JPA Criteria API cho phép ta tạo ra các câu truy vấn bằng Java Object thay vì việc khai báo trực tiếp trong String (JPQL) như thế này:

SELECT o FROM Office o

Tương đương với câu lệnh trên nhưng xây dựng bằng Criteria API thì sẽ như này:

CriteriaBuilder cb = em.getCriteriaBuilder();

CriteriaQuery<Office> q = cb.createQuery(Office.class);

Root<Office> c = q.from(Office.class);

q.select(c);

Nhìn có vẻ dài dòng và khó hiểu phải không >"< Yea, thì đúng là như thế đấy ==!

Nếu nó dài dòng như vậy, tại sao người ta lại tạo ra và sử dụng nó thay cho câu lệnh JPQL bình thường? Các bạn đọc phần tiếp theo sẽ rõ nha.

**JPQL vs Criteria API**

JPQL có thể làm đầy đủ chức năng chúng ta cần chỉ với 1 câu lệnh, tuy nhiên, chính vì điều đó, chúng ta thường khó tùy biến hay sử dụng lại nó, thậm chí khó kiểm soát lỗi của nó hơn. Với một câu lệnh phức tạp, chúng ta không biết được nó có lỗi hay không cho tới khi chạy chương trình hay debug (Mà chương trình đã chạy được rồi thì vẫn có lỗi tiềm ẩn :v chời đậu).

Criteria API thì ngược lại, nó cho phép chúng ta xây dựng câu lệnh một cách Dynamic, rất linh động, và không bị hardcode trong một String và có thể tái sử dụng lại được. Đặc biệt, vì là Java Object, nên chúng ta sẽ biết một câu lệnh bị lỗi, không đúng quy tắc ngay khi biên dịch chương trình rồi.

Túm váy lại, với một lệnh đơn giản như ví dụ đầu bài, thì các bạn nên xài JPQL, còn với những câu lệnh phức hợp, thay đổi theo context của chương trình thì nên sử dụng Criteria.

**How to use.**

Quay trở lại với ví dụ ban đầu nhé:

CriteriaBuilder builder = em.getCriteriaBuilder();

CriteriaQuery<Office> query = builder.createQuery(Office.class);

Root<Office> root = query.from(Office.class);

query.select(root);

Chúng ta cùng tìm hiểu từng dòng lệnh:

* CriteriaBuilder: Để xây dựng một câu query, các bạn sẽ cần tới interface CriteriaBuilder, mục đích của nó là giúp tạo ra đối tượng chứa câu lệnh truy vấn CriteriaQuery và cung cấp cơ số các phép biến đổi, phép logic, điều kiện cho câu lệnh (and, or, not, avg, greater than,v.v...)
* CriteriaQuery: Đối tượng chính của chúng ta đây, nó được tạo ra bởi builder.createQuery(Office.class). Mục đích là khai báo đối tượng bạn muốn lấy ra sau khi thực hiện query. Nó tương đương với đoạn ngoặc đơn ở dưới đây:

SELECT `o` FROM Office o

* Root: root là khai báo đối tượng bạn sẽ sử dụng trong query, tương đương với đối tượng sau mệnh đề FROM

SELECT o FROM `Office o`

Cuối cùng, để hoàn thiện câu lệnh SELECT chỉ đơn giản là lấy đối tượng CriteriaQuery đã khai báo là sử dụng function select. Đối tượng truyền vào chính là cái root (hay cái đối tượng của FROM) kia.

query.select(root);

Trông vậy chứ cũng dễ dễ rồi đấy nhỉ :)))

Okie, có điều này không biết đã bạn nào để ý chưa 😅 chúng ta mới tạo ra câu lệnh, chứ chưa hề gọi nó xuống Database 😅

Để sử dụng câu lệnh đã tạo, các bạn làm giống với JPQL đó là sử dụng đối tượng EntityManager

TypedQuery<Office> query = em.createQuery(query);

List<Office> results = query.getResultList();

Oh right, thế là implement xong ví dụ đơn giản đâu tiên, không cóa gì khó khăn 🤔 (chém). Bây giờ thử advanced lên tý nhỉ:

Bây giờ mình muốn lấy tất cả Office ở thành phố hanoi thì sẽ làm như nào?

SELECT o FROM Office o WHERE o.city = 'hanoi'

Lúc này query của chúng ta sẽ như thế này:

query.select(root).where(builder.equal(root.get("city"), "hanoi"));

Các bạn để ý đoạn này nhé. Mình sử builder để lấy hàm equal (phép toán logic, như mình đề cập ở trên, chuẩn chưa nào). Tiếp tới là cái root.get("city"), root chính là đối tượng chúng ta đã khai báo, bây giờ chúng ta sẽ lấy trường city của nó và kiểm tra nó với hanoi.

Có thể nói Criteria API đã hiện thực hóa rất thành công câu lệnh JPQL (hay HQL) thành những api java cực kì dễ dọc, dễ hiểu và dễ sử dụng. Khi đã hiểu được câu lệnh gốc, bạn có thể dễ dàng chuyển nó thành Criteria và ngược lại.

Trong bài viết tới ở chương Spring, mình sẽ hướng dẫn các bạn sử dụng tới Specifications kết hợp Criteria API để tạo ra một vụ nổ khi giao tiếp với db bằng Java (vãi cả chém 😂)

Chúc các bạn sử dụng thành công Criteria API và đừng quên like or chia sẻ bài viết cho bạn bè hihi, cảm ơn các bạn đã theo dõi!