AOP 프로그래밍 1

Aspect Oriented Programming의 약자로, 여러 객체에 공통으로 적용할 수 있는 기능을 분리해서

재사용성을 높여주는 프로그래밍 기법

준비사항

- aspectjweaver 모듈은 AOP를 설정하는데 필요한 애노테이션을 제공하므로 이 의존을 추가한다.

<dependency>
<groupId>org.aspectj</groupId>
<artifactId>aspectjweaver</artifactId>
<version>1.8.13</version>
</dependency>

프록시

- 기존 코드를 수정하지 않고 코드 중복도 피할 수 있는 방법
- 핵심 기능의 실행은 다른 객체에 위임하고 부가적인 기능을 제공하는 것

  (핵심 기능은 구현하지 않음)
- 실제 핵심 기능을 실행하는 객체는 대상 객체라고 부른다.

AOP 핵심 기능에 공통 기능 삽입 방법

- 컴파일 시점에 코드에 공통 기능을 삽입하는 방법 (스프링에서 지원 x)
- 클래스 로딩 시점에 바이트 코드에 공통 기능을 삽입하는 방법 (스프링에서 일부 지원)
- 런타임에 프록시 객체를 생성해서 공통 기능을 삽입하는 방법 (스프링에서 지원 o)

AOP 주요 용어

- Aspect : 공통 기능 ex) 트랜젝션, 보안
- Advice : 언제 공통 관심 기능을 핵심 로직에 적용할 지를 정의
- Joinpoint : Advice를 적용 가능한 지점을 의미 (메서드 호출, 필드 값 변경 등
  But 스프링은 메서드에 대한 Joinpoint만 지원)
- Pointcut : Joinpoint의 부분 집합으로서 실제 Advice가 적용되는 JoinPoint를 말함
- Weaving : Advice를 핵심 로직 코드에 적용하는 것을 뜻함

스프링에서 구현 가능한 Advice의 종류

- Before Advice : 대상 객체의 메서드 호출 전
- After Returning Advice : 대상 객체의 메서드가 익셉션 없이 실행된 이후
- After Throwing Advice : 대상 객체의 메서드를 실행하는 도중 익셉션이 발생한 경우
- After Advice : 익셉션 발생 여부에 상관 없이 대상 객체의 메서드 실행 후
- Around Advice : 대상 객체의 메서드 실행 전, 후 또는 익셉션 발생 시점

스프링 AOP 구현

- Aspect로 사용할 클래스에 @Aspect 애노테이션을 붙인다.
- @Pointcut 애노테이션으로 공통 기능을 적용할 Pointcut 정의
- 공통 기능을 구현한 메서드에 @Around 애노테이션 적용