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

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

Vaishnavi

5/7/2025

Spring Data JPA offers a declarative mechanism for defining repository query logic by simply declaring method names following a specific naming convention. This is known as **Query Methods**.  
Without writing a single line of JPQL or SQL, you can perform complex queries such as:

Searching with a substring

Sorting results

Filtering by prefix/suffix

Fetching between date ranges

Getting values greater than or less than a specific value

Fetching the top N results in a sorted order

**Objective:**

To demonstrate the power and simplicity of **Spring Data JPA Query Methods**

Implement search and filter functionalities on entities like Country and Stock

Enable auto-query generation using method names in repositories

Understand how Spring parses method names into SQL at runtime

**Implementation:**

**Country Entity Class: Country.java**

@Entity

@Table(name = "country")

public class Country {

@Id

@Column(name = "code")

private String code;

@Column(name = "name")

private String name;

// Getters, Setters, toString()

}

**Repository: CountryRepository.java**

@Repository

public interface CountryRepository extends JpaRepository<Country, String> {

// 1. Search by name containing a string (e.g., 'ou')

List<Country> findByNameContaining(String keyword);

// 2. Search and sort ascending by name

List<Country> findByNameContainingOrderByNameAsc(String keyword);

// 3. Filter by starting alphabet (e.g., 'Z')

List<Country> findByNameStartingWith(String prefix);

}

**MainApp**

private static void testQueryMethods() {

LOGGER.info("Start");

LOGGER.debug("Countries containing 'ou': {}", countryService.findByNameContaining("ou"));

LOGGER.debug("Countries containing 'ou' ordered: {}", countryService.findByNameContainingOrderByNameAsc("ou"));

LOGGER.debug("Countries starting with 'Z': {}", countryService.findByNameStartingWith("Z"));

LOGGER.info("End");

}

**Stock Entity Class: Country.java**

@Entity

@Table(name = "stock")

public class Stock {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private int id;

@Column(name = "st\_code")

private String code;

@Column(name = "st\_date")

private LocalDate date;

@Column(name = "st\_open")

private BigDecimal open;

@Column(name = "st\_close")

private BigDecimal close;

@Column(name = "st\_volume")

private Long volume;

// Getters, Setters, toString()

}

**Repository: CountryRepository.java**

@Repository

public interface StockRepository extends JpaRepository<Stock, Integer> {

// 1. All FB stock details in September 2019

List<Stock> findByCodeAndDateBetween(String code, LocalDate start, LocalDate end);

// 2. Google stock where price > 1250

List<Stock> findByCodeAndCloseGreaterThan(String code, BigDecimal price);

// 3. Top 3 stocks with highest volume

List<Stock> findTop3ByOrderByVolumeDesc();

// 4. Bottom 3 Netflix stocks by close value

List<Stock> findTop3ByCodeOrderByCloseAsc(String code);

}

**MainApp**

private static void testStockQueryMethods() {

LOGGER.info("Stock Queries Start");

LOGGER.debug("FB Sept 2019: {}", stockRepository.findByCodeAndDateBetween("FB",

LocalDate.of(2019, 9, 1), LocalDate.of(2019, 9, 30)));

LOGGER.debug("GOOGL > 1250: {}", stockRepository.findByCodeAndCloseGreaterThan("GOOGL",

new BigDecimal("1250")));

LOGGER.debug("Top 3 volumes: {}", stockRepository.findTop3ByOrderByVolumeDesc());

LOGGER.debug("Bottom 3 Netflix stocks: {}", stockRepository.findTop3ByCodeOrderByCloseAsc("NFLX"));

LOGGER.info("Stock Queries End");

}