SPRING AOP 간단 정리

1. 목적 및 특징
2. 용어
3. Advice의 종류
4. AspectJ 표현식
5. 코드 적용

목적 및 특징

- 특정한 메서드의 호출 전이나 후에 공통적인 처리가 필요할 때
- 관심의 분리(공통 관심기능과 핵심 관심기능)
- 공통관심기능 : 전체 시스템에서 사용되는 기능(로그처리, 트랜잭션, 인증, ...)
- 핵심관심기능 : 일반 업무 프로세스(게시글 등록, 수정, 삭제, 조회 등등…)
- 핵심관심기능은 기존의 OOP 기술로 쉽게 분리가 가능하나 횡단관심은 기존의 OOP 기술로 분리가 쉽지 않음
- OOP로 처리하기 까다로운 부분을 AOP를 도입하여 손쉽게 공통 기능을 추가, 수정, 삭제 할 수 있다.
- Spring AOP는 메서드 단위 처리를 지원
- Runtime 시점에 공통 기능을 적용

용어

이름	설명
JoinPoint	공통 기능이 삽입되어 동작될 수 있는 위치
PointCut	조인포인트 중에서 실행 하려는 위치를 결정
Advice	실제로 적용하는 기능(로그, 트랜잭션, 인증)으로 적용하는 시점을 가지고 있다..
Target	공통기능이 적용될 대상 클래스 객체
Weaving	Advice를 비즈니스 로직 코드에 삽입하는 것
Aspect	여러클래스나 기능에 사용되는 관심사를 모듈화함(포인트 컷과 어드바이스를 합쳐 놓은 것), 예> 트랜잭션 관리 기능

Advice 종류

이름	설명
@Before	미리 정의한 Pointcut 진입전에 수행
@AfterReturning	미리 정의한 Pointcut에서 return이 발생한 후에 수행
@AfterThrowing	미리 정의한 Pointcut에서 예외가 발생할 경우 수행
@After	미리 정의한 Pointcut에서 예외 발생 여부와 상관없이 무조건 수행
@Around	미리 정의한 Pointcut의 전,후에 필요한 동작을 수행

명시자 : execution

execution(AspectJ 표현식)
execution(제한자패턴? 리턴타입패턴? 이름패턴(파라미터패턴))
? - 생략가능

AspectJ 표현식

와일드카드	연산자
*	1개의 모든 값을 표현 매개변수 사용(1개의 파라미터) 패키지 사용(1개의 패키지)
..	0개 이상의 값을 표현
+	주어진 타입의 자식 클래스나 인터페이스를 표현

AspectJ 표현식 적용 예시

사용	설명
execution(public * *(..))	public 메소드 선택
execution(* set*(..))	이름이 set으로 시작하는 모든 메소드
execution(* get*(..))	이름이 get으로 시작하는 모든 메소드
execution(* main(..))	이름이 main인 메소드
execution(* com.ssafy..(..))	com.ssafy 패키지에 있는 클래스의 모든 메서드

AspectJ 표현식

사용 설명
execution(* com.ssafy.prj.model.UserService.*(..))
UserService 모든 메소드
execution(* com.ssafy.prj.model.service..(..))
service 패키지와 하위 패키지의 모든 메소드 실행

API 클래스

사용	주요메서드
JoinPoint	Object getTarget() 핵심기능 클래스 객체 Signature getSignature() 핵심기능 클래스의 메서드
ProceedingJointPoint	Signature getSignature() 핵심기능 클래스의 메서드 proceed() 핵심기능 클래스의 메서드 실행

어노테이션을 이용한 AOP 적용

- XML 설정부분을 간소화 하기 위해 어노테이션을 활용하는 방식
- XML 문서에 어노테이션을 활용하기 위한 선언 필요
  <aop:aspectj-autoproxy />
- 클래스 위쪽에 @Aspect 선언
- 메서드 위쪽에 어드바이스 선언

XML 적용 코드

```
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <context:component-scan base-package="com.ssafy.aop" />
    <aop:aspectj-autoproxy />
</beans>
```

Java 적용 코드

```
@Aspect
@Component
public class TimeAspect {
    @Around("execution(public * com.ssafy.aop..*Controller.list(..))")
    public Object executeTime(ProceedingJoinPoint pjp) throws Throwable {
        ...
        pjp.proceed();
        ...
    }
}
```