Spring Boot 10 [OSIV와 성능 최적화]

실전! 스프링 부트와 JPA 활용 2 - API 개발과 성능 최적화

김영한

2022.08.16

OSIV와 성능 최적화

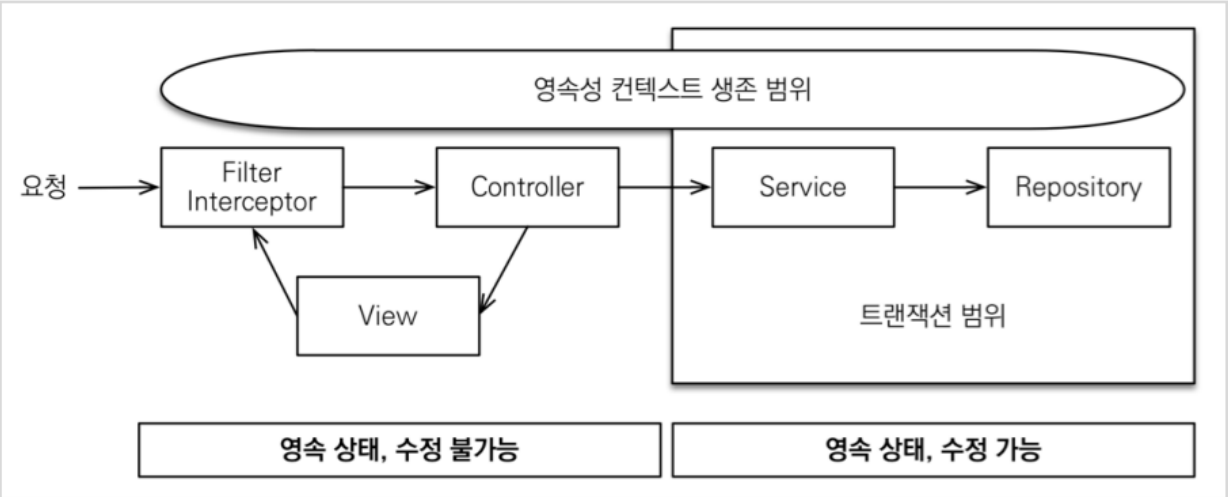
Open Session In View : 하이버네이트 **Open EntityManager In View :** JPA

관례상 OSIV 라고 하는 것.

OSIV ON `jpa.open-in-view : true`

기본적으로 스프링 프로젝트 실행시 WARN 로그를 남긴다.

```
2022-08-16 09:59:05.991  WARN 17364 --- [  restartedMain]
JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is
enabled by default. Therefore, database queries may be performed during
view rendering. Explicitly configure spring.jpa.open-in-view to disable
this warning
```



open-in-veiw 가 켜져 있다면 service등에서 트랜잭션이 실행되면 영속성 컨텍스트가 데이터 베이스 커넥션을 유지하게 된다. 종료는 어플리케이션에서 응답이 나갈 때 까지 그러기 때문에 지연로딩이 가능한 것, 지연로딩은 영속성 컨텍스트가 살아있어야 가능한다.

특징 이자 단점 - 커넥션 유지

open-in-veiw true 라면 DB 커넥션 데이터 베이스 커넥션 리소스를 사용하게 되고 커넥션이 모자를 수 있다.

OSIV OFF `jpa.open-in-view : false`

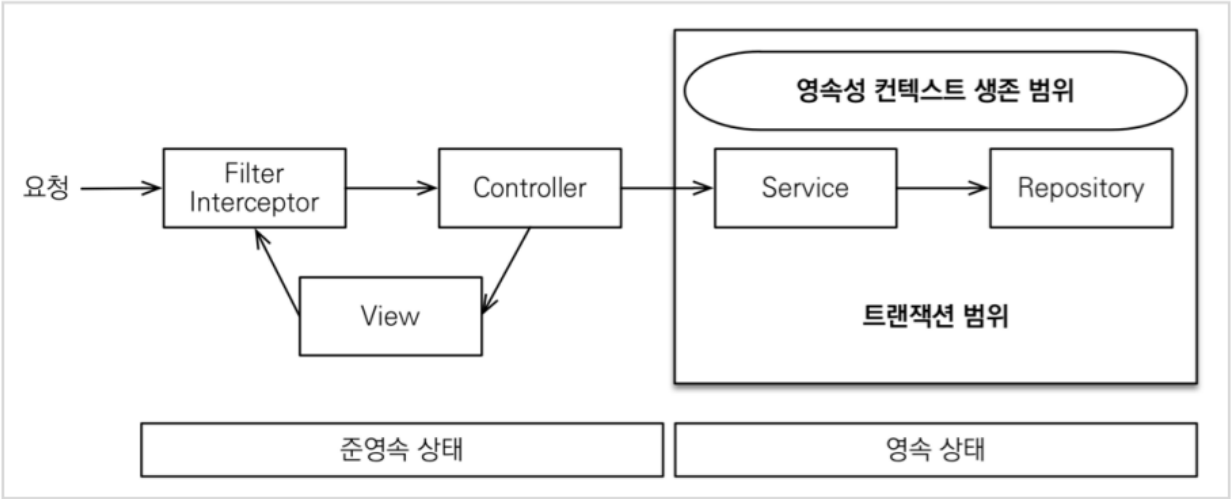


그림 과 같은 영속성 컨텍스트 생명 주기를 맞춘다. (트랜잭션 과)

트래픽이 많을 때 커넥션을 유연하게 사용 할 수 없다. 하지만 지연로딩이 동작하지 않기 때문에

트랜잭션이 종료되기 전에 지연로딩을 호출해 둬야 한다.

특징 이자 단점 - 영속성 컨텍스트가 사라짐

영속성 컨텍스트가 트랜잭션을 종료하기 전에 지연로딩을 사용해야 한다.

서비스 계층에서 사용하거나 fetch join을 사용해야한다.

컨트롤러에서 초기화 (지연로딩 에러)

```
"org.hibernate.LazyInitializationException: could not initialize proxy
[jpabook.jpashop.domain.Member#1] - no Session\r\n\tat
org.hibernate.proxy.AbstractLazyInitializer.initialize(AbstractLazyInitializer.java:176)\r\n
\tat org.hibernate.proxy.AbstractLazyInitializer.
```

@RestController

```
List<Order> all = orderRepository.findAllByString(new OrderSearch()); 트랜잭션 사용
for (Order order : all) {                                    트랙잭션 X 영속성컨텍스트X
   order.getMember().getName(); //강제 초기화
   order.getDelivery().getAddress();
   List<OrderItem> orderItems = order.getOrderItems();
   orderItems.stream().forEach(o -> o.getItem().getName());
}
```

[[의문점]]

Controller에서 특정 Mapping에 트랜잭션을 가지는것은 해결 방법이 될수 없는가?

-단순 문제 해결이라고 생각한다. 그냥 쿼리용 서비스로 빼는게 더 좋다

OSIV OFF 지연로딩 해결법

- 새로운 서비스를 만들어서 사용한다.

  쿼리와 커맨드를 분리한다**.**

  성능 이슈는 대부분 조회에서 문제가 발생한다. 핵심 비즈니스에 큰 영향을 주지는 않지만 복잡한 조회용 로직이 핵심 비즈니스 로직이 같이 존재하게된다면 유지보수가 힘들어진다. 보통 서비스 계층에서 트랜잭션을 유지한다.

- + 여러 방법이 존재.

스프링 데이터 **JPA** 간단소개

문서 경로

Spring-> project-> spring data -> spring Data JPA

중복되는 코드를 자동화

```
public void save(Member member){
   em.persist(member);
}

public Member findOne(Long id){
   return em.find(Member.class, id);
}

public List<Member> findAll(){
   return em.createQuery("select m from Member m", Member.class)
            .getResultList();
}
```

⇒일반적인 Repository의 반복되는 저장 하나 조회, 전체 조회

```
public interface MemberRepository extends JpaRepository<Member, Long> {

   //select m from Member m where m.name = ? 이라고 짜준다.
   List<Member> findByName(String name);
}
```

라고 짜도 동작하게 된다. 물론 위에 조회,저장, 전체 조회도 포함되어있다.

=마법 같다!

QueryDSL 간단 소개

http://ww.querydsl.com

오픈 소스 프로젝트

build.gradle 수정

```
buildscript {
  ext {
     queryDslVersion = "5.0.0"
  }
}

plugins {
  id 'org.springframework.boot' version '2.7.2'
  id 'io.spring.dependency-management' version '1.0.12.RELEASE'
  id 'java'
  id "com.ewerk.gradle.plugins.querydsl" version "1.0.10"
}

//apply plugin: "com.ewerk.gradle.plugins.querydsl"

group = 'jpabook'
version = '0.0.1-SNAPSHOT'
sourceCompatibility = '11'

configurations {
  compileOnly {
     extendsFrom annotationProcessor
  }
}

repositories {
  mavenCentral()
}

dependencies {
  implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
  implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'
  implementation 'org.springframework.boot:spring-boot-starter-web'
  implementation 'org.springframework.boot:spring-boot-starter-validation'
  implementation 'org.springframework.boot:spring-boot-devtools'
  implementation 'com.github.gavlyukovskiy:p6spy-spring-boot-starter:1.5.6'

  implementation 'com.fasterxml.jackson.datatype:jackson-datatype-hibernate5'


  compileOnly 'org.projectlombok:lombok'
  runtimeOnly 'com.h2database:h2'
  annotationProcessor 'org.projectlombok:lombok'
  testImplementation 'org.springframework.boot:spring-boot-starter-test'

  //querydsl 추가
```

```
  implementation 'com.querydsl:querydsl-jpa'
  //querydsl 추가
  implementation 'com.querydsl:querydsl-apt'
}

tasks.named('test') {
  useJUnitPlatform()
}

def querydslDir = 'src/main/generated'

querydsl {
  library = "com.querydsl:querydsl-apt"
  jpa = true
  querydslSourcesDir = querydslDir
}
sourceSets {
  main {
     java {
        srcDirs = ['src/main/java', querydslDir]
     }
  }
}
compileQuerydsl{
  options.annotationProcessorPath = configurations.querydsl
}
configurations {
  compileOnly{
     extendsFrom annotationProcessor
  }
  querydsl.extendsFrom compileClasspath
}
```