

# Spring Data JPA

## Mục lục

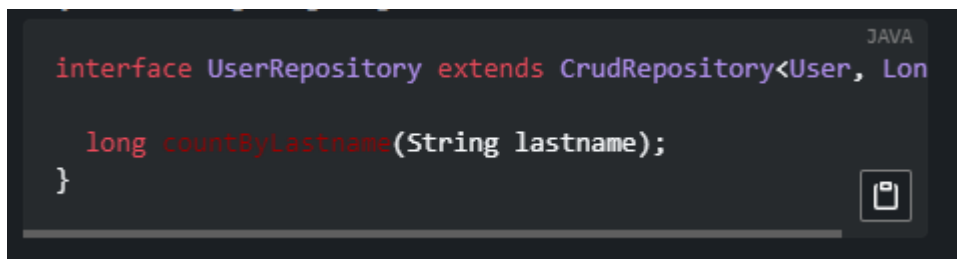
<b>1.Khái niệm cơ bản.....</b>	<b>3</b>
<b>2.Định nghĩa Repository Interface. ....</b>	<b>3</b>
<b>3.Cấu hình .....</b>	<b>3</b>
<b>4. Xác định phương thức truy vấn. ....</b>	<b>4</b>
4.1 Tạo truy vấn.....	4
4.2 Phương thức Repository trả về với Collections hoặc Iterables. ....	5
<b>4.2.1 Sử dụng Streamable làm kiểu trả về phương thức truy vấn .....</b>	<b>5</b>
<b>4.2.2 Trả về Streamable được tùy chỉnh .....</b>	<b>5</b>
<b>4.2.3 Streaming query results .....</b>	<b>6</b>
<b>4.2.4 Asynchronous query results.....</b>	<b>7</b>
4.3 Paging, Iterating Large Results, Sorting & Limiting.....	7
<b>5 Các phương thức truy vấn trong JPA.....</b>	<b>7</b>
5.1 Các từ khoá hỗ trợ trong JPA.....	7
5.2 Sử dụng annotation @Query .....	8
<b>6. Projections .....</b>	<b>9</b>

## 1. Khái niệm cơ bản.

**Spring Boot JPA** là một bản ghi chi tiết của Java để quản lý dữ liệu **quan hệ** trong các ứng dụng Java. Nó cho phép chúng ta truy cập và lưu trữ dữ liệu giữa các object/class Java và database quan hệ. JPA tuân theo **Object-Relation Mapping** (ORM). Nó là một tập hợp các interface. Nó cũng cung cấp một API **EntityManager** runtime để xử lý các câu query và giao dịch trên các object dựa trên database. Nó sử dụng ngôn ngữ truy vấn hướng đối tượng độc lập nền tảng JPQL (Java Persistent Query Language).

## 2. Định nghĩa Repository Interface.

Để định nghĩa một repository interface, đầu tiên cần định nghĩa một repository interface cho một class cụ thể. Interface phải được kế thừa từ một Repository với class và kiểu ID.



```
interface UserRepository extends CrudRepository<User, Long> {

    long countByLastname(String lastname);
}
```

## 3. Cấu hình

Cấu hình dưới đây thiết lập một embedded HSQL database bằng cách sử dụng EmbeddedDatabaseBuilder API của spring-jdbc. Sau đó Spring data thiết lập một EntityManagerFactory và sử dụng Hibernate làm mẫu nhà cung ứng bền vững. Các thành phần cấu trúc cuối được khai báo ở JpaTransactionManager. Cuối cùng kích hoạt Spring data JPA repository bằng sử dụng annotation `@EnableJpaRepositories`, về cơ bản nó mang các thuộc tính giống XML namespace. Nếu không có gói cơ sở nào được cấu hình, nó sẽ sử dụng gói cơ sở chứa lớp cấu hình.

```

main / java / com / springcore / repository / Example.java
@Configuration
@EnableJpaRepositories
@EnableTransactionManagement
class ApplicationConfig {

    @Bean
    public DataSource dataSource() {

        EmbeddedDatabaseBuilder builder = new EmbeddedDatabaseBuilder();
        return builder.setType(EmbeddedDatabaseType.HSQL).build();
    }

    @Bean
    public LocalContainerEntityManagerFactoryBean entityManagerFactory() {

        HibernateJpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();
        vendorAdapter.setGenerateDdl(true);

        LocalContainerEntityManagerFactoryBean factory = new LocalContainerEntityManagerFactoryBean();
        factory.setJpaVendorAdapter(vendorAdapter);
        factory.setPackagesToScan("com.acme.domain");
        factory.setDataSource(dataSource());
        return factory;
    }

    @Bean
    public PlatformTransactionManager transactionManager(EntityManagerFactory entityManagerFactory) {

        JpaTransactionManager txManager = new JpaTransactionManager();
        txManager.setEntityManagerFactory(entityManagerFactory);
        return txManager;
    }
}

```

#### 4. Xác định phương thức truy vấn.

Có hai cách để truy vấn dữ liệu là Bằng cách lấy truy vấn trực tiếp từ tên phương thức. và Bằng cách sử dụng truy vấn được xác định theo cách thủ công.

##### 4.1 Tạo truy vấn

Cơ chế xây dựng truy vấn được tích hợp trong cơ sở hạ tầng Spring data repository rất hữu ích để xây dựng các truy vấn ràng buộc đối với các thực thể của kho lưu trữ.

```

interface PersonRepository extends Repository<Person, Long> {

    List<Person> findByEmailAddressAndLastname(EmailAddress emailAddress, String lastname);

    // Enables the distinct flag for the query
    List<Person> findDistinctPeopleByLastnameOrFirstname(String lastname, String firstname);
    List<Person> findPeopleDistinctByLastnameOrFirstname(String lastname, String firstname);

    // Enabling ignoring case for an individual property
    List<Person> findByLastnameIgnoreCase(String lastname);
    // Enabling ignoring case for all suitable properties
    List<Person> findByLastnameAndFirstnameAllIgnoreCase(String lastname, String firstname);

    // Enabling static ORDER BY for a query
    List<Person> findByLastnameOrderByFirstnameAsc(String lastname);
    List<Person> findByLastnameOrderByFirstnameDesc(String lastname);
}

```

*Tạo truy vấn từ tên phương thức*

Phần đầu (find...By, exists...By) xác định đối tượng của truy vấn, phần thứ hai tạo ra điều kiện. Cụm từ giới thiệu (subject) có thể chứa các biểu thức bổ sung. Bất kỳ văn bản nằm giữa từ khoá giới thiệu như find (hoặc từ khoá giới thiệu khác) và By được xem là mô tả trừ khi sử dụng các từ khoá giới hạn kết quả như Distinct để thiết lập một cờ distinct cho truy vấn hoặc Top/First để giới hạn kết quả truy vấn.

Ứng dụng chia làm hai phần: từ khoá giới thiệu và từ khoá mô tả điều kiện. Từ khoá By đầu tiên hoạt động như một ngăn cách để chỉ ra bắt đầu của tiêu chí thực sự. Một cách rất cơ bản, bạn có thể xác định điều kiện trên các thuộc tính của đối tượng và nối chúng với And và Or.

## 4.2 Phương thức Repository trả về với Collections hoặc Iterables.

Các phương thức truy vấn trả về nhiều kết quả có thể sử dụng Java Iterable, List và Set tiêu chuẩn. Ngoài ra, còn hỗ trợ trả về Streamable của Spring Data, một tiện ích mở rộng tùy chỉnh của Iterable, cũng như các loại bộ sưu tập do Vavr cung cấp.

### 4.2.1 Sử dụng Streamable làm kiểu trả về phương thức truy vấn

Bạn có thể sử dụng Streamable thay thế cho Iterable hoặc bất kỳ loại bộ sưu tập nào. Nó cung cấp các phương thức tiện lợi để truy cập Luồng không song song (thiếu trong Iterable) và khả năng trực tiếp ....filter(...) và ....map(...) trên các phần tử và ghép Streamable với các phần tử khác:

```
interface PersonRepository extends Repository<Person, Long> {
    Streamable<Person> findByFirstnameContaining(String firstname);
    Streamable<Person> findByLastnameContaining(String lastname);
}

Streamable<Person> result = repository.findByFirstnameContaining("av")
    .and(repository.findByLastnameContaining("ea"));
```

*Sử dụng Streamable để kết hợp kết quả phương thức truy vấn*

### 4.2.2 Trả về Streamable được tùy chỉnh

Pattern cung cấp các loại bọc (wrapper types) riêng biệt cho các bộ sưu tập là một kỹ thuật phổ biến để cung cấp một API cho kết quả truy vấn trả về nhiều phần tử. Thông thường, các loại này được sử dụng bằng cách gọi một phương thức repository trả về một kiểu giống bộ sưu tập và tạo một thể hiện của loại bọc bằng cách thủ công. Bạn có thể tránh bước thêm này vì Spring Data cho phép bạn sử dụng trực tiếp các loại bọc này như là kiểu trả về của phương thức truy vấn nếu chúng đáp ứng các tiêu chí sau:

+Loại đóng vai trò của Streamable

+Loại đóng vai trò có một constructor hoặc một phương thức tạo tĩnh có tên là of(...) hoặc valueOf(...) nhận Streamable làm đối số.

```

class Product {
    MonetaryAmount getPrice() { ... }
}

@RequiredArgsConstructor(staticName = "of")
class Products implements Streamable<Product> {

    private final Streamable<Product> streamable;

    public MonetaryAmount getTotal() {
        return streamable.stream()
            .map(Priced::getPrice)
            .reduce(Money.of(0), MonetaryAmount::add);
    }

    @Override
    public Iterator<Product> iterator() {
        return streamable.iterator();
    }
}

interface ProductRepository implements Repository<Product, Long> {
    Products findAllByDescriptionContaining(String text);
}

```

- **Product:** Đối tượng Product cung cấp API để truy cập giá sản phẩm.
- **Products:** Loại bọc cho Streamable<Product>, có thể được tạo bằng cách sử dụng Products.of(...) (phương thức tạo được tạo bằng chú thích Lombok). Một constructor tiêu chuẩn nhận Streamable<Product> cũng sẽ hoạt động.
- Loại bọc mở rộng API, tính toán giá trị mới trên Streamable<Product>.
- Triển khai giao diện Streamable và ủy quyền cho kết quả thực tế.
- Loại bọc Products có thể được sử dụng trực tiếp như một kiểu trả về của phương thức truy vấn. Bạn không cần trả về Streamable<Product> và đóng gói nó thủ công sau truy vấn trong repository client.

#### 4.2.3 Streaming query results

Kết quả của các phương thức truy vấn có thể được xử lý một cách tăng dần bằng cách sử dụng Java 8 Stream<T> như kiểu trả về. Thay vì bọc kết quả truy vấn trong một Stream, các phương thức cụ thể cho cơ sở dữ liệu được sử dụng để thực hiện việc stream.

```

@Query("select u from User u")
Stream<User> findAllByCustomQueryAndStream();

Stream<User> readAllByFirstnameNotNull();

@Query("select u from User u")
Stream<User> streamAllPaged(Pageable pageable);

```

#### 4.2.4 Asynchronous query results.

Có thể chạy các truy vấn repository bất đồng bộ bằng cách sử dụng khả năng chạy bất đồng bộ của Spring. Điều này có nghĩa là phương thức trả về ngay lập tức sau khi được gọi trong khi truy vấn thực tế xảy ra trong một nhiệm vụ đã được gửi đến một Spring TaskExecutor. Các truy vấn bất đồng bộ khác biệt so với truy vấn reactive và không nên được kết hợp.

```
@Async
Future<User> findByFirstname(String firstname);

@Async
CompletableFuture<User> findOneByFirstname(String firstname);
```

### 4.3 Paging, Iterating Large Results, Sorting & Limiting

JPA cung cấp một số kiểu cụ thể như Pageable, Sort và Limit để áp dụng phân trang, sắp xếp và giới hạn truy vấn của bạn động.

```
Page<User> findByLastname(String lastname, Pageable pageable);

Slice<User> findByLastname(String lastname, Pageable pageable);

List<User> findByLastname(String lastname, Sort sort);

List<User> findByLastname(String lastname, Sort sort, Limit limit);

List<User> findByLastname(String lastname, Pageable pageable);
```

## 5 Các phương thức truy vấn trong JPA.

### 5.1 Các từ khoá hỗ trợ trong JPA.

Keyword	Sample	JPQL snippet
Distinct	findDistinctByLastnameAndFirstname	select distinct ... where x.lastname = ?1 and x.firstname = ?2
And	findByLastnameAndFirstname	... where x.lastname = ?1 and x.firstname = ?2
Or	findByLastnameOrFirstname	... where x.lastname = ?1 or x.firstname = ?2
Is, Equals	findByFirstname, findByFirstnameIs, findByFirstnameEquals	... where x.firstname = ?1
Between	findByStartDateBetween	... where x.startDate between ?1 and ?2
LessThan	findByAgeLessThan	... where x.age < ?1
LessThanEqual	findByAgeLessThanEqual	... where x.age <= ?1

GreaterThan	findByAgeGreaterThan	... where x.age > ?1
GreaterThan Equal	findByAgeGreaterThanEqual	... where x.age >= ?1
After	findByStartDateAfter	... where x.startDate > ?1
Before	findByStartDateBefore	... where x.startDate < ?1
IsNull, Null	findByAge(Is)Null	... where x.age is null
IsNotNull, NotNull	findByAge(Is)NotNull	... where x.age not null
Like	findByFirstnameLike	... where x.firstname like ?1
NotLike	findByFirstnameNotLike	... where x.firstname not like ?1
StartingWith	findByFirstnameStartingWith	... where x.firstname like ?1
EndingWith	findByFirstnameEndingWith	... where x.firstname like ?1
Containing	findByFirstnameContaining	... where x.firstname like ?1
OrderBy	findByAgeOrderByLastnameDesc	... where x.age = ?1 order by x.lastname desc
Not	findByLastnameNot	... where x.lastname <> ?1
In	findByAgeIn(Collection<Age> ages)	... where x.age in ?1
NotIn	findByAgeNotIn(Collection<Age> ages)	... where x.age not in ?1
True	findByActiveTrue()	... where x.active = true
IgnoreCase	findByFirstnameIgnoreCase	... where UPPER(x.firstname) = UPPER(?1)

## 5.2 Sử dụng annotation @Query

Sử dụng các truy vấn có tên để khai báo các truy vấn cho các thực thể là một phương pháp hợp lệ và hoạt động tốt cho một số lượng nhỏ các truy vấn. Do các truy vấn chính thức liên quan đến phương thức Java chạy chúng, bạn có thể liên kết chúng trực tiếp bằng cách sử dụng chú thích @Query của Spring Data JPA thay vì chú thích chúng vào lớp domain. Điều này giúp giải phóng lớp domain khỏi thông tin cụ thể về lưu trữ và đặt truy vấn tại nơi gọi là repository interface.

```
@Query("SELECT u FROM User u")
Streamable<User> findByCustomQueryAndStreamable();
```

Sử dụng Native queries trong JPA cho phép chúng ta sử dụng tất cả các tính năng được hỗ trợ bởi cơ sở dữ liệu. Điều này rất phù hợp cho một số trường hợp cần sử dụng câu lệnh truy vấn phức tạp để trích xuất thông tin cần thiết



## 6. Projections

Spring Data query methods thường trả về một hoặc nhiều phiên bản của aggregate root được quản lý bởi repository. Tuy nhiên, đôi khi có thể mong muốn tạo ra các projections dựa trên một số thuộc tính cụ thể của các loại đó. Spring Data cho phép mô hình hóa các kiểu trả về được dành riêng, để lấy các góc nhìn một phần của các aggregates được quản lý.

*A projection interface to retrieve a subset of attributes*

```
interface NamesOnly {  
  
    String getFirstname();  
    String getLastName();  
}
```

JAVA



```
interface PersonRepository extends Repository<Person, UUID> {  
  
    Collection<NamesOnly> findByLastname(String lastname);  
}
```