

스프링 스터디 3주차 (2021.08.29)

스프링 입문 - 코드로 배우는 스프링 부트, 웹 MVC, DB 접근 기술 (완료)

- 섹션 6. 스프링 DB 접근 기술
- 섹션 7. AOP

👉 Spring의 다양한 DB 접근 기술(JDBC, JDBC Template, JPA, 스프링 데이터 JPA)들을 발전 순서대로 적용해 보았고, AOP기술을 적용해 모든 메소드의 호출 시간을 처리해 보았다!

스프링 핵심 원리 - 기본편

- 섹션 1. 객체 지향 설계와 스프링

👉 스프링의 역사와 스프링의 다양한 기술과 유래, 좋은 객체 지향 설계의 5가지 원칙(SOLID)에 대해 개념적으로 알아보았다!

번외: 자바 EE와 스프링과의 관계

<https://okky.kr/article/415474>

DB 트랜잭션?? @Transactional??

스프링 부트 통합 테스트 코드를 작성 할 때, AfterEach와 BeforeEach를 지우고 **@Transactional**이라는 어노테이션을 활용한다.



JPA~ DB에서 트랜잭션이 항상 있어야해요~
데이터 변경이 트랜잭션 안에서 수행 되어야 합니다~~

트랜잭션이 무엇인지, **@Transactional**은 어떤 역할을 하는지 간단하게 조사해 보자!

트랜잭션 Transaction



트랜잭션은 하나의 작업을 수행하기 위해 필요한 데이터베이스의 연산들을 모아 놓은 것으로, 데이터베이스에서 논리적인 작업의 단위이며 장애가 발생했을 때 데이터를 복구하는 작업의 단위이다.

트랜잭션이라는 것을 관리함으로써 데이터베이스의 회복과 병행 제어가 가능!

트랜잭션(SQL문들의 모임)의 모든 명령문이 완벽하게 처리되거나 하나도 처리되지 않아야 데이터 베이스 모순이 없는 일관된 상태를 유지한다.

- 온라인 쇼핑몰에서 계좌 이체 사례
 1. 내 계좌에서 2만원을 인출 및 이체한다.
 2. 내 계좌에 잔액이 차감 된다.
 3. 업체에 내 이름으로 2만원이 입금된다.
 4. 업체 계좌에 잔액이 더해진다.



만약, 시스템 문제로 내 계좌는 2만원이 빠져나갔는데, 업체는 돈을 받지 못한 상황이라면??

트랜잭션의 특성을 활용해 진행 중이던 작업을 취소하고 처음부터 다시 진행할 수 있다.

트랜잭션의 4가지 특징(ACID)



- 'A' tolicity - 원자성
- 'C' onsistency - 일관성
- 'I' solation - 독립성
- 'D' urability - 지속성

원자성 (Atomicity)

트랜잭션을 구성하는 연산들이 모두 정상적으로 실행되거나 하나도 실행되지 않아야 한다는 All-or-Nothing 방식을 의미!

ex) 내 계좌에서 2만원이 빠져나갔으면 업체에서도 돈을 받아야 한다. (돈을 받지 못하면 안 된다.)

일관성 (Consistency)

트랜잭션이 성공적으로 수행된 후에도 데이터베이스가 일관성 있는 상태를 유지해야 함을 의미!

ex) 계좌 이체 진행 시 진행 전의 돈의 합과 진행 후의 돈의 합이 같아야 한다.

독립성 (Isolation)

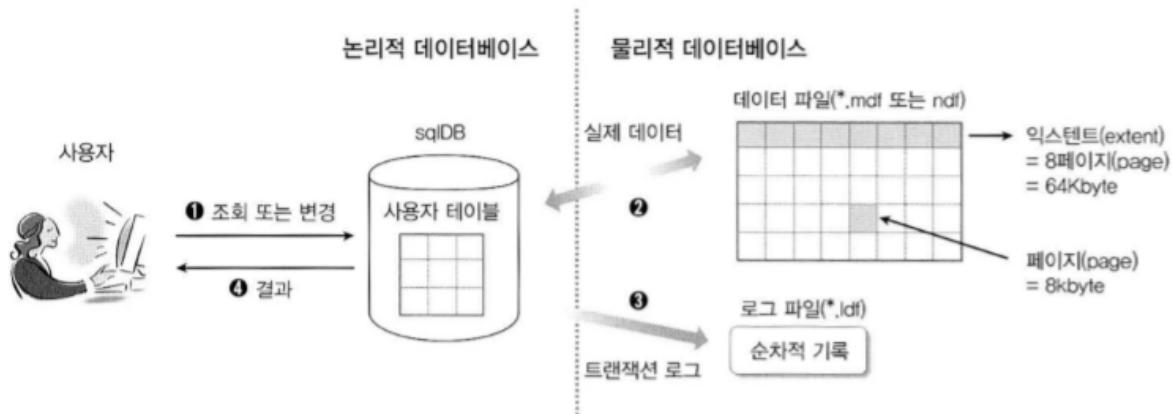
현재 수행 중인 트랜잭션이 완료될 때까지 트랜잭션이 생성한 중간 연산 결과에 다른 트랜잭션들이 접근할 수 없음을 의미!

ex) 내가 먼저 돈을 보내고 있는데 다른 사용자가 중간에 끼어들어서 돈을 보낼 수 없다.
하지만, 성능 관련 이유로 유연성 있는 제약 조건!

지속성 (Durability)

트랜잭션이 성공적으로 완료된 후 데이터베이스에 반영한 수행 결과는 어떠한 경우에도 손실되지 않고 영구적이어야 함을 의미!

트랜잭션은 로그로 남고 시스템 장애 발생 전 상태로 되돌릴 수 있다.



[그림 10-2] 데이터베이스의 간단한 구조

@Transactional



스프링에서 제공하는 트랜잭션 처리 어노테이션이다.

@Transactional을 메서드 또는 클래스에 명시하게 되면 특정 메서드 또는 클래스가 제공하는 모든 메서드에 대해 내부적으로 AOP를 통해 트랜잭션 처리 코드가 전후로 수행된다.

AOP는 일반적으로 두 가지 방식이 있다.

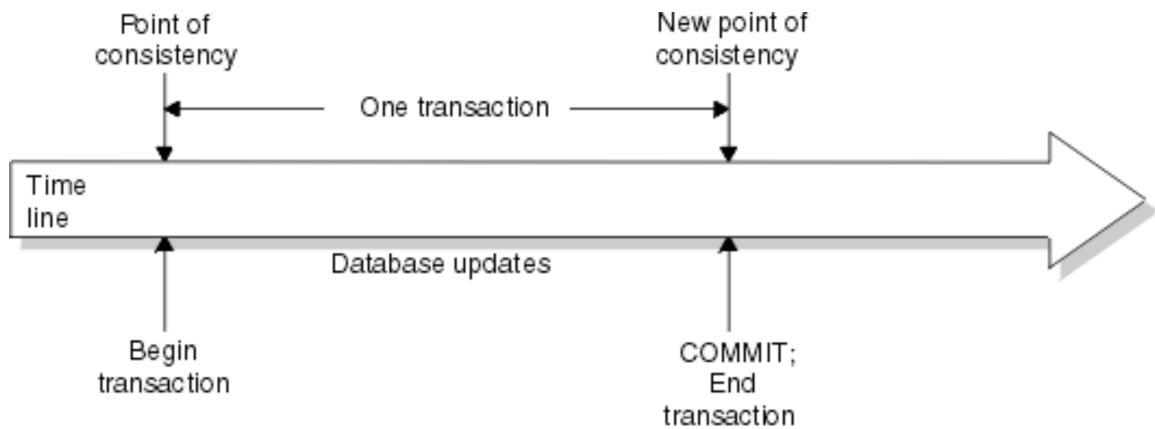
1. JDK Dynamic Proxy 방식
2. CGLib 방식

Commit과 Rollback



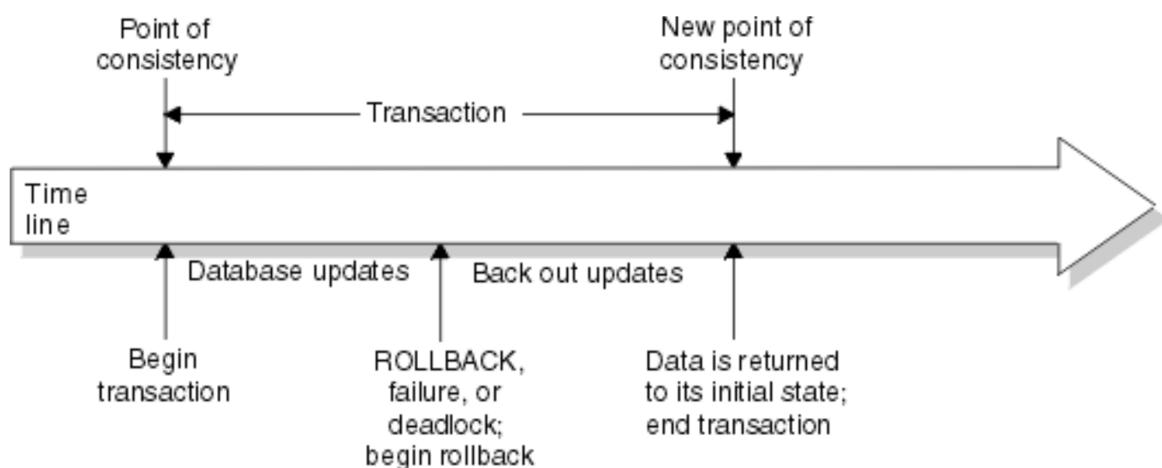
보통 DB를 사용할 때, 쿼리를 날리면 데이터 베이스에 반영이 된다고 생각하는데, DB에 반영되는 시점은 트랜잭션 연산이 성공적으로 완료되는 시점에 실제 데이터 베이스에 반영이 된다. 그리고 트랜잭션 연산에는 크게 두 가지 과정이 있다.

Commit 연산 (트랜잭션이 성공적으로 수행 → 작업 완료)



Commit 연산의 실행을 통해 트랜잭션의 수행이 성공적으로 완료되었음을 선언하고 결과를 최종 데이터베이스에 반영.

Rollback 연산 (트랜잭션이 수행을 실패 → 작업 취소)



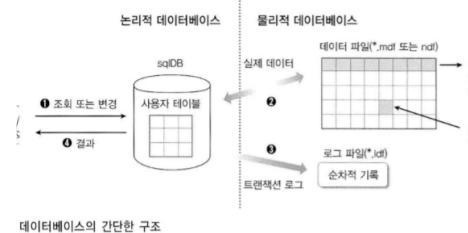
Rollback 연산이 실행되면 트랜잭션이 지금까지 실행한 연산의 결과가 취소되고 트랜잭션이 수행되기 전의 상태로 다시 돌아간다.

Ref.

트랜잭션(Transaction)의 4가지 특성 ACID

트랜잭션은 데이터베이스 내에서 데이터를 처리하는 작업 수행의 논리적인 작업 단위를 (하나의 그룹을) 의미한다. 예를 들어 온라인 쇼핑을 있다고 가정해보자. 주문 완료를 위해 쇼핑몰 업체에 계좌이체를 하

☞ <https://hyoni-k.tistory.com/55>



[DataBase] DB를 지탱하는 트랜잭션

데이터 베이스는 우리가 흔히 사용하는 파일 시스템과는 달리 기본적으로 4가지 특징이 존재한다. 1) 실시간 접근성 2) 계속적인 변화 3) 동시 공유 4) 내용에 따른 참조 이외에도 디비의 장점이라고 볼 수 있
☞ <https://brunch.co.kr/@skeks463/27>



@Transactional

스프링에서 제공하는 트랜잭션 처리 중 하나이다. 애노테이션으로 트랜잭션 처리를 지원한다. 선언적 트랜잭션이라고도 부른다.
@Transactional은 Spring AOP를 기반으로 동작한다. 프록시 객체

☞ <https://velog.io/@pond1029/transactional>

velog

@Transactional 동작 원리

해당 포스팅은 Spring boot 2.2.0.RELEASE 환경에서 진행됐다. 비즈니스로직이 트랜잭션 처리를 필요로 할 때 트랜잭션 처리 코드가 비즈니스 로직과 공존한다면 코드 중복이 발생하고 비즈니스 로직에 집

☞ <https://hwannny.tistory.com/98>

