**SPRING DATA JPA**

**Spring Data Jpa**

* Trước khi Spring Data JPA ra đời, khi làm việc với JPA, chúng ta thường phải viết các câu truy vấn SQL để thao tác với cơ sở dữ liệu bằng cách sử dụng EntityManager và các đối tượng Entity
* Để sử dụng Spring Data JPA trong ứng dụng Spring, chúng ta cần thêm dependency spring-boot-starter-data-jpa. Dependency này cung cấp các thành phần cần thiết để làm việc với JPA và Spring Data JPA
* Spring Data JPA là một phần mở rộng của JPA được Spring cung cấp để đơn giản hóa và tăng cường khả năng làm việc với cơ sở dữ liệu trong ứng dụng Java. Nó cung cấp các tiện ích và tính năng để giảm thiểu mã lặp lại và tăng tốc quá trình phát triển
* Một số điểm nổi bật của Spring Data JPA:
  + Repository Interface: Spring Data JPA cho phép chúng ta định nghĩa các Repository Interface, mà không cần viết mã nguồn cụ thể. Các phương thức trong Repository Interface được tự động triển khai bởi Spring. Ví dụ nhu: save(), findById(), findAll(), deleteById(),...
  + Spring Data JPA cung cấp các tính năng và tiện ích để tự động tạo các truy vấn SQL dựa trên các phương thức được đặt tên theo quy ước. Chúng ta không cần viết mã SQL thủ công mà chỉ cần định nghĩa các phương thức trong các Repository interface mà không cần viết mã SQL tường minh. Ví dụ: findByFirstName(), findByLastName(),..
  + Paging và Sorting: Spring Data JPA hỗ trợ phân trang và sắp xếp kết quả truy vấn một cách dễ dàng. Chúng ta có thể chỉ định số lượng bản ghi trả về, số trang, và các tiêu chí sắp xếp
  + Query Methods: Spring Data JPA cho phép chúng ta định nghĩa các phương thức truy vấn tùy chỉnh bằng cách sử dụng các annotation như **@Query**, **@Param**, **@NamedQuery**,...
  + Hỗ trợ các tính năng JPA: Spring Data JPA hỗ trợ các tính năng của JPA như kết nối, quản lý phiên, quản lý transaction,...
* Trong Spring Data JPA, có ba interface chính là CrudRepository, PagingAndSortingRepository, và JpaRepository

A picture containing text, screenshot, font, line

Description automatically generated

* + CrudRepository
    - CrudRepository là một interface cung cấp các phương thức cơ bản để thực hiện các thao tác CRUD (Create, Read, Update, Delete) trên các đối tượng
    - Các phương thức cung cấp sẵn trong CrudRepository bao gồm save(), findById(), findAll(), deleteById(),...
    - Interface này là một phiên bản mở rộng của Repository và không cung cấp các tính năng phân trang hoặc sắp xếp kết quả truy vấn
  + PagingAndSortingRepository
    - PagingAndSortingRepository là một phiên bản mở rộng của CrudRepository và cung cấp các phương thức bổ sung để phân trang và sắp xếp kết quả truy vấn
    - Ngoài các phương thức của CrudRepository, PagingAndSortingRepository cũng cung cấp các phương thức như findAll(Pageable), findAll(Sort),... để thực hiện phân trang và sắp xếp kết quả truy vấn
  + JpaRepository
    - JpaRepository là một phiên bản mở rộng của PagingAndSortingRepository. Nó cung cấp các phương thức bổ sung cho bộ chuẩn JPA và Hibernate
    - Ngoài các phương thức của PagingAndSortingRepository, JpaRepository cung cấp các phương thức như flush() để đồng bộ dữ liệu từ Persistence Context vào cơ sở dữ liệu, deleteInBatch() để xóa một tập hợp các đối tượng, và nhiều phương thức truy vấn tùy chỉnh khác

**SimpleJpaRepository**

Trong Spring Data JPA, class SimpleJpaRepository là một implementation mặc định của interface JpaRepository. Nó cung cấp các implementation cho các phương thức đã được định nghĩa trong JpaRepository như save(), findById(), findAll(), deleteById(),...

**Class triển khai interface extends JpaRepository**

**A black text on a white background

Description automatically generated with low confidence**

* Khi chúng ta extends một interface repository từ JpaRepository, CrudRepository, PagingAndSortingRepository, hoặc các interface tương tự khác, Spring Data JPA sẽ tự động tạo ra một implementation cho interface đó bằng cách sử dụng cơ chế proxy và reflection trong Spring Framework
* Khi Spring Data JPA tự động tạo implementation, nó sử dụng SimpleJpaRepository làm lớp cơ sở để triển khai tất cả các phương thức đã được định nghĩa trong JpaRepository (hoặc tương ứng với CrudRepository hoặc PagingAndSortingRepository). Class implementation sẽ triển khai thêm các phương thức định nghĩa riêng trong interface repository cha nếu có, bao gồm cả các phương thức sử dụng cơ chế tự động tạo truy vấn dựa trên tên method (như findBy..., countBy..., deleteBy...) và các phương thức được đánh dấu bằng **@Query** để định nghĩa truy vấn tùy chỉnh
* Class implementation được tạo ra sẽ được đăng ký trong Spring ApplicationContext và sử dụng như một bean để tiếp cận các phương thức của repository
* Mô phỏng cấu trúc class implementation mà Spring tự động tạo

A black text on a white background

Description automatically generated with low confidence

**@NoRepositoryBean**

* **@NoRepositoryBean** là một annotation được sử dụng trong Spring Data JPA để đánh dấu một interface repository không cần tạo ra một implementation cụ thể. Khi một interface được đánh dấu bằng **@NoRepositoryBean**, Spring Data JPA sẽ không tự động tạo ra một implementation cho nó
* **@NoRepositoryBean** được sử dụng khi chúng ta muốn định nghĩa một interface base repository chung cho nhiều repository khác nhau mà không muốn Spring Data JPA tự động tạo ra một implementation riêng cho interface đó

**@Transactional**

* **@Transactional** là một annotation trong Spring Framework được sử dụng để quản lý transaction trong môi trường Java. Annotation này có thể được áp dụng trên một phương thức hoặc một lớp, và chỉ định rằng các phương thức trong đó sẽ thực hiện trong một transaction
* Khi một phương thức được đánh dấu bằng **@Transactional**, Spring sẽ tự động bao bọc phương thức đó trong một transaction và thực hiện commit hoặc rollback sau khi phương thức hoàn thành
* Có 2 cách sử dụng **@Transactional** trong Spring:
  + Đánh dấu trực tiếp trên phương thức: Chúng ta có thể đánh dấu **@Transactional** trực tiếp trên phương thức mà chúng ta muốn thực hiện trong một transaction

**A screenshot of a computer program

Description automatically generated with low confidence**

* + Đánh dấu trên lớp: Chúng ta có thể đánh dấu **@Transactional** trên lớp để áp dụng cho tất cả các phương thức bên trong lớp đó

**A picture containing text, screenshot, font, number

Description automatically generated**

* **@Transactional** thường được sử dụng trong các phương thức thực hiện nhiều hơn một thao tác cập nhật dữ liệu như insert, update, delete. Mục đích chính của **@Transactional** là đảm bảo tính nhất quán và đồng bộ của các thao tác cập nhật dữ liệu
* Trong trường hợp chỉ có một thao tác cập nhật dữ liệu, việc sử dụng **@Transactional** không mang lại ý nghĩa đặc biệt, vì giao dịch chỉ cần bao bọc xung quanh nhiều thao tác để đảm bảo sự nhất quán. Tuy nhiên, trong một số trường hợp, chúng ta có thể muốn sử dụng **@Transactional** để kiểm soát rõ ràng và dễ dàng mở rộng cho tương lai
* Đối với các thao tác select dữ liệu (đọc), việc sử dụng **@Transactional** không bắt buộc và không có ý nghĩa đặc biệt. **@Transactional** thường được sử dụng chủ yếu cho các thao tác cập nhật dữ liệu (insert, update, delete) để đảm bảo tính nhất quán và toàn vẹn dữ liệu trong quá trình thực thi các thao tác này

**Cơ chế xây dựng Query từ tên của method**

* Tên method
  + Tiền tố: Bắt đầu bằng tiền tố như findBy, getBy, queryBy, readBy, hoặc countBy
  + Giới hạn số lượng kết quả trả về bằng cách thêm từ khóa First hoặc Top trước By: findTopBy, findFirstBy, findTop5By, findFirs5tBy
  + Thuộc tính: Tiếp theo là tên thuộc tính của entity mà chúng ta muốn truy vấn
  + Các từ khóa kết hợp: Sử dụng các từ khóa như And, Or, Between, LessThan, GreaterThan, Like, IsNull, NotNull, v.v. để kết hợp các điều kiện truy vấn
  + Kết thúc: Cuối cùng, chúng ta có thể thêm các từ khóa như OrderBy, …
* Param method: Các tham số cần tìm kiếm được định nghĩa trong method
* Kiểu dữ liệu trả về: Phương thức có thể trả về entity, các loại collection, array entity, các kiểu dữ liệu nguyên thủy, kiểu dữ liệu Wrapper tương ứng hoặc optional, ...

Chú ý: Trong trường hợp tìm kiếm thuộc tính nằm trong một thuộc tính của entity khác, chúng ta có thể sử dụng toán tử '\_' để truy xuất đến thuộc tính đó. Thông thường, mỗi '\_' tương ứng với một mức truy xuất vào thuộc tính của entity:

A picture containing text, font, screenshot, line

Description automatically generated

**@Query**

* Khi làm việc với các bảng không có mối quan hệ trực tiếp với entity hiện tại, cơ chế xây dựng truy vấn từ tên method không thể áp dụng được. Trong trường hợp này, chúng ta cần tự viết câu lệnh truy vấn sử dụng **@Query** để định nghĩa truy vấn tùy chỉnh
* Khi sử dụng **@Query** để viết truy vấn tùy chỉnh, cơ chế xây dựng truy vấn từ tên method không còn áp dụng cho phương thức đó nữa
* Trong Spring Data JPA, có hai ngôn ngữ truy vấn chính được sử dụng với **@Query** là JPQL (Java Persistence Query Language) và SQL nguyên thủy
  + JPQL: Khi sử dụng JPQL, chúng ta viết câu truy vấn tương tự như viết SQL, nhưng thay vì tham chiếu đến các bảng và cột, chúng ta sử dụng tên entity và thuộc tính của nó
  + SQL nguyên thủy: Khi sử dụng truy vấn SQL nguyên thủy, chúng ta có thể viết truy vấn SQL trực tiếp để truy vấn dữ liệu

A picture containing text, screenshot, font, line

Description automatically generated

* Để tham chiếu đến các tham số trong câu truy vấn, chúng ta có thể sử dụng :paramName hoặc ?1, ?2,... để đại diện cho các tham số theo thứ tự. Đối với việc sử dụng :paramName, chúng ta có thể xác định và truyền giá trị cho tham số trong **@Query** thông qua **@Param**
* Kết quả trả về của truy vấn **@Query** có thể là entity, collection, array, kiểu dữ liệu nguyên thủy, Optional hoặc DTO. Chúng ta có thể tuỳ chỉnh kiểu dữ liệu trả về để phù hợp với nhu cầu của mình. Lưu ý: giá trị trong câu truy vấn SELECT và kiểu dữ liệu trả về của method phải khớp với nhau: entity, dto, String, kiểu nguyên thủy, …

A picture containing text, screenshot, font, line

Description automatically generated

* **@Query** trong Spring Data JPA thường được sử dụng để thực hiện các truy vấn SELECT (truy vấn dữ liệu), trong khi các truy vấn DML (UPDATE, DELETE, INSERT) không được hỗ trợ trực tiếp bởi **@Query**. Để thực hiện các truy vấn DML, chúng ta cần sử dụng **@Query** kết hợp với **@Modifying** và **@Transactional**
  + **@Modifying** cho phép chúng ta đánh dấu truy vấn là một truy vấn cập nhật dữ liệu (DML) và nó cần được sử dụng kết hợp với **@Query** trong truy vấn DML

A screenshot of a computer program

Description automatically generated with low confidence

**Trả về DTO**

* **Trả về DTO từ cơ chế tạo truy vấn từ tên method**: Khi sử dụng cơ chế tạo truy vấn tự động từ tên method (Query Creation), Spring Data JPA không hỗ trợ trực tiếp việc trả về DTO. Kết quả truy vấn sẽ trả về các entity được chỉ định trong repository. Tuy nhiên, chúng ta có thể sử dụng các bộ chuyển đổi (mapper) hoặc thư viện ánh xạ tự động để chuyển đổi kết quả truy vấn từ entity sang DTO như ModelMapper, MapStruct hay Dozer, …
* **Trả về DTO từ @Query**

A screenshot of a computer code

Description automatically generated with low confidence

* + Sử dụng Constructor-based DTO Projection

A screenshot of a computer code

Description automatically generated with low confidence

* + Sử dụng Interface-based DTO projections

A screenshot of a computer program

Description automatically generated with low confidence

**Paging và Sorting trong Spring Data JPA**

**A picture containing text, screenshot, font, diagram

Description automatically generated**

* Interface PagingAndSortingRepository hỗ trợ 2 method để paging và sorting sẵn:

**A computer code on a white background

Description automatically generated with low confidence**

* Tạo đối tượng PageRequest từ class PageRequest:

A screen shot of a computer code

Description automatically generated with low confidence

Ví dụ:

A picture containing text, screenshot, font

Description automatically generated

* Thực hiện paging và sorting

Với hàm có sẵn:



Với cơ chế tự tạo query từ tên method:

A black text on a white background

Description automatically generated with low confidence

Với **@Query**:

A picture containing text, font, screenshot, line

Description automatically generated

* Pageable là một cách tiếp cận rất tiện lợi, tuy nhiên chúng ta cần biết rằng các hàm sử dụng Pageable sẽ thực hiện query 2 lần xuống DB. Một lần là để lấy ra tổng số lượng bản ghi, một lần là query lấy ra page chúng ta cần

**Khi các tính năng truy vấn dữ liệu của Spring Data JPA không đáp ứng được yêu cầu cụ thể**

Khi các tính năng truy vấn dữ liệu của Spring Data JPA không đáp ứng được yêu cầu cụ thể, chúng ta có thể quay lại sử dụng JPA trực tiếp để tạo câu truy vấn động. JPA cung cấp một số cơ chế như Criteria API, Native Queries và JPQL để tạo và thực thi các truy vấn linh hoạt. Khi chúng ta cần xử lý các yêu cầu phức tạp hơn, việc sử dụng JPA trực tiếp có thể linh hoạt và mạnh mẽ hơn

**A picture containing text, screenshot, font, document

Description automatically generated**

**A picture containing text, screenshot, line, font

Description automatically generated**

**Cơ chế Auditing trong Spring Data JPA**

* Trong Spring Data JPA, cơ chế Auditing là một tính năng mạnh mẽ giúp tự động theo dõi và quản lý các thông tin như ngày tạo, ngày cập nhật, người tạo, người cập nhật và các thuộc tính liên quan khác của các đối tượng trong hệ thống của bạn. Điều này giúp chúng ta theo dõi và theo dõi lịch sử thay đổi của các đối tượng một cách dễ dàng
* Cơ chế Auditing trong Spring Data JPA được xây dựng dựa trên annotation và các đối tượng liên quan. Các annotation chính được sử dụng trong cơ chế Auditing bao gồm:
  + **@EnableJpaAuditing**: Đây là một annotation được đặt trên lớp cấu hình của ứng dụng Spring Boot để bật tính năng Auditing
  + **@EntityListeners(AuditingEntityListener.class)**: Đây là một annotation được đặt trên lớp entity để chỉ định rằng entity đó sẽ tham gia vào quá trình Auditing
  + **@CreatedDate**, **@LastModifiedDate**, **@CreatedBy**, **@LastModifiedBy**: Đây là các annotation được đặt trên các trường của entity. Giá trị của các trường này sẽ tự động được cập nhật khi người dùng thực hiện thao tác trên đối tượng

A screenshot of a computer program

Description automatically generated with medium confidence

A screen shot of a computer code

Description automatically generated