초격차 패키지 Online.

복잡한 쿼리의 작성과 응용

PART1 | Querydsl vs. Jooq

Querydsl 과 Jooq 기본 살펴보기

PART2 | Querydsl 활용

Querydsl + Spring Boot 2.5

PART3 | Jooq 활용

Jooq, 정말 쓸만할까요?

PART4 | eager fetch, lazy fetch, N+1 문제

JPA 사용하면서 가장 흔히 겪는 문제 1탄

PART5 | 순환 참조 문제

JPA 사용하면서 가장 흔히 겪는 문제 2탄

복잡한 쿼리의 작성과 응용

1 Querydsl vs. Jooq

Querydsl

"Unified Queries for Java. Querydsl is compact, safe and easy to learn."

- 자바 코드(엔티티) -> DB 쿼리 생성 도구
- HQL 생성 라이브러리
  - type-safety 가 부족한 HQL(JPQL)의 대안
  - 읽기 어려운 Criteria API의 대안

Querydsl

기존 기술과 비교

- JPQL: type-safety 가 좀 아쉽다
- Criteria: 어렵다

IDE 가 도와주고는 있음

```
@Query("select e from Event e " +
        "where e.place.placeName = :placeName" +
        "  and e.eventName = :eventName" +
        "  and e.eventStatus = :eventStatus" +
        "  and e.eventStartDatetime >= :eventStartDatetime" +
        "  and e.eventEndDatetime <= :eventEndDatetime")
Page<EventViewResponse> findEventViewPageBySearchParams(
        String placeName,
        String eventName,
        EventStatus eventStatus,
        LocalDateTime eventStartDatetime,
        LocalDateTime eventEndDatetime,
        Pageable pageable
);
```

검증하지 않은 JPQL - 대략 느낌만 보세요

문자열로 컬럼명을 쓰지 않고 메타모델(metamodel)을 사용하면, type-safety 확보 가능

```
CriteriaBuilder builder = em.getCriteriaBuilder();
CriteriaQuery<EventViewResponse> query = builder.createQuery(EventViewResponse.class);
Root<Event> root = query.from(Event.class);
Join<Event, Place> join = root.join( attributeName: "place", JoinType.INNER);

query.select(builder.construct(
        EventViewResponse.class,
        root.get("id"),
        root.get("placeName"),
        root.get("eventName"),
        root.get("eventStatus"),
        root.get("eventStartDatetime"),
        root.get("eventEndDatetime"),
        root.get("currentNumberOfPeople"),
        root.get("capacity"),
        root.get("memo")
))
        .where(builder.like(join.get("placeName"),  pattern: "%" + placeName + "%"))
        .where(builder.like(join.get("eventName"),  pattern: "%" + eventName + "%"))
        .where(builder.equal(join.get("eventStatus"), eventStatus))
        .where(builder.greaterThanOrEqualTo(join.get("eventStartDatetime"), eventStartDatetime))
        .where(builder.lessThanOrEqualTo(join.get("eventEndDatetime"), eventEndDatetime));

TypedQuery<EventViewResponse> typedQuery = em.createQuery(query);
List<EventViewResponse> results = typedQuery
        .setFirstResult(pageable.getPageNumber())
        .setMaxResults(pageable.getPageSize())
        .getResultList();
new PageImpl(results, pageable, count); // count 는 따로 쿼리 만들고 조회
```

검증하지 않은 Criteria Query - 대략 느낌만 보세요

Querydsl

Querydsl 코드

- 보다 readable 한 쿼리 작성
- 편리한 join
- 스프링 Pageable 과 매끄러운 연동

```
QEvent event = QEvent.event;

JPQLQuery<EventViewResponse> query = from(event)
        .select(Projections.constructor(
                EventViewResponse.class,
                event.id,
                event.place.placeName,
                event.eventName,
                event.eventStatus,
                event.eventStartDatetime,
                event.eventEndDatetime,
                event.currentNumberOfPeople,
                event.capacity,
                event.memo
        ));

if (placeName != null && !placeName.isBlank()) {
    query.where(event.place.placeName.contains(placeName));
}
if (eventName != null && !eventName.isBlank()) {
    query.where(event.eventName.contains(eventName));
}
if (eventStatus != null) {
    query.where(event.eventStatus.eq(eventStatus));
}
if (eventStartDatetime != null) {
    query.where(event.eventStartDatetime.goe(eventStartDatetime));
}
if (eventEndDatetime != null) {
    query.where(event.eventEndDatetime.loe(eventEndDatetime));
}

List<EventViewResponse> events = Optional.ofNullable(getQuerydsl())
        .orElseThrow(() -> new GeneralException(ErrorCode.DATA_ACCESS_ERROR,
        .applyPagination(pageable, query).fetch();

return new PageImpl<>(events, pageable, query.fetchCount());
```

Jooq

"jOOQ generates Java code from your database and lets you build type safe SQL queries through its fluent API."

- DB schema -> Java class 생성 도구
- ORM framework 가 아님
- "jOOQ is not a replacement for JPA"
  - SQL 이 잘 어울리는 곳엔, Jooq 가 잘 맞아요
  - Object Persistence 가 잘 어울리는 곳엔, JPA 가 잘 맞아요
- Jooq says: "Jooq + JPA"

Jooq

어떻게 생겼나 슬쩍 보기

```
final Event EVENT = Event.EVENT;
final Place PLACE = Place.PLACE;

Condition condition = trueCondition();

SelectField<?>[] select = {
        EVENT.ID,
        PLACE.PLACE_NAME,
        EVENT.EVENT_NAME,
        EVENT.EVENT_STATUS,
        EVENT.EVENT_START_DATETIME,
        EVENT.EVENT_END_DATETIME,
        EVENT.CURRENT_NUMBER_OF_PEOPLE,
        EVENT.CAPACITY,
        EVENT.MEMO
};

if (placeName != null && !placeName.isBlank()) {
    condition = condition.and(PLACE.PLACE_NAME.containsIgnoreCase(placeName));
}
if (eventName != null && !eventName.isBlank()) {
    condition = condition.and(EVENT.EVENT_NAME.contains(eventName));
}
if (eventStatus != null) {
    condition = condition.and(EVENT.EVENT_STATUS.eq(eventStatus));
}
if (eventStartDatetime != null) {
    condition = condition.and(EVENT.EVENT_START_DATETIME.ge(eventStartDatetime));
}
if (eventEndDatetime != null) {
    condition = condition.and(EVENT.EVENT_END_DATETIME.le(eventEndDatetime));
}

int count = dslContext
        .selectCount()
        .from(EVENT)
        .innerJoin(PLACE)
        .onKey()
        .where(condition)
        .fetchSingleInto(int.class);

List<EventViewResponse> pagedList = dslContext
        .select(select)
        .from(EVENT)
        .innerJoin(PLACE)
        .onKey()
        .where(condition)
        .limit(pageable.getOffset(), pageable.getPageSize())
        .fetchInto(EventViewResponse.class);

return new PageImpl<>(pagedList, pageable, count);
```

Reference

- https://querydsl.com/
- https://www.jooq.org/
- https://www.jooq.org/doc/3.15/manual-single-page/#jooq-and-jpa