```java
package com.myspring.pro30.common.log;

import java.util.Arrays;

import org.aspectj.lang.JoinPoint;
import org.aspectj.lang.ProceedingJoinPoint;
import org.aspectj.lang.annotation.After;
import org.aspectj.lang.annotation.Around;
import org.aspectj.lang.annotation.Aspect;
import org.aspectj.lang.annotation.Before;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Component;

@Component
@Aspect
public class LoggingAdvice {
    private static final Logger logger = LoggerFactory.getLogger

    // target 메서드의 파라미터등 정보를 출력합니다.
    @Before("execution(* com.myspring.pro30.*.service.*.*(..)) c
            + "execution(* com.myspring.pro30.*.dao.*.*(..))")
    public void startLog(JoinPoint jp) {
        logger.info("-------------------------------------");
        logger.info("-------------------------------------");

        /* 전달되는 모든 파라미터들을 Object의 배열로 가져옵니다. */
        logger.info("1:" + Arrays.toString(jp.getArgs()));

        /* 해당 Advice의 타입을 알아냅니다. */
        logger.info("2:" + jp.getKind());

        /* 실행하는 대상 객체의 메소드에 대한 정보를 알아낼 때 사용합니다. *
        logger.info("3:" + jp.getSignature().getName());

        /* target 객체를 알아낼 때 사용합니다. */
        logger.info("4:" + jp.getTarget().toString());

        /* Advice를 행하는 객체를 알아낼 때 사용합니다. */
        logger.info("5:" + jp.getThis().toString());

```

```
42      }
43
44      @After("execution(* com.myspring.pro30.*.service.*.*(..)) or
45              + "execution(* com.myspring.pro30.*.dao.*.*(..))")
46      public void after(JoinPoint jp) {
47          logger.info("-------------------------------------");
48          logger.info("-------------------------------------");
49
50          // 전달되는 모든 파라미터들을 Object의 배열로 가져옵니다.
51          logger.info("1:" + Arrays.toString(jp.getArgs()));
52
53          // 해당 Advice의 타입을 알아냅니다.
54          logger.info("2:" + jp.getKind());
55
56          // 실행하는 대상 객체의 메소드에 대한 정보를 알아낼 때 사용합니다.
57          logger.info("3:" + jp.getSignature().getName());
58
59          // target 객체를 알아낼 때 사용합니다.
60          logger.info("4:" + jp.getTarget().toString());
61
62          // Advice를 행하는 객체를 알아낼 때 사용합니다
63          logger.info("5:" + jp.getThis().toString());
64
65      }
66
67      /*
68      // target 메소드의 동작 시간을 측정합니다.
69      @Around("execution(* com.myspring.pro30.*.service.*.*(..)) o
70              + "execution(* com.myspring.pro30.*.dao.*.*(..))")
71      public Object timeLog(ProceedingJoinPoint pjp) throws Throwa
72          long startTime = System.currentTimeMillis();
73          logger.info(Arrays.toString(pjp.getArgs()));
74
75          // 실제 타겟을 실행하는 부분이다. 이 부분이 없으면 advice가 적용된
76          Object result = pjp.proceed(); // proceed는 Exception 보다
77
78          long endTime = System.currentTimeMillis();
79          logger.info(pjp.getSignature().getName() + " : " + (endT
80          logger.info("==============================");
81
82          // Around를 사용할 경우 반드시 Object를 리턴해야 합니다.
```

```
83          return result;
84      }*/
85
86  }
87
88
```