Querydsl in combination with JPA

1. Querydsl
2. Vẫn là querydsl
3. là

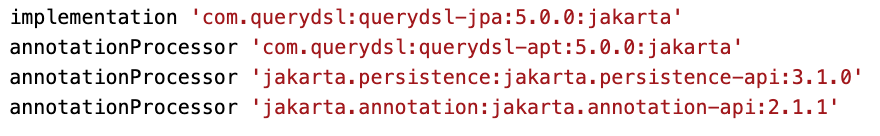
**Querydsl** là một thư viện Java mạnh mẽ giúp xây dựng các truy vấn kiểu an toàn (type-safe) cho các cơ sở dữ liệu, API, và các hệ thống khác. Khi kết hợp với **JPA (Java Persistence API)**, Querydsl cung cấp một cách linh hoạt và hiệu quả để thực hiện các truy vấn phức tạp mà không cần viết các chuỗi JPQL thủ công, giảm thiểu lỗi và tăng tính bảo trì của mã nguồn.

1. tại sao nên dùng

* type-safe queries: tạo lớp meta (Q) trên thực thể jpa (entity?)
* DSL (domain specific lang): các ngôn ngữ chuyên dụng truy vấn, dễ đọc dễ hiểu
* hỗ trợ nhiều backend: ngoài jpa, còn nhiều sql, mongo, hibernate search, …
* dynamic quering: ví dụ ở dưới cho trưc quan

1. haizzz
2. thêm dependency 🙂
3. Biết nói j đây :/

chat không biết cái ni so tra lòi mắt stack, medium ….. bla bla ko ra, cuối cùng lấy mỗi chỗ 1 tí lại được, ảo quá :v



1. hết rồi :v
2. Các đối tượng chính của QueryDSL
3. Lớp Q (Q-Type)

tự động sinh ra bởi QueryDSL từ các thực thể JPA. Các lớp này được dùng để biểu diễn các thực thể trong truy vấn một cách an toàn về cú pháp.

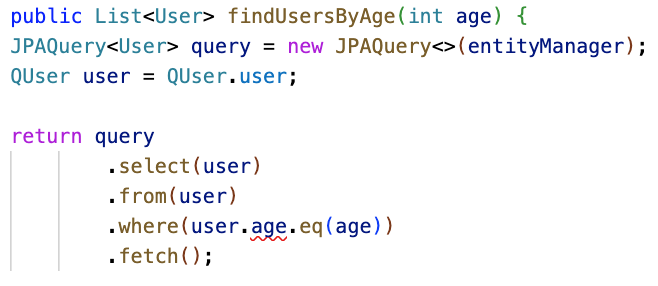
1. JPAQuery (EntityManager)
2. là …

JPAQuery là đối tượng truy vấn chính, cho phép bạn xây dựng các câu truy vấn thông qua các phương thức như .select(), .from(), .where(), và .orderBy().

1. khi …

trực tiếp để xây dựng truy vấn trong trường hợp truy vấn đơn giản, không cần thiết lập phức tạp, không quan tâm đến tái sử dụng EntityManager, có thể chỉ cần JPAQuery.

1. Ví dụ



1. JPAQueryFactory
2. dùng ..

dùng như một factory để tạo ra các instance của JPAQuery. Nó giúp dễ dàng tạo các đối tượng JPAQuery mà không cần phải khởi tạo EntityManager cho mỗi truy vấn.

1. với …

trong các ứng dụng lớn, giúp quản lý và tái sử dụng EntityManager để tối ưu hóa hiệu suất. Bạn có thể inject JPAQueryFactory vào các repository, giúp mã dễ bảo trì và đọc hiểu hơn.

1. ví dụ



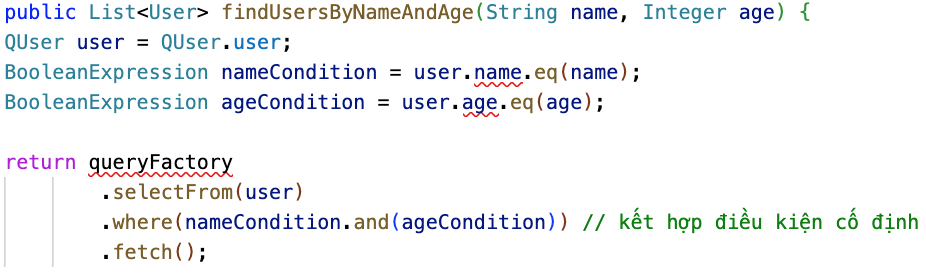
1. BooleanExpression
2. là …

đại diện cho các điều kiện logic (where), thường được tạo ra tự động dựa trên các thuộc tính của lớp Q-type. Nó cho phép kết hợp các điều kiện với and() và or() để tạo ra các biểu thức phức tạp.

1. khi …

Thường dùng BooleanExpression khi bạn có một điều kiện tĩnh hoặc chỉ cần kết hợp một số điều kiện cố định.

1. ví dụ



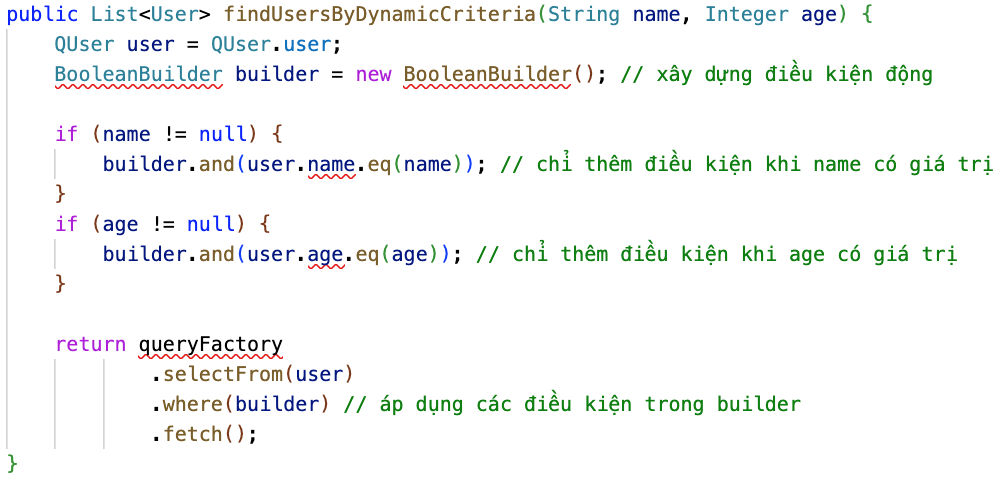
1. BooleanBuilder
2. để …

xây dựng các điều kiện động, có thể thêm hoặc loại bỏ các điều kiện dựa trên các giá trị đầu vào. Nó cung cấp cách tiện lợi để khởi tạo và kết hợp các BooleanExpression mà không cần nối thủ công các điều kiện.

1. khi …

quản lý các điều kiện phức tạp khi nhiều điều kiện khác nhau cần được áp dụng dựa trên các giá trị runtime.

1. ví dụ



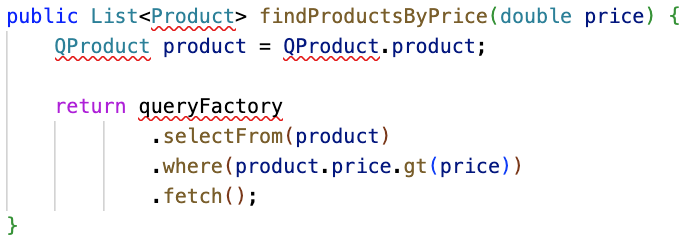
1. Path
2. là …

Path là đại diện của các trường trong thực thể, thường được tự động sinh ra bởi các lớp Q-type (như QUser, QOrder). Nó giúp bạn truy cập các thuộc tính của entity một cách trực tiếp.

1. khi …

khi bạn có một entity đã định nghĩa sẵn, và bạn muốn truy vấn dựa trên các thuộc tính của entity đó.

1. ví dụ



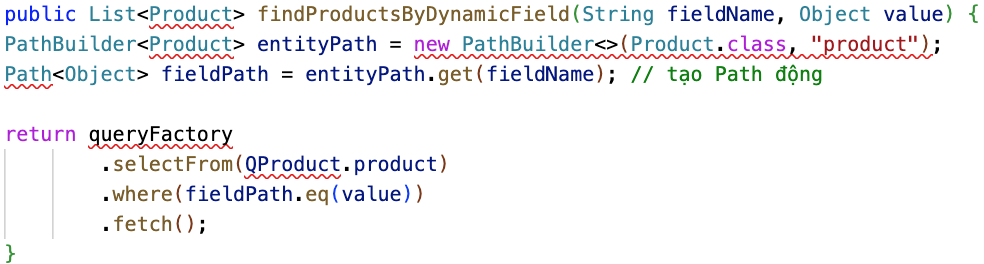
1. PathBuilder
2. để …

PathBuilder cho phép tạo Path một cách động, thường được sử dụng khi cần xây dựng truy vấn mà không biết trước tên của các trường trong entity (chẳng hạn khi làm việc với các truy vấn động).

1. khi …

PathBuilder giúp xây dựng các Path khi tên của thuộc tính cần truy vấn chỉ được xác định trong thời gian chạy (runtime). Ví dụ: new PathBuilder<>(User.class, "user").

1. ví dụ



1. Projection
2. để

chọn các trường cụ thể từ kết quả truy vấn thay vì phải lấy toàn bộ đối tượng (entity).

1. các loại + ex (Giả sử chúng ta có entity User với các thuộc tính id, name, và email. Bây giờ, bạn chỉ cần name và email của User)

* **Projections.bean()**: Tạo một đối tượng Java (POJO) từ các trường được chỉ định.

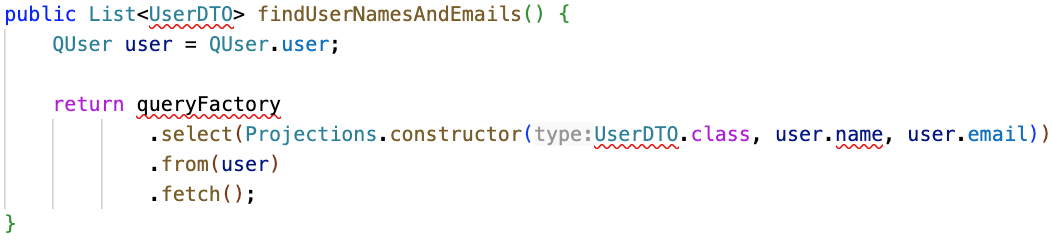


* **Projections.fields()**: Trả về một đối tượng với các thuộc tính được ánh xạ đến các trường cụ thể.



* **Projections.constructor()**: Gọi constructor của một class với các giá trị từ kết quả truy vấn.

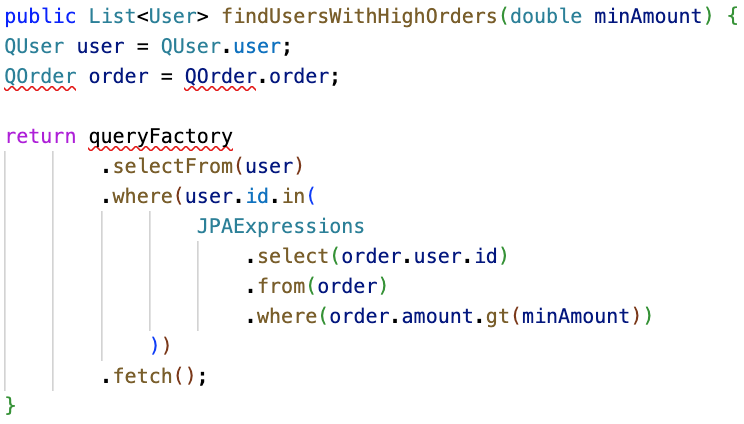
\* Projections.constructor() yêu cầu lớp UserDTO có constructor tương ứng với các tham số name và email



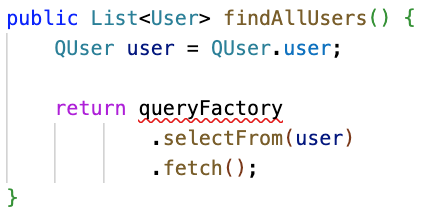
1. Subquery
2. là …

select lồng nhau

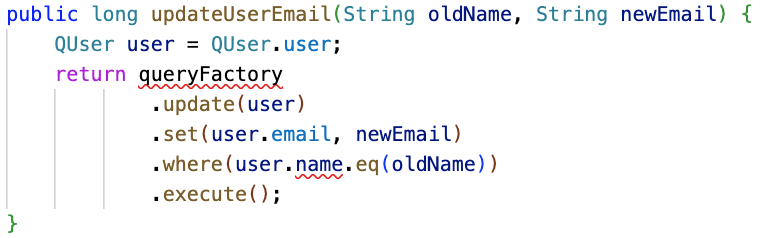
1. ví dụ : User và Order, và bạn muốn lấy danh sách User có Order có giá trị lớn hơn một số tiền nhất định.



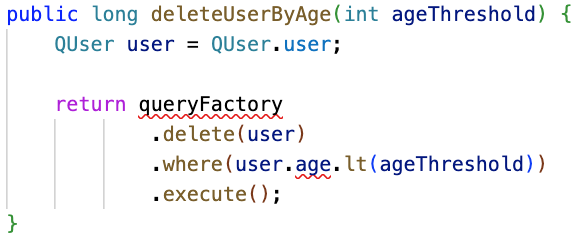
1. Tương tác với cơ sở dữ liệu
2. SELECT



1. UPDATE

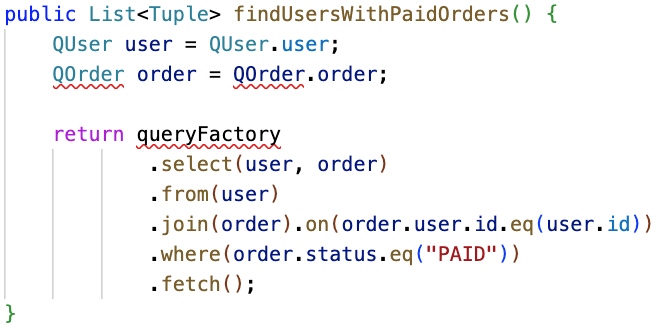


1. DELETE

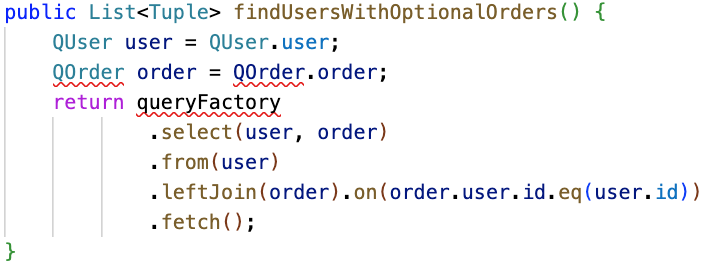


1. JOIN (Ex với - order 1-n)

1. INNER JOIN (giao 2 bảng)



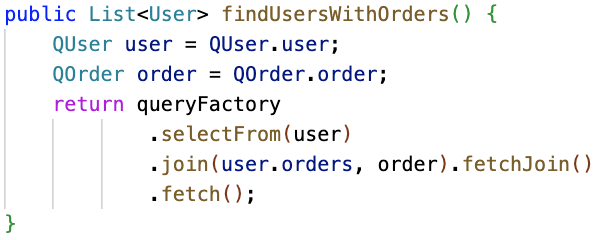
2. LEFT JOIN trả tất cả các bản ghi từ bảng User, kể cả khi không có bản ghi tương ứng trong Order



3. RIGHT JOIN

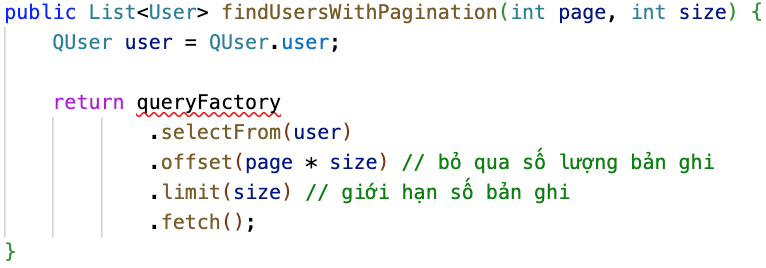
**RIGHT JOIN** không được hỗ trợ trực tiếp trong QueryDSL vì nó ít phổ biến và có thể được mô phỏng bằng cách thay đổi thứ tự của bảng.

5. FETCH JOIN

* tránh vấn đề n+1 : )
* ví dụ:

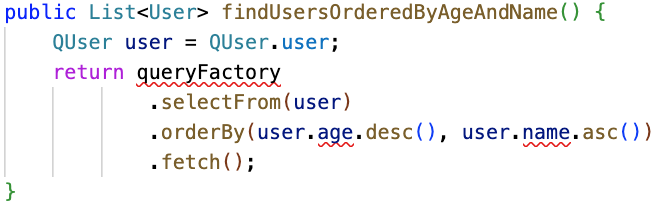
1. Phân trang .offset() và .limit()

Ví dụ: bỏ qua page \* size bản ghi đầu tiên và chỉ lấy size bản ghi tiếp theo.



1. .orderBy()

Ví dụ: sắp xếp User theo age giảm dần và name tăng dần.



1. Các Phương thức và Toán tử Thường dùng trong QueryDSL
2. Toán tử Điều kiện

1. eq(): Bằng.

2.ne(): Khác.

3.gt(): Lớn hơn.

4.lt(): Nhỏ hơn.

5.goe(): Lớn hơn hoặc bằng.

6.loe(): Nhỏ hơn hoặc bằng.

7.in(): Kiểm tra giá trị có thuộc tập giá trị nào đó.

8.isNull(), isNotNull(): Kiểm tra giá trị null.

1. Các Phương thức Truy vấn Khác
2. E tìm hiểu có những vậy, có gì e bổ sung sau :/