🥈

라이브 세션-Spring Data JDBC를 통한 데이터 액세스 계층 구현-2022.08.29(월)

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)

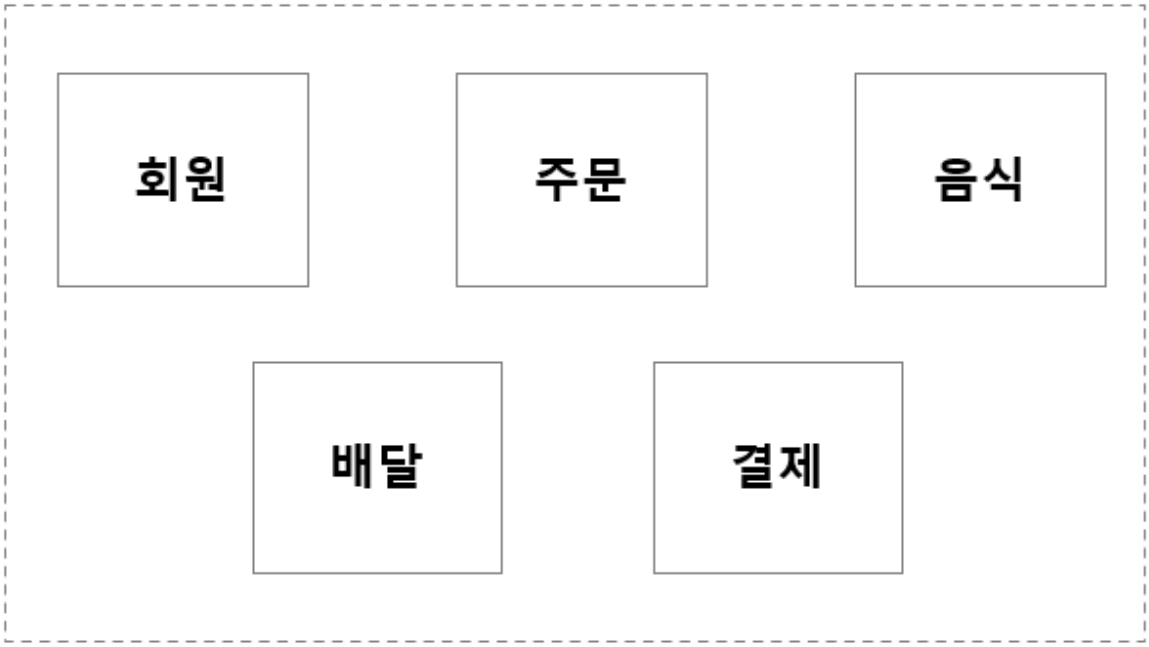
```java
@Getter
@Setter
@NoArgsConstructor
public class Member {
    @Id
    private Long memberId;

    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)

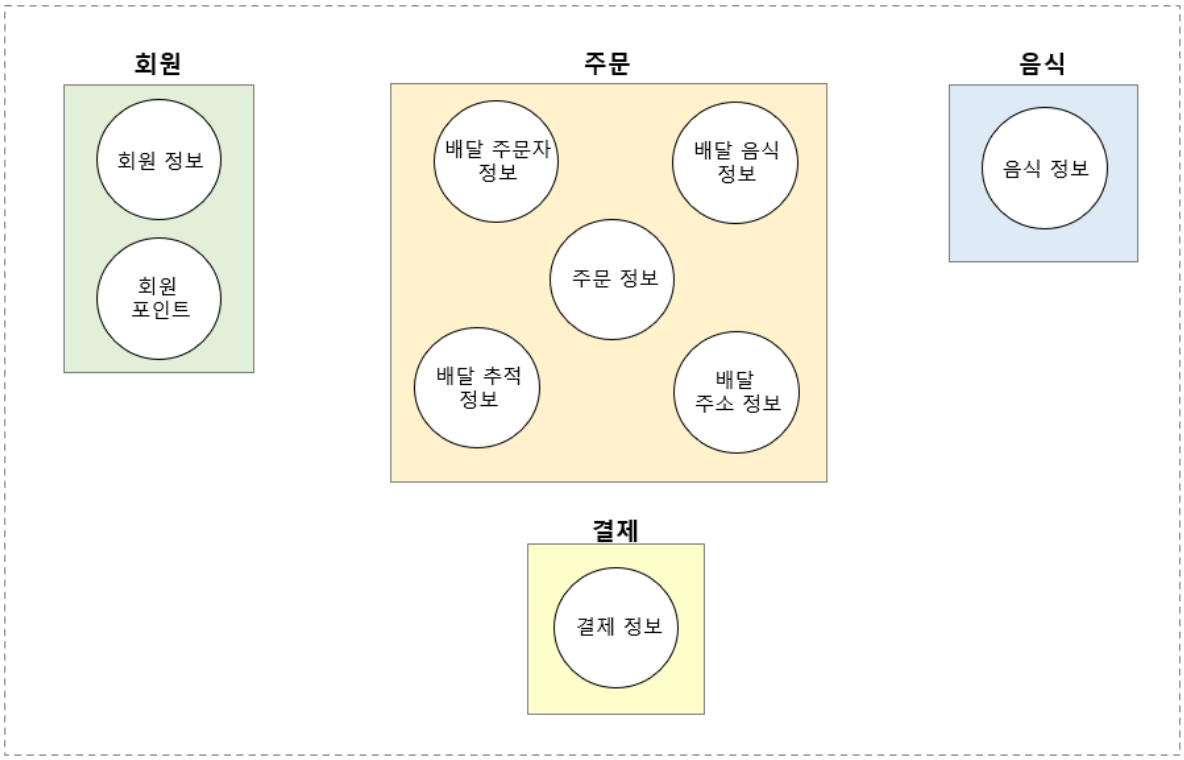
애그리거트(Aggregate)란?

비슷한 업무 도메인들의 묶음

배달 주문 앱의 도메인 모델 예



배달 주문 앱 도메인에서의 애그리거트(Aggregate)

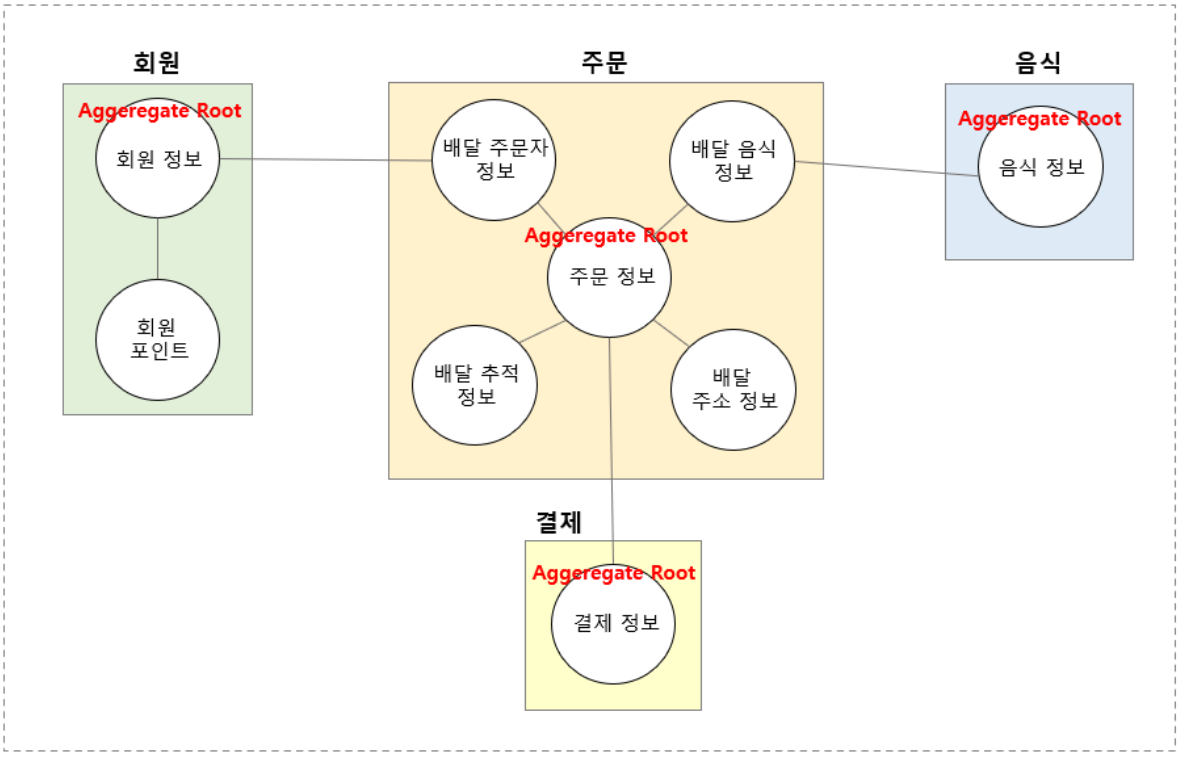


애그리거트 루트(Aggregate Root)

하나의 애그리거트를 대표하는 도메인

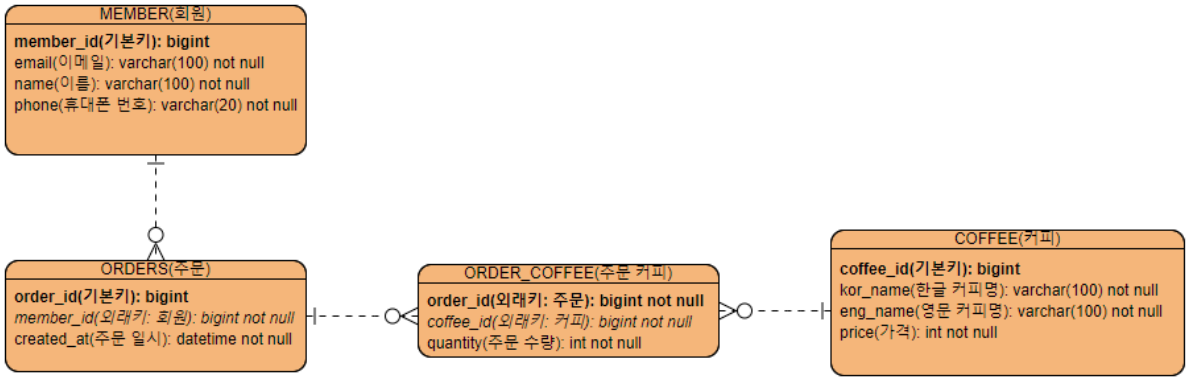
배달 주문 앱 도메인에서의 애그리거트 루트(Aggregate Root)

- 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.

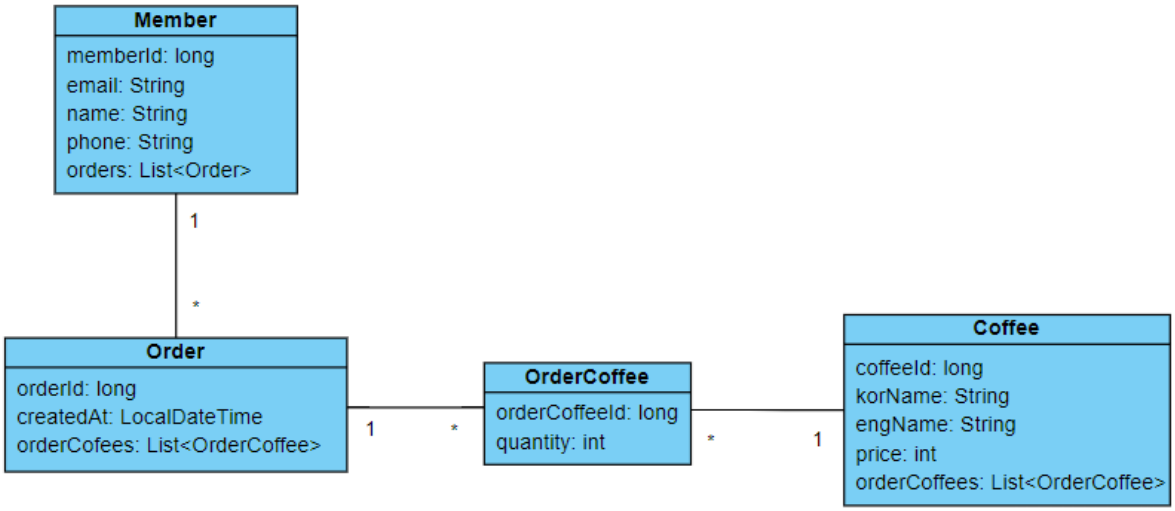


커피 주문 샘플 애플리케이션 테이블 및 도메인 엔티티 설계

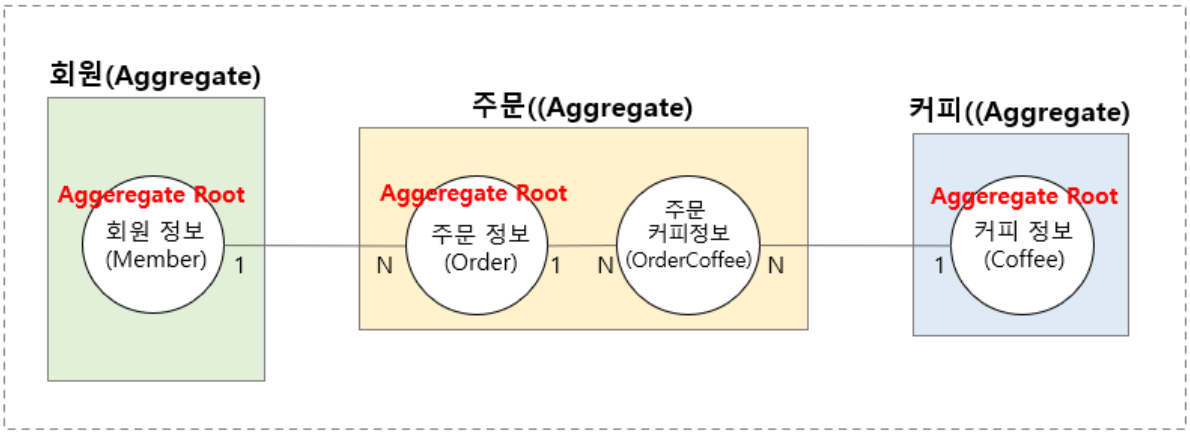
테이블 설계



도메인 엔티티 설계(DDD 적용 전)



커피 주문 샘플 애플리케이션의 애그리거트 루트(Aggregate Root) 찾기



설계 끝났으니 이제 구현입니다.

기능 구현

도메인 엔티티 클래스 정의

Member와 Order의 관계

- 1 대 N의 관계
- 둘 다 Aggregate Root
  - ID 참조

Order와 Coffee의 관계

- N 대 N의 관계
- 1 대 N, N 대 1의 관계로 재 설계
- 두 가지 관점에서 생각해야 한다.
  - CoffeeRef(ORDER_COFFEE)는 Order와 Coffee 사이에서 **AggregateRef 같은 역할**을 한다.

        - Order가 Coffee를 참조하기 위해 coffeeId를 가지는 참조 클래스의 역할을 한다.
    - **동일한 Aggregate 내에서 1 대 N의 관계**도 생각해야 됨.

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..

OrderController

OrderPostDto

OrderCoffeeDto

OrderCoffeeResponseDto

OrderResponseDto

OrderMapper

ExceptionCode