AOP

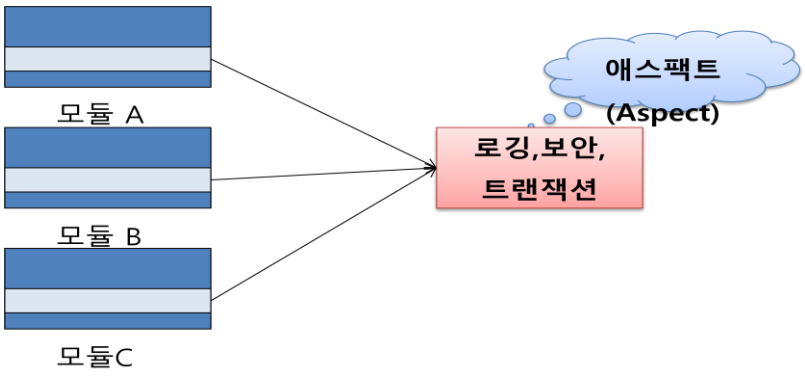
4-1. 관점지향 프로그래밍

관점지향 프로그래밍

AOP(Aspect Oriented Programming)

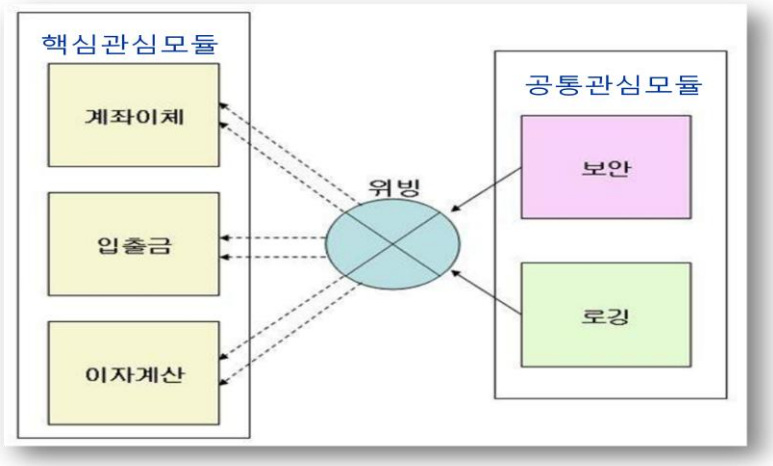
Aspect : 기존 객체지향 프로그래밍에서 모듈화 기법으로 추가된 중복코드(공통 처리 부분)를 별도의 독립된 클래스(Aspect)로 만들어 놓은 것

각 모듈이 실행될 때 Aspect를 위임해서 실행되게 하는 프로그래밍 방식



핵심관심모듈 vs 공통관심모듈(횡단관심모듈)

핵심관심모듈은 개발자가 관심을 갖고 개발할 대상의 애플리케이션을 의미하며, 공통관심모듈은 핵심관심모듈에서 반복되는 코드 부분을 별도의 클래스로 구현한 모듈을 의미한다.

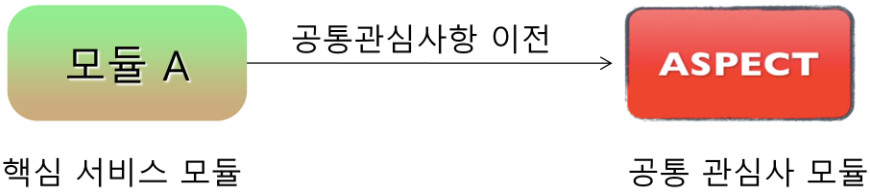


관점 지향 프로그래밍 이점

전체 애플리케이션에 흩어져 있는 관심사항이 하나의 장소로 응집

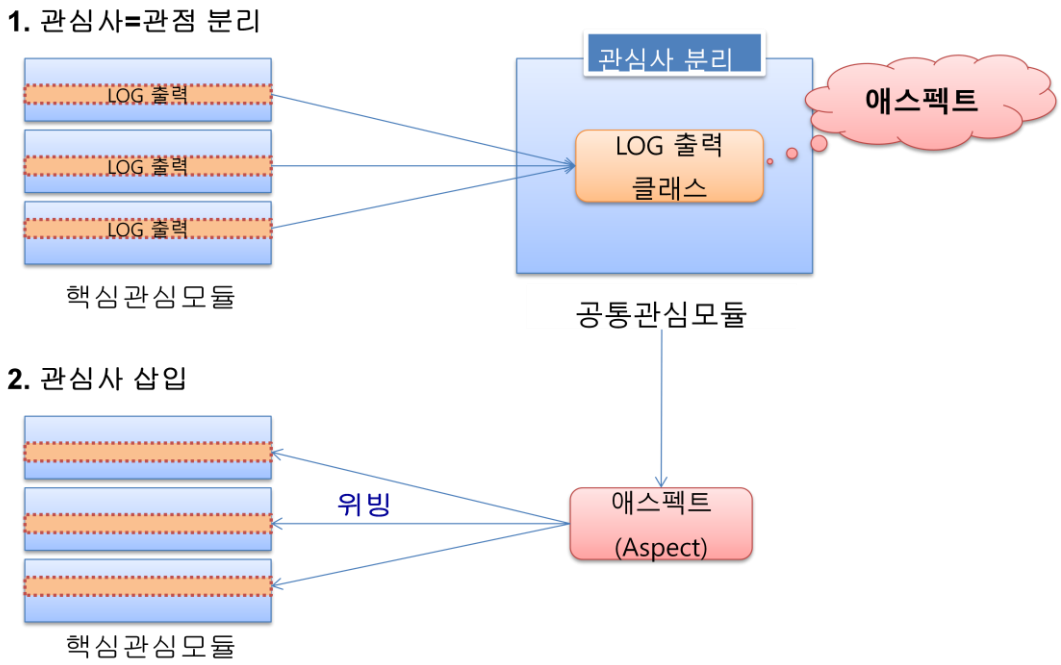
일반 서비스 모듈에서 주요 부분에 집중할 수 있음

공통 관심사는 애스펙트로 옮겨지기 때문에 코드의 생산성, 재사용성(↑)



Spring AOP 예

Spring에서는 핵심 서비스 모듈에서 관심사를 분리하여 공통관심 모듈(애스펙트)을 작성한 다음 핵심 서비스 모듈에 애스펙트를 위빙하는 과정으로 AOP를 구현한다.



Spring AOP 주요 용어

Target : 핵심모듈(클래스)

어드바이스(Advice) : Target 클래스에 제공할 공통기능(단순 애스펙트)

조인포인트(Join Point) : 어드바이스가 적용될 위치(메서드)

포인트 컷(PointCut) : 조인포인트로 지정된 메서드 선정

위빙(Weaving) : 어드바이스를 포인트 컷으로 삽입하는 과정

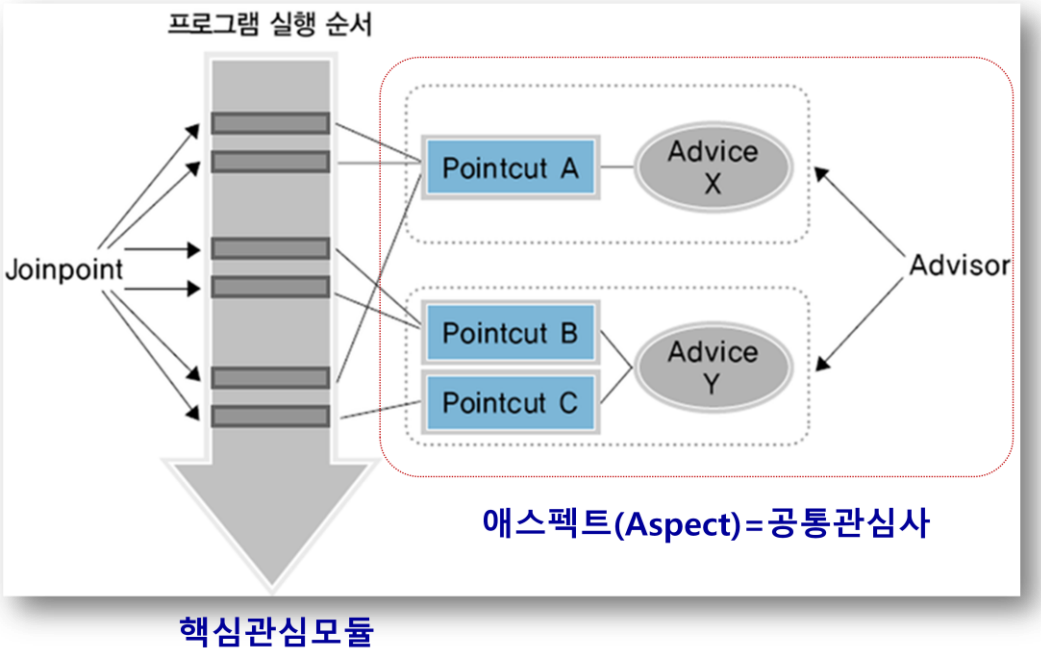
어드바이저(Advisor) : Advice + PointCut = 애스펙트(Aspect)

애스펙트(Aspect) : 1개 또는 2개 이상의 어드바이저(공통관심사)

프록시(Proxy) : AOP를 지원하는 객체(Object)

Advisor = Advice + PointCut 예

어드바이저는 어드바이스와 포인트컷으로 구성된다. 어드바이스는 단순한 애스펙트를 의미하며, 포인트 컷은 애스펙트가 핵심관심모듈에 추가될 위치를 의미한다. 따라서 어드바이저는 애스펙트와 위빙의 위치를 갖는 포인트컷의 구성으로 완전한 애스펙트라고 볼 수 있다.

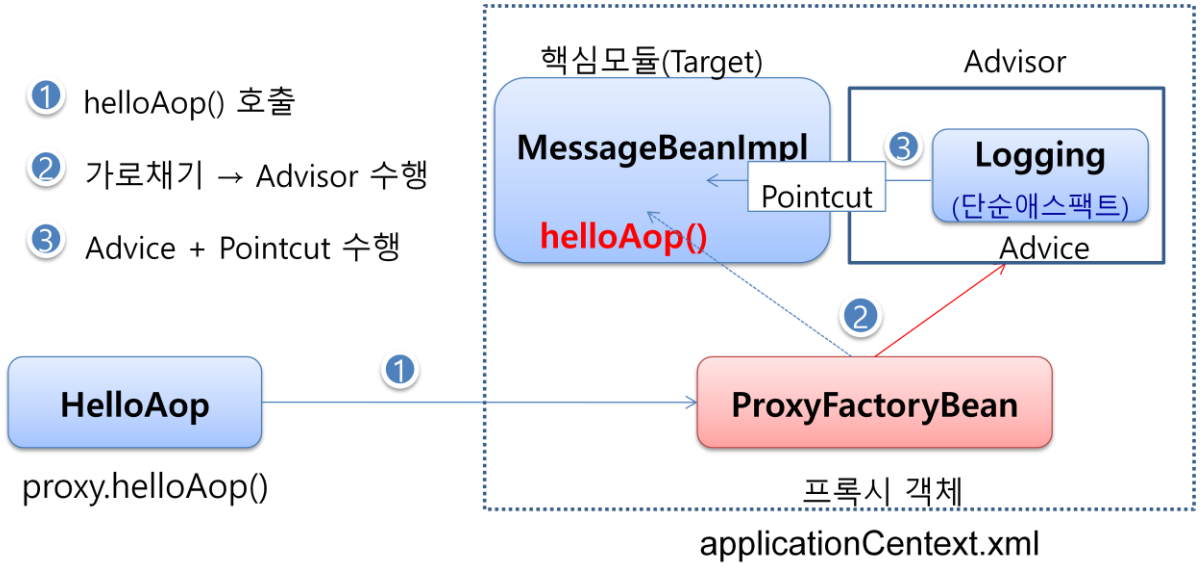


Spring AOP 구현

(1) Proxy 객체를 이용한 구현 방식

핵심모듈의 포인트컷 메서드를 호출할 경우 프록시 객체에 의해서 프로그램 수행 순서를

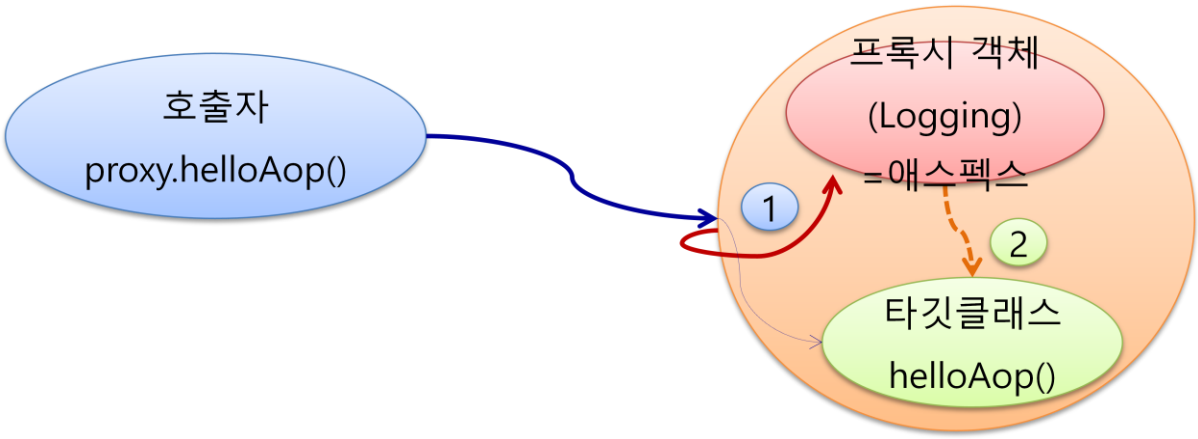
가로채기하여 애스펙트가 호출되도록 하는 방식으로 AOP를 구현하는 전통적인 방식이다.



*** 메서드 가로채기(interceptorNames) 예**

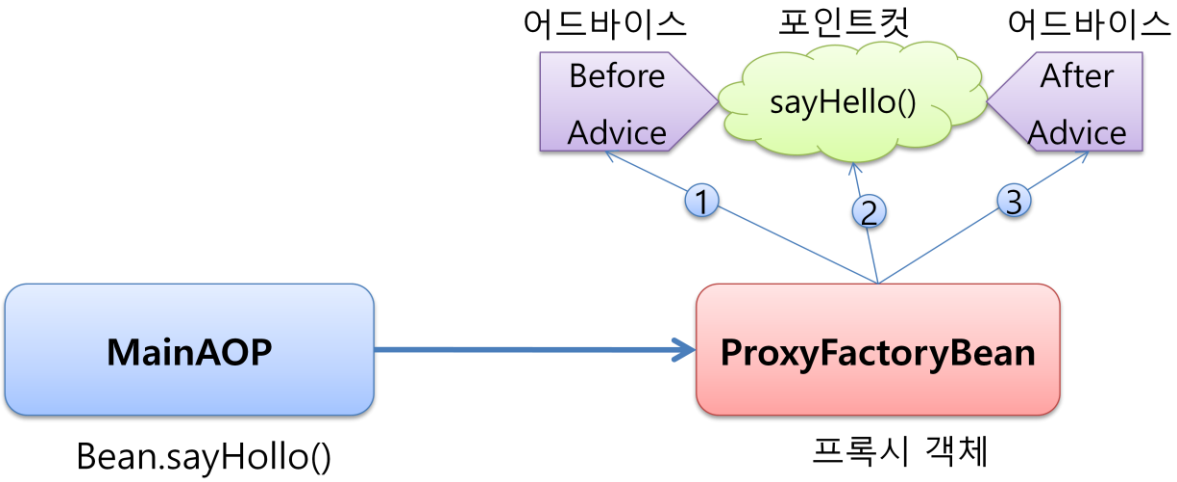
(1) 호출자가 타깃 클래스의 helloAop()을 호출할 경우 프록시 객체(Logging)가 타깃 객체(MessageImpl)로 위장해서 메서드 호출을 가로채기 한다.

(2) 이 과정에서 프록시(Logging) 객체의 invoke() 메서드가 자동 호출되며, proceed()에 의해서 타깃 메서드(helloAop)가 호출된다.



(2) Advice 객체를 이용한 구현 방식

포인트 컷 메서드의 호출 전에 수행할 내용은 Before Advice 객체를 통해서 구현하고, 포인트 컷 메서드의 호출 후에 수행할 내용은 After Advice 객체를 통해서 구현하는 방식이다.



(3) aop 엘리먼트를 이용한 설정방법

전통적인 AOP 방식을 개선한 방식으로 aop 엘리머트를 이용하여 AOP 설정을 쉽고 간편하게 설정할 수 있는 이점을 제공한다.

```
<!-- 애스펙트 클래스 정의 빈 -->
    <bean id="aspect" class="spring.aop.LogginTest"/>
    <aop:config>
        <aop:aspect   id="aspectTest" ref="aspect">
            <aop:pointcut   expression="execution(* helloAop())" id="point"/>
            <aop:around method="logAround" pointcut-ref="point"/>
        </aop:aspect>
    </aop:config>
    <!– 타킷 클래스 정의 빈 -->
    <bean id="target" class="spring.aop.MessageImpl">
        <property name="name" value="홍길동"/>
    </bean>
```