AOP 프로그래밍 3

풍부한 서식의 문서를 신속하게 작성해 보세요.

프록시 생성 방식

- AOP를 위한 프록시 객체를 생성할 때 실제 생성할 빈 객체가 인터페이스를 상속하면

인터페이스를 이용해서 프록시를 생성한다.

- 빈객체가 인터페이스를 상속할 때 인터페이스가 아닌 클래스를 이용해서 프록시를
- 생성하고 싶다면 @EnableAspectJAutoProxy(proxyTargetClass = true)로 설정하면 된다.

proxyTargetClass = true로 지정하면 인터페이스가 아닌 자바 클래스를 상속받아

프록시를 생성한다.

execution 명시자 표현식

- 수식어패턴은 생략 가능하며 public, protected 등이 온다. (스프링AOP는 public만 지원)
- 리턴타입패턴은 리턴타입을 명시
- 클래스이름패턴과 메서드이름패턴은 클래스 이름 및 메서드 이름을 패턴으로 명시
- 파라미터패턴은 매칭될 파라미터에 대해서 명시
- *을 이용하여 모든값을 표현, ..을 이용해서 0개 이상이라는 의미를 표현
- ex)

예	설명
execution(public void set*(..))	리턴 타입이 void이고, 메서드 이름이 set으로 시작하고, 파라미터가 0개 이상인 메서드 호출

예	설명
execution(* chap07.*.*())	chap07 패키지의 타입에 속한 파라미터가 없는 모든 메서드 호출
execution(* chap07..*.*(..)	chap07 패키지 및 하위 패키지에 있는, 파라미터가 0개 이상인 메서드 호출
execution(Long chap07.Calculator.factorial(..))	리턴타입이 Long인 Calculaotor 타입의 factorial() 메서드 호출
execution(* get*(*))	이름이 get으로 시작하고 파라미터가 한개인 메서드 호출
execution(* get*(*, *))	이름이 get으로 시작하고 파라미터가 두개인 메서드 호출
execution(* read*(Integer, ..))	메서드 이름이 read로 시작하고, 첫 번째 파라미터 타입이 Integer이며, 한개 이상의 파라미터를 갖는 메서드 호출

Advice 적용 순서

- 어떤 Aspect가 먼저 적용될지는 스프링 프레임 워크나 자바 버전에 따라 달라질 수 있다.
- @Order 애노테이션으로 적용 순서를 정할 수 있다
- @Order 애노테이션의 값이 작으면 먼저 적용하고 크면 나중에 적용한다.

```
@Aspect
@Order(1)  //숫자가 작으므로 선 적용
public class example1(){
}

@Aspect
@Order(2) //숫자가 크므로 후 적용
public class example2(){
}
```