JPA ORM Chapter 13

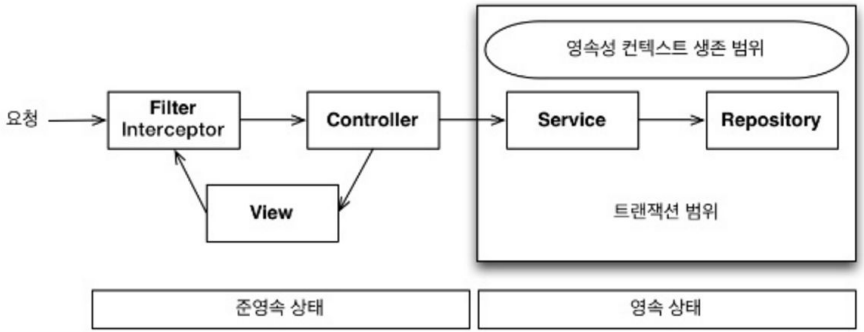
Bangjang

트랜잭션 범위의 영속성 컨텍스트

- 순수 Java 환경에서 JPA를 사용 시 개발자가 직접 엔티티 생성 / 트랜잭션 관리 필요
- Spring 또는 J2EE 컨테이너 환경에서 JPA 사용 시 컨테이너 제공 전략 따라야 함

Spring Container 기본 전략

- 기본 전략 - 트랜잭션 범위의 영속성 컨텍스트 전략
- 트랜잭션 시작 시 영속성 생성, 끝날 때 종료
- 같은 트랜잭션 안에서는 같은 영속성 컨텍스트에 접근
- 서비스 계층 @Transcational : 트랜잭션 AOP
- 트랜잭션 커밋 -> 영속성 Flush -> DB 커밋



준영속 상태와 지연 로딩

- 서비스 계층 종료 시점에 트랜잭션 및 영속성 컨텍스트 종료
- Controller, View 등의 Presentation 계층에서는 준영속 상태 : 변경 감지, 지연 로딩 X
- 준영속 상태의 지연 로딩 문제 해결법 : 엔티티 미리 로딩, OSIV 사용
- 미리 로딩의 3가지 방법
    - 글로벌 페치 전략 수정 / JPQL Fetch Join / 강제 초기화

글로벌 페치 전략 수정

- 글로벌 페치 전략을 Lazy에서 Eager로 변경
- 연관 엔티티를 항상 함께 로딩
- 사용하지 않는 엔티티 로딩
- N+1 문제 발생 : 글로벌 Fetch 전략이 아닌 JPQL 자체만 사용 -> 조회 성능에 치명적

JPQL 페치 조인

- SQL Join을 사용해 페치 조인 대상까지 함께 조회 : N+1 해결
- 메소드 증가, 논리적 의존관계 발생
- 최적화와 의존관계 사이에서 적절한 타협점을 찾는 것이 합리적

강제로 초기화

- 프레젠테이션 계층이 엔티티를 강제로 초기화 후 반환

- FACADE 계층을 추가함으로서 View를 위한 프록시 초기화를 담당함 : 서비스, 프레젠테이션 논리적 의존성 분리

준영속 상태와 지연 로딩의 문제점

- 엔티티 미리 초기화 -> 오류 발생 가능성
- 물리적 분리, 논리적으로는 의존
- 모든 문제는 엔티티가 프리젠테이션 계층에서 준영속 상태이기 때문에 발생
- 영속성 컨텍스트를 View까지 살아있게 열어두는 것 -> OSIV

OSIV

- Open Session In View
- 영속성 컨텍스트를 View까지 열어둠 -> View에서도 지연 로딩 사용 가능
- OSIV는 Hibernate 용어, JPA는 OEIV(Open EntityManager In View)

OSIV : 요청 당 트랜잭션

- 가장 단순한 구현
- Client 요청 -> 트랜잭션 시작 -> 같이 종료
- 프리젠테이션 계층이 엔티티를 변경 가능
- View 렌더링 후 트랜잭션 커밋 - 심각한 오류
- 읽기 전용 인터페이스, 엔티티 래핑, DTO만 반환

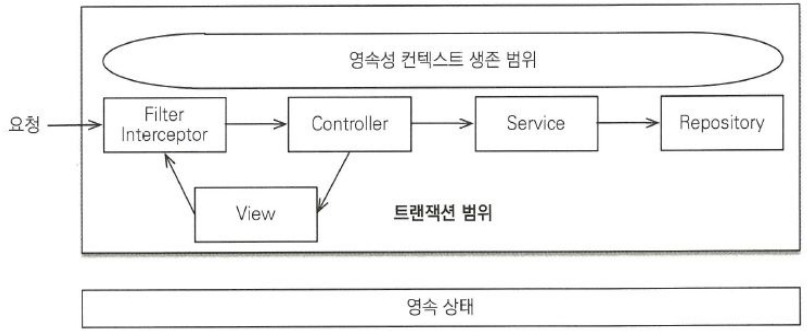


그림 13.6 초창기 OSIV: 요청당 트랜잭션

Spring OSIV : 비즈니스 계층 트랜잭션

- 요청 -> 영속성 생성 -> 서비스 시작 ->
  트랜잭션 & 영속성 사용 -> 서비스 종료 ->
  트랜잭션 삭제 -> 요청 종료 -> 영속성 종료
- 트랜잭션 없이 읽기 : 지연 로딩

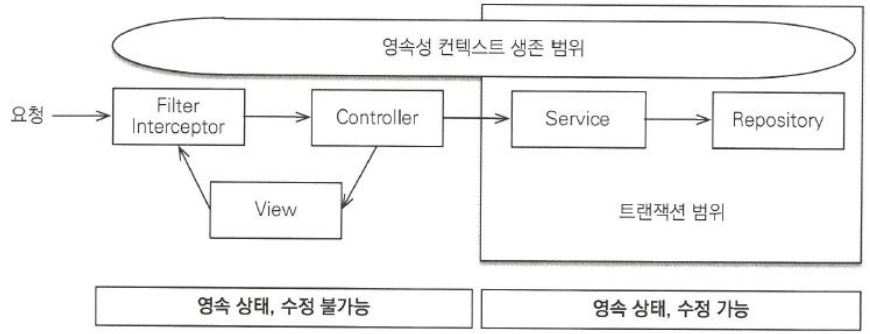


그림 13.7 스프링 OSIV – 비즈니스 계층 트랜잭션

Spring OSIV 주의사항

- 프리젠테이션 계층에서 엔티티 수정 직후 서비스 계층 호출시 문제 발생
- 거의 발생하지 않는 문제
- 같은 영속성 컨텍스트를 여러 트랜잭션이 공유하기 때문에 발생

OSIV 정리

- 한 요청의 생명 주기에 같은 영속성 컨텍스트 유지
- 엔티티의 수정은 트랜잭션이 있는 계층에서만 동작
- 같은 영속성 컨텍스트를 여러 트랜잭션이 공유하는 점 주의
- 지연 로딩에 의한 SQL - 성능 튜닝 시 확인해야 할 부분
- 대안으로는 FACADE / DTO
- 같은 JVM을 벗어난 원격 상황에서는 사용 불가능