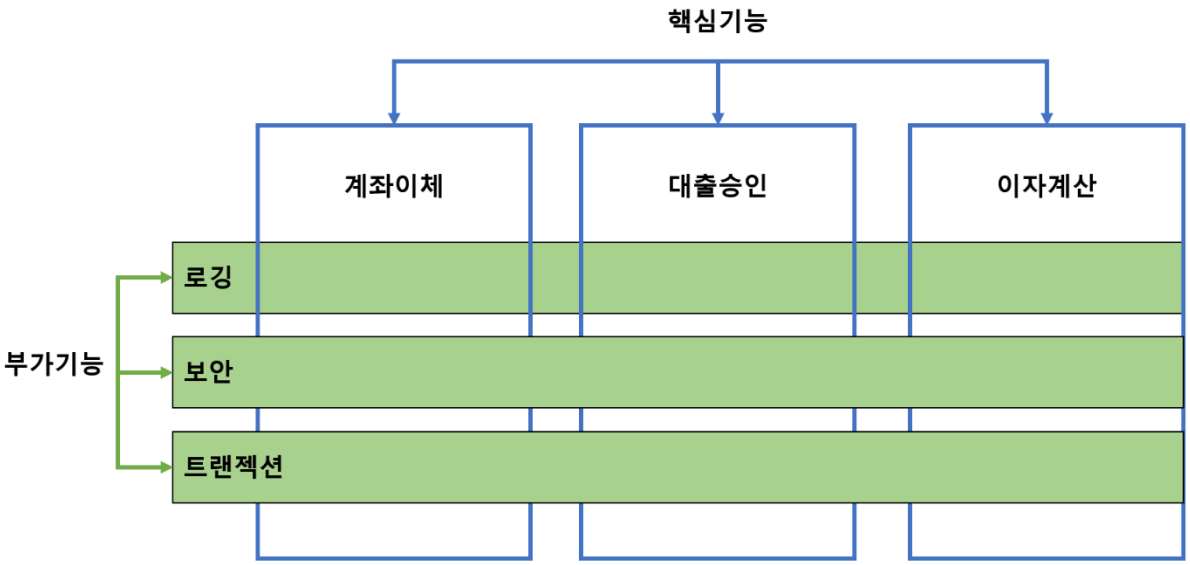
AOP (Aspect Oriented Programming) 기본

< 1. AOP란? >

OOP를 보완하는 수단으로, 여러 곳에서 쓰이는 공통 기능을 모듈화하고, 쓰이는 곳에 필요할 때 연결함으로써, 유지 보수 혹은 재상용에 용이하도록 프로그래밍하는 것이다. 중복되는 코드 제거, 효율적인 유지보수, 높은 생산성, 재활용성 극대화, 변화 수용이 용이하다는 장점들이 있다.

< 2. AOP와 관련된 용어 >



1) Aspect(애스펙트)

Advice(어드바이스)와 PointCut(포인트컷)을 합친 개념으로, 분리한 부가기능을 모듈형태로 만들어 놓은 것이다. AOP의 기본 모듈이며 싱글톤 형태의 객체로 존재한다.

2) Advice(어드바이스)

부가기능을 정의한 코드 즉, Target(타겟)에 제공할 부가기능을 담고 있는 모듈이다. 위의 그림에서 로깅 / 보안/ 트랜젝션이 Advice(어드바이스)에 해당된다

3) PointCut(포인트 컷)

Advice(어드바이스)를 적용할 Target(타겟)의 메서드를 선별하는 정규표현식으로 쉽게 말해 부가기능을 어떤 핵심기능에 적용할지를 결정하는 것이라고 보면 된다.

4) Target(타겟)

핵심 기능을 담고 있는 모듈로 Target(타겟)은 부가기능을 부여할 대상이 된다. 위의 그림에서 계좌이체 / 대출승인 / 이자계산이 Target(타겟)에 해당된다.

5) JoinPoint(조인포인트)

Advice(어드바이스)가 적용될 수 있는 위치를 말한다. Target(타겟) 객체가 구현한 인터페이스의 모든 메서드는 JoinPoint(조인포인트)가 된다.

6) Weaving(위빙 -> 짜집기)

PointCut(포인트 컷)에 의해서 결정된 Target(핵심기능)의 JoinPoint(조인포인트)에 Advice(부가/공통 기능)를 삽입하는 과정을 의미한다. Weaving(위빙)은 AOP가 Target(핵심기능)의 코드에 영향을 주지 않으면서 필요한 Advice(부가/공통 기능)를 추가할 수 있도록 해주는 핵심적인 처리과정이다.

< 3. Spring AOP의 특징 >

1) Spring은 Proxy(프록시) 기반 AOP를 지원한다

- 프록시는 Advice(부가/공통 기능)을 Target(핵심기능) 객체에 적용하면서 생성되는 객체
- Target(핵심기능)을 감싸는 프록시는 실행시간(RunTime)에 생성

2) 프록시는 호출을 가로챈다

- 프록시는 Target(핵심기능) 객체에 대한 호출을 가로챈 다음 Advice(부가/공통 기능)의 로직을 수행하고 난 뒤에 Target(핵심기능) 로직을 호출한다. 이를 Before Advice라고 함.
- 또는 Target(핵심기능) 로직을 호출한 뒤에 Advice(부가/공통 기능)를 수행하는 경우도 있다. 이를 After Advice라고 함.

3) Spring AOP는 메서드 JoinPoint만 지원한다.

- Target(핵심기능)의 메서드가 호출되는 RunTime 시점에만 Advice(부가/공통 기능)을 적용할 수 있다.