KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

AOP

[KB] IT's Your Life

KB 국민은행

AOP Aspect Oriented Programming

- 관점(Aspect)지향 프로그래밍

- 관심사의 예
  - 파라미터가 올바르게 들어왔을까?
  - 이 작업을 하는 사용자가 적절한 권한을 가진 사용자인가?
  - 이 작업에서 발생할 수 있는 모든 예외는 어떻게 처리해야 하는가?

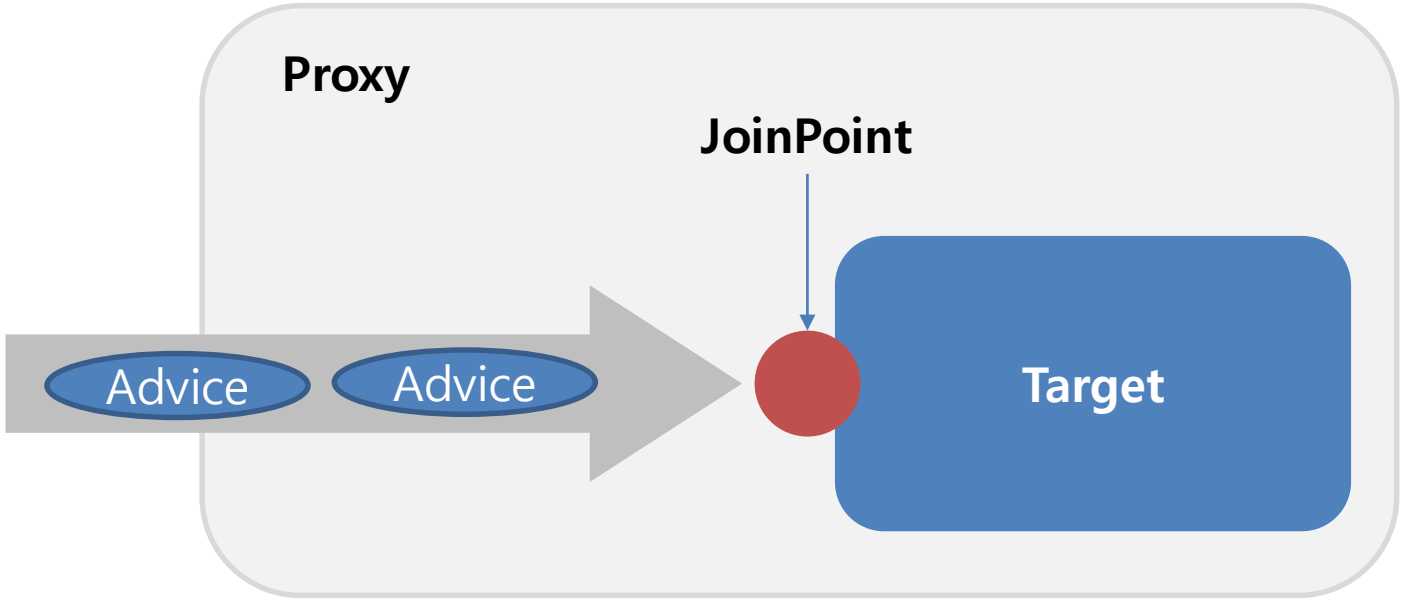
- 관심사의 분리
  - 개발자가 염두에 두어야하는 일들은 별도의 관심사로 분리하고,
  - 핵심 비즈니스 로직만을 작성할 것을 권장

→ 기존의 코드(핵심 비즈니스 로직)를 수정하지 않고, 원하는 기능(관심사)들과 결합

AOP 용어들

- Target: 개발자가 작성한 핵심 비즈니스 로직을 가지는 객체
- Advice: 관심사 로직
- Proxy: 타겟을 감싸는 래퍼 클래스, 내부에서 타겟을 호출. 그 과정에서 관심사(adice)를 거치도록 작성
- JoinPoint: AOP를 적용할 타겟 객체의 메서드

- 외부에서의 호출은 Proxy 객체를 통해서 Target 객체의 JoinPoint를 호출

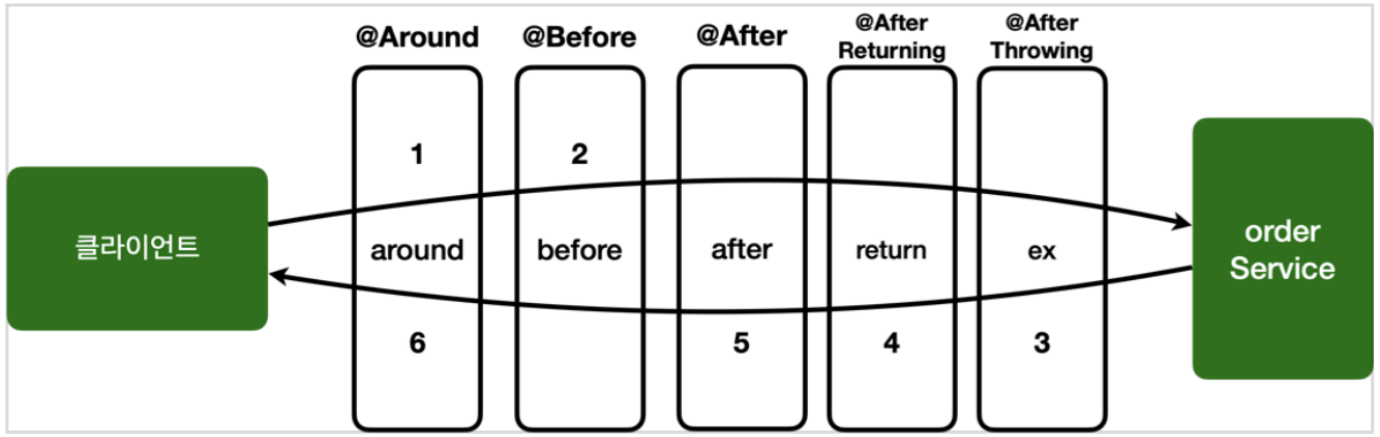


AOP 용어들

Advice

- 분리된 관심사 코드(메서드)
- Adice가 동작되는 위치를 어노테이션으로 지정

어노테이션	설명
@Around	메서드 실행 전/후로 실행
@Before	JointPoint 호출 전 실행
@AfterReturning	모든 실행이 정상적으로 이루어진 후에 실행
@AfterThrowing	예외가 발생한 뒤에 실행
@After	정삭적으로 실행되거나 예외가 발생했을 때 구분없이 실행되는 코드



AOP 용어들

Pointcut : Advice를 어떤 JointPoint에 결합할 것인지 결정하는 표현식

구분	설명
execution(@execution)	메서드를 기준으로 Pointcut을 설정
within(@within)	특정 타입(클래스)을 기준으로 Pointcut을 설정
this	주어진 인터페애스를 구현한 객체를 대상으로 Pointcut을 지정
args(@args)	특정 파라미터를 가지는 대상들만 Pointcut으로 설정
@annotation	특정한 어노테이션이 적용된 대상들만을 Pointcut으로 설정

- 예: Advice + Pointcut

리턴타입 제한없음

지정한 이름으로 시작하는 클래스명

매개변수 제한없음

메서드명 제한없음

```
@Before("execution(* org.scoula.sample.service.SampleService*.*(..))")
public void logBefore() {
    ...
}
```

의존 라이브러리

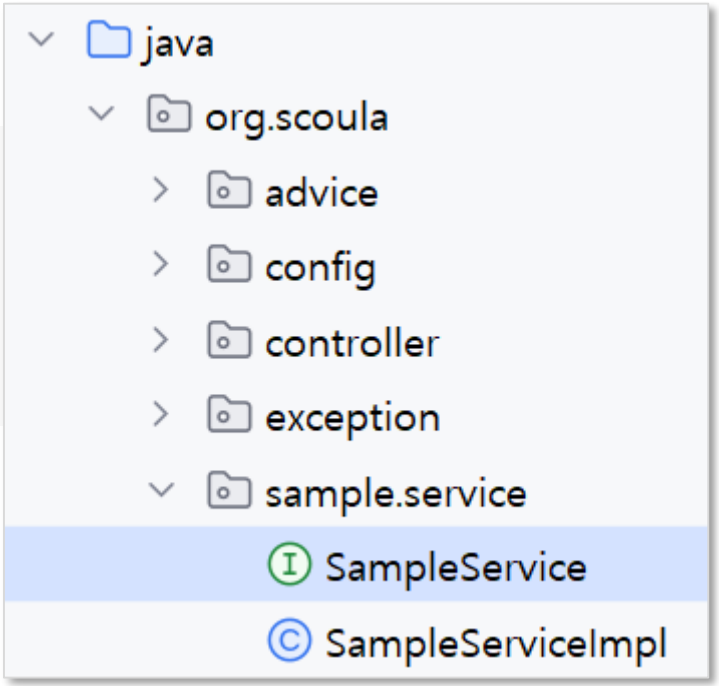
```
implementation 'org.aspectj:aspectjrt:1.9.20'
implementation 'org.aspectj:aspectjweaver:1.9.20'
```

프로젝트

- 템플릿: SpringLegacy
- name: aopex

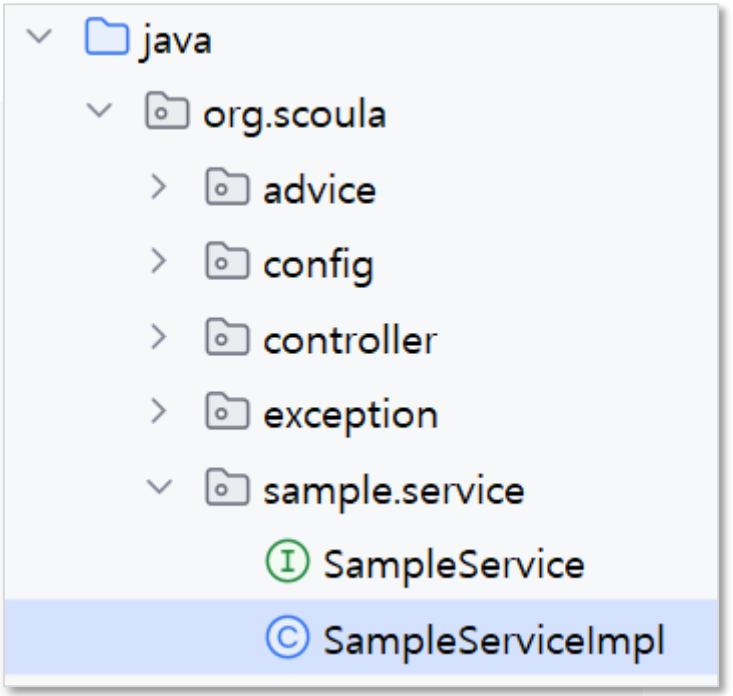
SampleService.java

```
package org.scoula.sample.service;

public interface SampleService {
    public Integer doAdd(String str1, String str2) throws Exception;
}
```



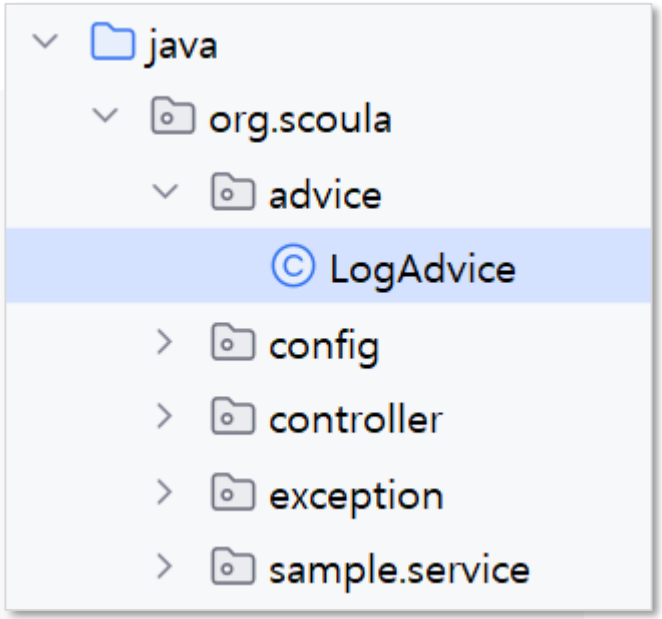
SampleServiceImpl.java

```
package org.scoula.sample.service;

import org.springframework.stereotype.Service;

@Service
public class SampleServiceImpl implements SampleService {

    @Override
    public Integer doAdd(String str1, String str2) throws Exception {
        return Integer.parseInt(str1) + Integer.parseInt(str2);
    }
}
```



LogAdvice

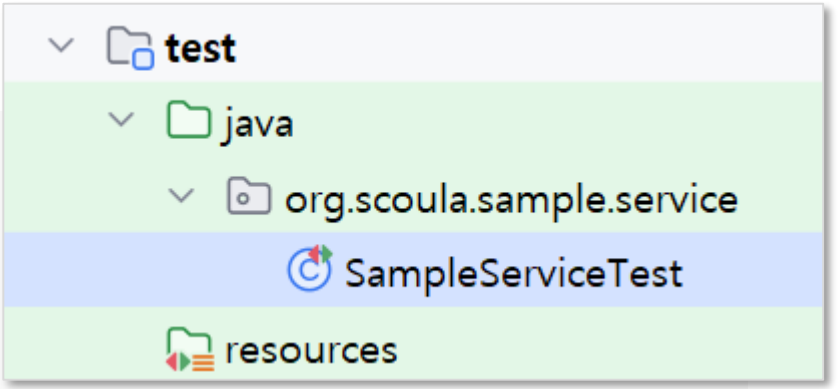
```
package org.scoula.advice;

import lombok.extern.log4j.Log4j;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.springframework.stereotype.Component;

@Aspect
@Log4j
@Component
public class LogAdvice {

    @Before("execution(* org.scoula.sample.service.SampleService*.*(..))")
    public void logBefore() {
        log.info("================================");
    }

}
```



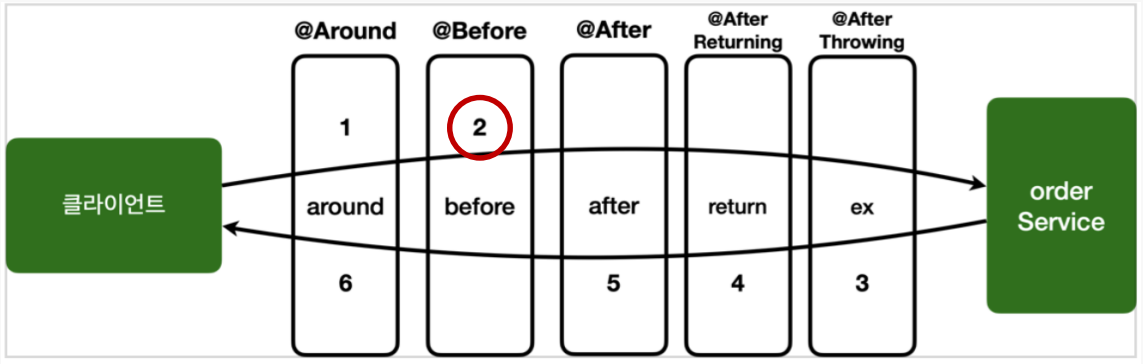
Rootconfig

```
@Configuration
@ComponentScan(basePackages = {
        "org.scoula.advice",
        "org.scoula.sample.service"
})
@EnableAspectJAutoProxy
public class RootConfig {

}
```

test::SampleServiceTest.java




```
package org.scoula.sample.service;
…
@ExtendWith(SpringExtension.class)
@ContextConfiguration(classes = { RootConfig.class })
@Log4j
class SampleServiceTest {
    @Autowired
    private SampleService service;

    @Test
    public void doAdd() throws Exception {
        log.info(service.doAdd("123", "456"));
    }
}
```



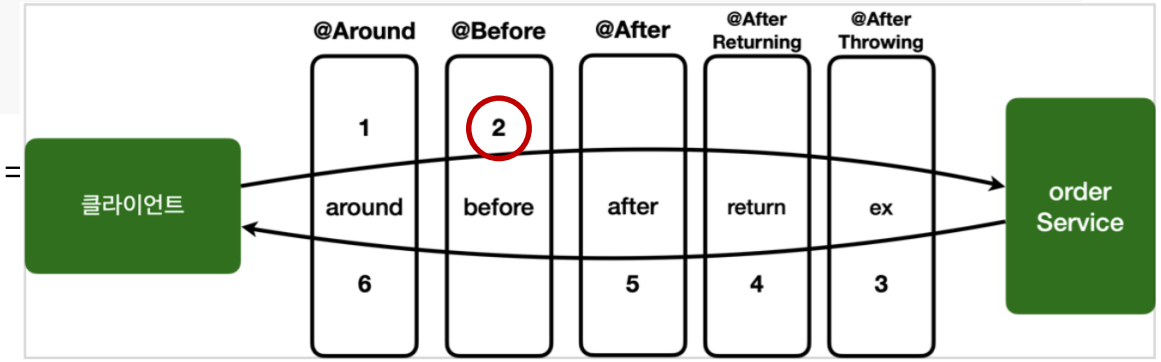

```
INFO : org.scoula.advice.LogAdvice - ==================================
INFO : org.scoula.sample.service.SampleServiceTest - 579
> Task :test
```

args를 이용한 파라미터 추적

- 해당 메서드에 전달되는 파라미터가 무엇인지 기록하건, 예외가 발생했을 때 어떤 파라미터에 문제가 있는지 알고 싶은 경우
- Pointcut에 args를 이용한 파라미터 추적 설정 추가

```
…
public class LogAdvice {

  @Before("execution(* org.scoula.sample.service.SampleService*.doAdd(String, String)) && args(str1, str2)")
  public void logBeforeWithParam(String str1, String str2) {
    log.info("str1:" + str1);
    log.info("str2:" + str2);
  }

}
```

```
INFO : org.scoula.advice.LogAdvice - ============================
INFO : org.scoula.advice.LogAdvice - str1:123
INFO : org.scoula.advice.LogAdvice - str2:456
INFO : org.scoula.sample.service.SampleServiceTest - 579
```



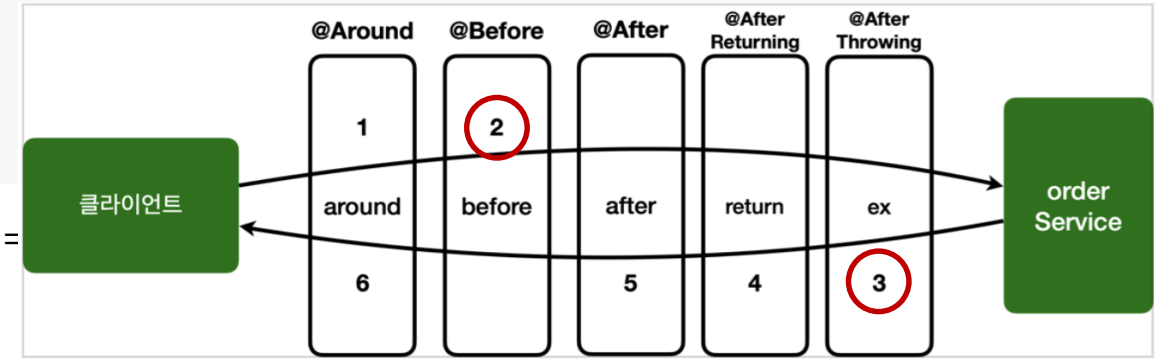
@AfterThrowing

- 지정된 대상이 예외를 발생한 후에 동작

```
@Aspect
@Log4j
@Component
public class LogAdvice {
  …

  @AfterThrowing(pointcut = "execution(* org.scoula.sample.service.SampleService*.*(..))", throwing="exception")
  public void logException(Exception exception) {
    log.info("Exception...!!!!");
    log.info("exception: " + exception);
  }
}
```

test::SampleServiceTest.java

```
public class SampleServiceTest {
    …

    @Test
    public void addError() throws Exception {
        log.info(service.doAdd("123", "ABC"));
    }
}
```



INFO : org.scoula.advice.LogAdvice - ==========
INFO : org.scoula.advice.LogAdvice - str1:123
INFO : org.scoula.advice.LogAdvice - str2:ABC
INFO : org.scoula.advice.LogAdvice - Exception...!!!!
INFO : org.scoula.advice.LogAdvice - exception: java.lang.NumberFormatException: For input string: "ABC"

@Around와 ProceedingJoinPoint

- 메서드의 실행 전과 실행 후에 처리가 가능

```
…
public class LogAdvice {

  @Around("execution(* org.scoula.sample.service.SampleService*.*(..))")
  public Object logTime(ProceedingJoinPoint pjp) {
     long start = System.currentTimeMillis();

     log.info("Target: " + pjp.getTarget());
     log.info("Param: " + Arrays.toString(pjp.getArgs()));

     Object result = null;
     try {
         result = pjp.proceed(); // 실제 메서드 호출
     } catch(Throwable e) {
         e.printStackTrace();
     }
```

@Around와 ProceedingJoinPoint

```
        long end = System.currentTimeMillis();

        log.info("TIME: " + (end - start));

        return result;
    }
}
```

INFO : org.scoula.advice.LogAdvice - Target: org.scoula.sample.service.SampleServiceImpl@74971ed9
INFO : org.scoula.advice.LogAdvice - Param: [123, 456]
INFO : org.scoula.advice.LogAdvice - ==========
INFO : org.scoula.advice.LogAdvice - str1:123
INFO : org.scoula.advice.LogAdvice - str2:456
INFO : org.scoula.advice.LogAdvice - TIME: 3
INFO : org.scoula.sample.service.SampleServiceTest - 579