🎀

라이브 세션-2022.08.30(화)-Spring Data JDBC를 이용한 데이터 액세스 실습

아바타 실습

✅ 공지사항 기능 구현하면서 리뷰하기

도메인 엔티티 클래스와 테이블 설계

DDD란?

- 도메인 주도 설계(Domain Driven Design)
- 한마디로 모든 기능을 도메인 모델 위주로 돌아가는 설계 기법
- 도메인(Domain)이란?
  - **비즈니스적인 어떤 업무 영역**
  - **우리가 실제로 현실 세계에서 접하는 업무의 한 영역**
- 코드로 이해

✅ 빈약한 도메인 모델

MemberService(서비스 클래스)

```java
@Service
public class MemberService {
    private final MemberRepository memberRepository;
    private final JdbcTemplate jdbcTemplate;
    public MemberService(MemberRepository memberRepository, JdbcTemplate jdbcTemplate) {
        this.memberRepository = memberRepository;
        this.jdbcTemplate = jdbcTemplate;
    }

    public Member createMember(Member member) {...}

    public Member updateMember(Member member) {...}

    public Member findMember(long memberId) { return findVerifiedMember(memberId); }

    public List<Member> findMembers() {...}

    public void deleteMember(long memberId) {...}

    public Member findVerifiedMember(long memberId) {...}

    private void verifyExistsEmail(String email) {...}
}
```

서비스 클래스에 기능 집중

Member(도메인 엔티티 클래스)

```java
@Getter
@Setter
@NoArgsConstructor
public class Member {
    @Id
    private Long memberId;

    private String email;

    private String name;

    private String phone;
}
```

기능이 없는 빈약한 도메인 모델

✅ 풍부한 도메인 모델

MemberService(서비스 클래스)

```java
@Service
public class MemberService {
    private final MemberRepository memberRepository;
    private final JdbcTemplate jdbcTemplate;
    public MemberService(MemberRepository memberRepository, JdbcTemplate jdbcTemplate) {
        this.memberRepository = memberRepository;
        this.jdbcTemplate = jdbcTemplate;
    }
    ...
    ...
}
```

서비스 클래스의 기능 축소

기능 이전

Member(도메인 엔티티 클래스)

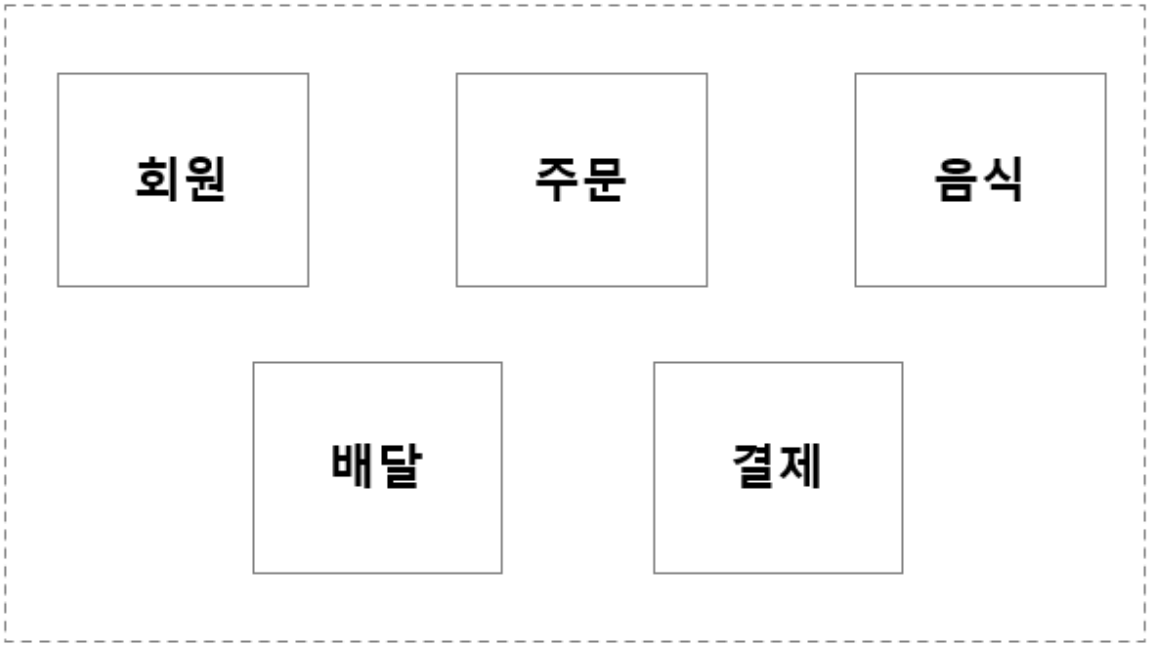
```java
@Getter
@Setter
@NoArgsConstructor
public class Member {
    @Id
    private Long memberId;

    private String email;

    private String name;

    private String phone;

    public Member createMember(Member member) {...}

    public Member updateMember(Member member) {...}

    public Member findMember(long memberId) { return findVerifiedMember(memberId); }

    public List<Member> findMembers() {...}

    public void deleteMember(long memberId) {...}

    public Member findVerifiedMember(long memberId) {...}

    private void verifyExistsEmail(String email) {...}
}
```

기능이 많은 풍부한 도메인 모델(Rich Domain)

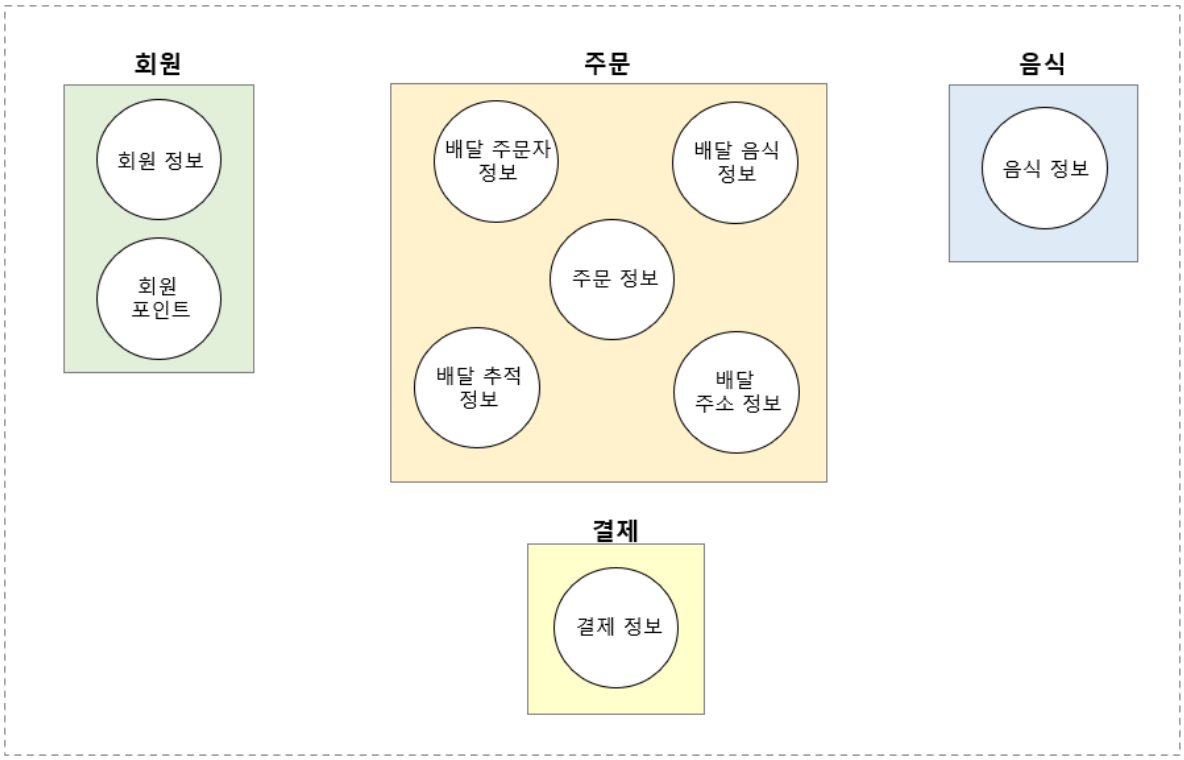
애그리거트(Aggregate)란?

비슷한 업무 도메인들의 묶음

✅ 배달 주문 앱의 도메인 모델 예



✅ 배달 주문 앱 도메인에서의 애그리거트(Aggregate)

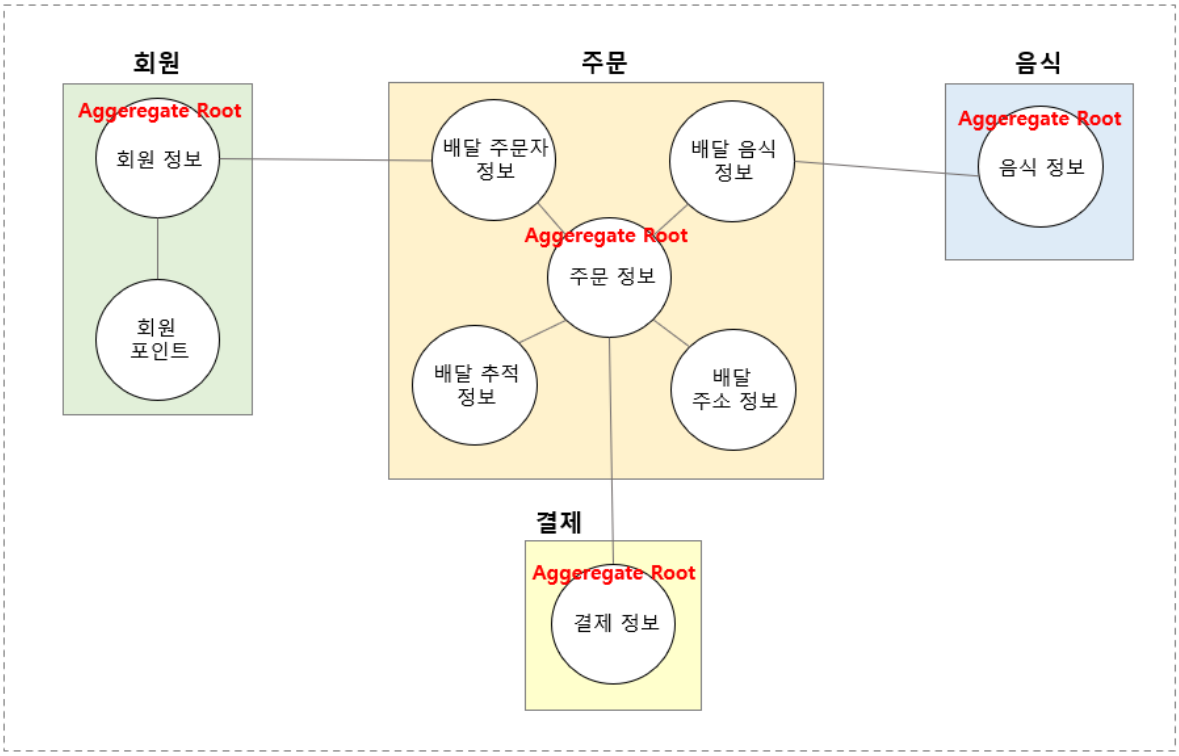


✅ 애그리거트 루트(Aggregate Root)

하나의 애그리거트를 대표하는 도메인

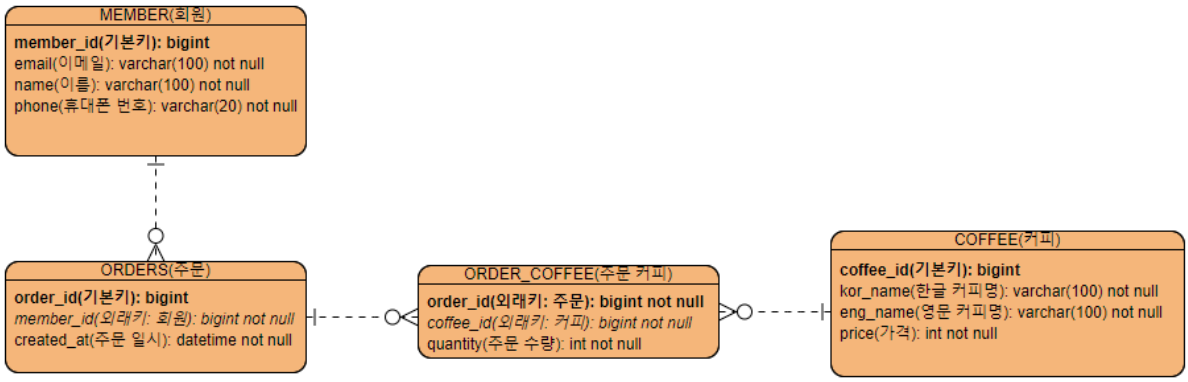
✅ 배달 주문 앱 도메인에서의 애그리거트 루트(Aggregate Root)

- 애그리거트에서 대장 격인 도메인
- 다른 도메인과 직간접적으로 연결되는 도메인
- 데이터베이스의 테이블 간 관계에서는 부모 테이블이 애그리거트 루트, 자식 테이블은 애그리거트 루트가 아닌 다른 도메인이 된다.

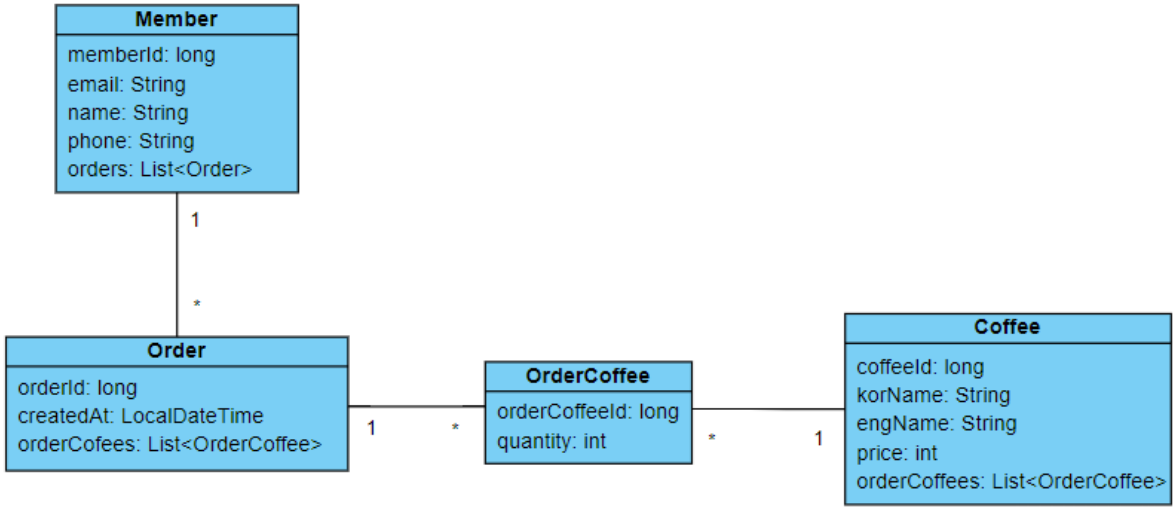


커피 주문 샘플 애플리케이션 테이블 및 도메인 엔티티 설계

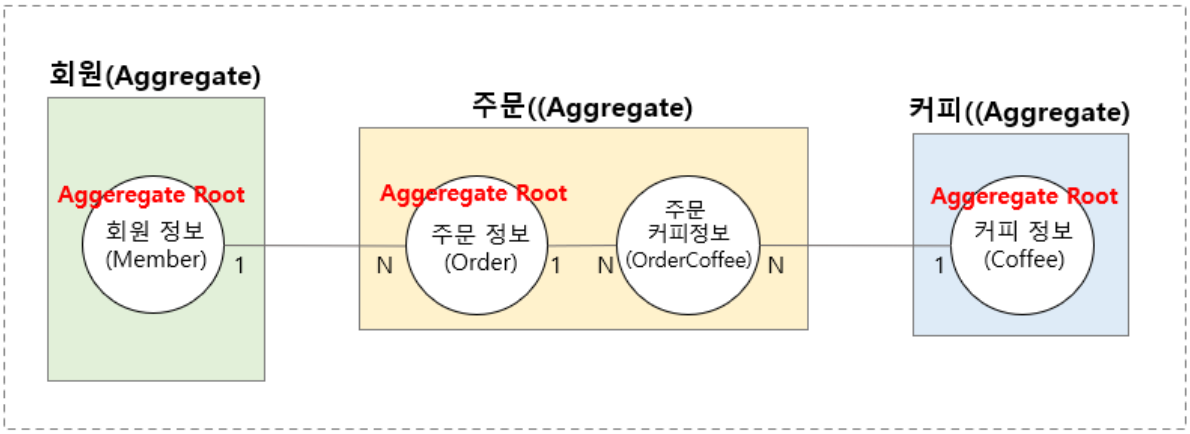
✅ 테이블 설계

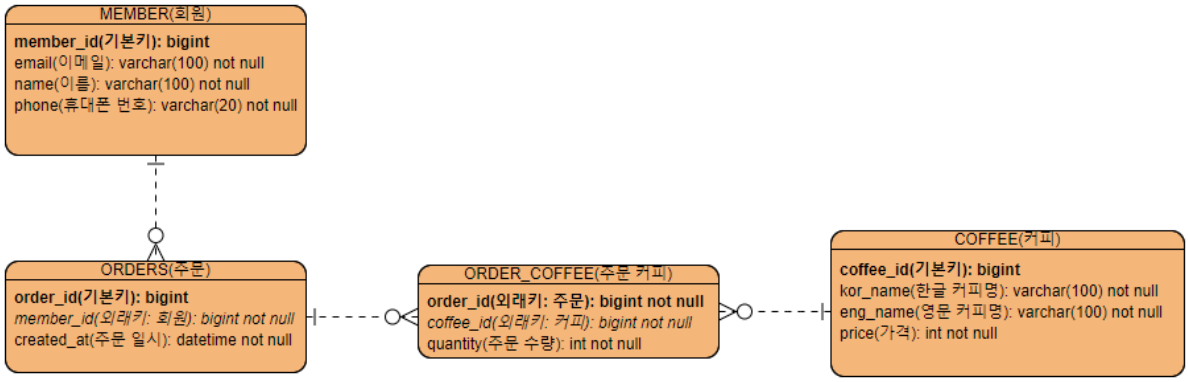


✅ 도메인 엔티티 설계(DDD가 아닌 일반적인 객체 간의 관계)



✅ 커피 주문 샘플 애플리케이션의 애그리거트 루트(Aggregate Root) 찾기





기능 구현

도메인 엔티티 클래스 정의

Member와 Order의 관계

Order와 Coffee의 관계

서비스, 리포지토리 구현

리포지토리(Repository) 인터페이스

- **쿼리 메서드 작성 방법**
- **네이티브 쿼리 작성 방법**

서비스 클래스

- 등록, 수정 시 verifyxxxx() 부분 설명
- Stream API 사용 설명
- Optional 사용 설명
  - `Optional.ofNullable()`
  - `Optional.orElseThrow()`

기타 주문 기능 수정으로 변경된 클래스

- 주문에 대한 설계가 대폭 수정되었으므로..

실습 과제 리뷰

- 페이지네이션 실습 과제 solution 코드 설명
-

✅ 페이지네이션 처리 방식

오프셋 방식

- 가져와야 되는 데이터까지 오프셋 개수 만큼 카운팅해서 찾아가는 방식
- 직접 찾아가야 되므로 시간이 좀 걸릴 수 있다.

- 100만 건 ~ 200만 건 정도에서 클라이언트가 심하게 느리다고 할 정도는 아님
- 쿼리 예

```
SELECT *
FROM MEMBER
ORDER BY MEMBER_ID DESC LIMIT 10 OFFSET 500_000;
```

커서 방식

- WHERE 절을 조건으로 마지막으로 조회한 부분 부터 탐색하므로 속도가 빠르다.
- 마지막 조회한 위치를 가지고 있어야 한다.
- 쿼리 예

```
SELECT *
FROM MEMBER
WHERE ID < {이 전에 조회한 마지막 MEMBEr_ID}  //<- 커서
ORDER BY MEMBER_ID DESC LIMIT 10;
```
