.findAll();

}

public Country getByName(String name) {

return repo.findByName(name);

}

public List<Country> getByCapitalContaining(String keyword) {

return repo.findByCapitalContaining(keyword);

}

public Country getByCodeIgnoreCase(String code) {

return repo.findByCodeIgnoreCase(code);

}

}

}

GET /countries/name/India

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

GET /countries/capital/contains/Par

[

{

"id": 2,

"name": "France",

"capital": "Paris",

"code": "FR"

}

]

GET /countries/code/ignorecase/in

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

**5. Demonstrate implementation of O/R Mapping**

What is O/R Mapping?

**Object-Relational Mapping (O/R Mapping)** is the process of linking Java objects (classes) with database tables using annotations like @Entity, @Id, @OneToMany, etc.

Mapping Country and State (One-to-Many Relationship):

ORMappingApp.java:

package com.example.ormapping;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

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

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import jakarta.persistence.\*;

import java.util.\*;

@SpringBootApplication

@RestController

@RequestMapping("/countries")

public class ORMappingApp {

private final CountryService countryService;

public ORMappingApp(CountryService countryService) {

this.countryService = countryService;

}

public static void main(String[] args) {

SpringApplication.run(ORMappingApp.class, args);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.saveCountry(country);

}

@GetMapping

public List<Country> getAllCountries() {

return countryService.getAllCountries();

}

@Entity

public static class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String code;

@OneToMany(mappedBy = "country", cascade = CascadeType.ALL)

private List<State> states = new ArrayList<>();

public Country() {}

public Country(String name, String code, List<State> states) {

this.name = name;

this.code = code;

this.states = states;

for (State s : states) {

s.setCountry(this);

}

}

public Long getId() { return id; }

public String getName() { return name; }

public String getCode() { return code; }

public List<State> getStates() { return states; }

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

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

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

public void setStates(List<State> states) {

this.states = states;

for (State s : states) {

s.setCountry(this);

}

}

}

@Entity

public static class State {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

@ManyToOne

@JoinColumn(name = "country\_id")

private Country country;

public State() {}

public State(String name) {

this.name = name;

}

public Long getId() { return id; }

public String getName() { return name; }

public Country getCountry() { return country; }

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

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

public void setCountry(Country country) { this.country = country; }

}

@Repository

interface CountryRepository extends JpaRepository<Country, Long> {}

@Repository

interface StateRepository extends JpaRepository<State, Long> {}

@Service

public static class CountryService {

private final CountryRepository countryRepo;

public CountryService(CountryRepository countryRepo) {

this.countryRepo = countryRepo;

}

public Country saveCountry(Country country) {

return countryRepo.save(country);

}

public List<Country> getAllCountries() {

return countryRepo.findAll();

}

}

}

POST /countries:

{

"name": "India",

"code": "IN",

"states": [

{ "name": "Tamil Nadu" },

{ "name": "Kerala" }

]

}

Output(GET/countries):

[

{

"id": 1,

"name": "India",

"code": "IN",

"states": [

{

"id": 1,

"name": "Tamil Nadu"

},

{

"id": 2,

"name": "Kerala"

}

]

}

]

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

QueryDemoApp.java:

package com.example.querydemo;

import org.springframework.boot.SpringApplication;

import org.springframework.boot.autoconfigure.SpringBootApplication;

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

import org.springframework.data.jpa.repository.\*;

import org.springframework.stereotype.Repository;

import org.springframework.stereotype.Service;

import jakarta.persistence.\*;

import java.util.List;

@SpringBootApplication

@RestController

@RequestMapping("/countries")

public class QueryDemoApp {

private final CountryService countryService;

public QueryDemoApp(CountryService countryService) {

this.countryService = countryService;

}

public static void main(String[] args) {

SpringApplication.run(QueryDemoApp.class, args);

}

@PostMapping

public Country addCountry(@RequestBody Country country) {

return countryService.addCountry(country);

}

@GetMapping("/hql/{name}")

public Country findByNameHQL(@PathVariable String name) {

return countryService.findByNameHQL(name);

}

@GetMapping("/native/{code}")

public Country findByCodeNative(@PathVariable String code) {

return countryService.findByCodeNative(code);

}

@Entity

public static class Country {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

private String name;

private String capital;

private String code;

public Country() {}

public Country(String name, String capital, String code) {

this.name = name;

this.capital = capital;

this.code = code;

}

public Long getId() { return id; }

public String getName() { return name; }

public String getCapital() { return capital; }

public String getCode() { return code; }

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

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

public void setCapital(String capital) { this.capital = capital; }

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

}

@Repository

interface CountryRepository extends JpaRepository<Country, Long> {

@Query("SELECT c FROM Country c WHERE c.name = :name")

Country getCountryByNameHQL(@Param("name") String name);

@Query(value = "SELECT \* FROM country WHERE code = ?1", nativeQuery = true)

Country getCountryByCodeNative(String code);

}

@Service

public static class CountryService {

private final CountryRepository repo;

public CountryService(CountryRepository repo) {

this.repo = repo;

}

public Country addCountry(Country country) {

return repo.save(country);

}

public Country findByNameHQL(String name) {

return repo.getCountryByNameHQL(name);

}

public Country findByCodeNative(String code) {

return repo.getCountryByCodeNative(code);

}

}

}

POST /countries

{

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

GET /countries/hql/India (HQL Query)

{

"id": 1,

"name": "India",

"capital": "New Delhi",

"code": "IN"

}

### GET /countries/native/IN (Native SQL Query)

### {

### "id": 1,

### "name": "India",

### "capital": "New Delhi",

### "code": "IN"

### }