



박 경 태

comsi.java@gmail.com

고급 자바 프로그래밍  
: STS를 이용한 Spring 프로그래밍

# 강의 내용

순서	내 용
1	<ul style="list-style-type: none"><li>• Spring IoC를 이용한 비즈니스 컴포넌트 만들기</li></ul>
2	<ul style="list-style-type: none"><li>• <b>Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기</b></li><li>• Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리</li></ul>
3	<ul style="list-style-type: none"><li>• Spring MVC를 이용한 MVC 아키텍처 적용하기</li></ul>
4	<ul style="list-style-type: none"><li>• Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등)</li></ul>
5	<ul style="list-style-type: none"><li>• Spring과 MyBatis 연동하기</li><li>• Spring과 JPA 연동하기</li></ul>

# AOP(Aspect Oriented Programming)개요

## ■ 기존 OOP의 한계점

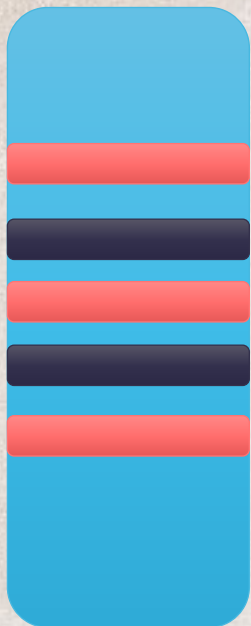
- 어플리케이션을 개발할 때, 메소드 내에 핵심적인 비즈니스 로직(핵심관심사)을 담당하는 소스코드와 Logging, Exception 처럼 비즈니스 로직과 관계없는 코드(횡단관심사)를 같이 구현할 수 밖에 없다.
- Logging, Exception을 처리하기 위한 소스코드는 어플리케이션 소스 코드의 전체 영역에 걸쳐 비슷한 패턴으로 중복해서 구현하는 것이 일반적이다.
- Logging, Exception 정책이 바뀔 경우 개발자가 예제 소스를 일일이 찾아서 수정해야 하는 번거로움을 감수해야 한다.

■ OOP의 한계점을 극복하기 위한 대안으로 등장한 것이  
**AOP (Aspect Oriented Programming)**이다.

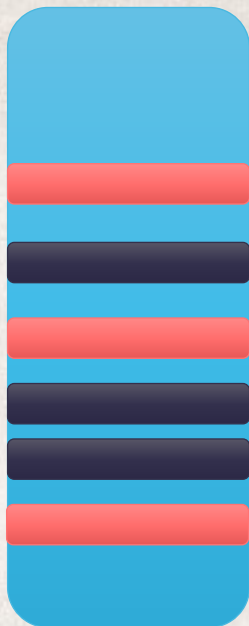
- OOP는 Logging, Exception과 같은 Infrastructure를 해결하지 못함.
- AOP를 OOP기반 하에서의 OOP문제점을 보완하는 역할을 함.

# OOP(Object-Oriented Programming)

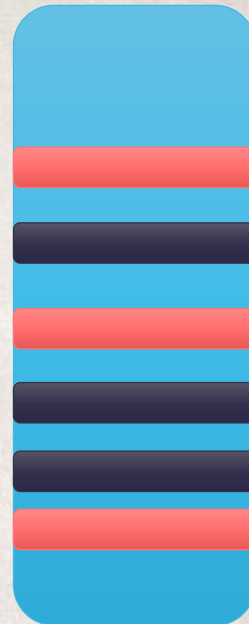
모듈A  
(insertBoard())



모듈B  
(selectBoard())



모듈C  
(deleteBoard())



핵심 기능과 로그가 혼합



코드 중복 상승

유지보수성 저하

생산성 저하

재사용성 저하



주 기능 구현(핵심기능)



로그(유틸리티기능)

# AOP(Aspect Oriented Programming)

- 어플리케이션을 다양한 관점으로 분해하여 객체지향에서 추구하는 모듈화를 더 잘 지원하도록 하는 기법
- 가장 기초가 되는 개념은 ‘관심의 분리(Separation of Concerns)’이고, **핵심 관심사**에 대한 관점과 **횡단 관심사**에 대한 관점으로 나뉜다.
  - 핵심 관심사(core concerns): 각 모듈에서 수행해야 하는 기본적이고 대표적인 업무 기능
  - 횡단 관심사(cross-cutting concerns): 여러 개의 모듈에 걸치는 시스템의 부가적인 업무로 로깅, 예외처리, 성능 관리 등의 기능
- 현재 주류인 객체 지향을 **대신하는 것이 아니라 보완**하는 새로운 패러다임

# AOP(Aspect Oriented Programming)

- OOP의 한계?

객체 지향 프로그램은 추상화나 캡슐화를 통해 관심사에 대해 모듈화를 수행하고 사용하므로 횡단 관심사에 해당하는 로깅, 예외 모듈 등을 핵심 관심사에 직접 포함하게 됨.

- AOP 사용

로깅, 예외 등의 횡단 관심사를 직접 호출하는 것이 아니라 Aspect를 활용해 컴파일 시나 런타임 시에 weaving 과정을 거쳐 하나의 시스템으로 조립하는 것이 가능하게 됨.

**Weaving?** 핵심관심사와 횡단관심사를 컴파일이나 런타임 시에 합성하여 주는 프로세스로 핵심관심사에서 횡단 관심사를 자동으로 삽입하여 호출 실행 시켜줌.

# 스프링(Spring)은...

객체지향  
(OOP)

+

의존관계주입  
(DI)

+

관점지향프로그래밍  
(AOP)



객체 지향보다 독립성을 높여 재사용성과 보전성 향상

# AOP 구현

- 비즈니스 메소드가 실행되기 직전에 공통으로 처리할 로직을 LogAdvice클래스에 printLog()메소드로 구현
- LogAdvice클래스의 printLog() 메소드를 BoardService 컴포넌트에서 사용할 수 있도록 BoardServiceImpl 클래스를 수정
  - 각 매소드에서 비즈니스 로직을 수행하기 전에 printLog() 메소드를 호출

# LogAdvice 클래스

- 비즈니스 로직 수행전에 로그 출력하는 클래스

```
BoardDAO.java  UserDao.java  JDBCUtil.java  LogAdvice.java ✕
1 package kr.ac.inje.comsi.common;
2
3 public class LogAdvice {
4     public void printLog(){
5         System.out.println("[공통 로그] 비즈니스 로직 수행 전 동작");
6     }
7 }
8
```

# BoardServiceImpl.java 클래스 수정

BoardDAO.java UserDao.java JDBCUtil.java LogAdvice.java BoardServiceImpl.java BoardServiceClient.java

```
1 package kr.ac.inje.comsi.board.impl;
2
3 import java.util.List;
4
5 import org.springframework.beans.factory.annotation.Autowired;
6 import org.springframework.stereotype.Service;
7
8 import kr.ac.inje.comsi.board.BoardService;
9 import kr.ac.inje.comsi.board.BoardVO;
10 import kr.ac.inje.comsi.common.LogAdvice;
11
12 @Service("boardService")
13 public class BoardServiceImpl implements BoardService {
14
15     @Autowired
16     private BoardDAO boardDAO;
17     private LogAdvice log; → 추가
18
19
20     public BoardServiceImpl() {
21         log = new LogAdvice(); → 추가
22     }
23
24     @Override
25     public void insertBoard(BoardVO vo) {
26         log.printLog(); → 추가
27         boardDAO.insertBoard(vo);
28     }
29
30     @Override
31     public void updateBoard(BoardVO vo) {
32         log.printLog(); → 추가
33         boardDAO.updateBoard(vo);
34     }
35 }
```

```

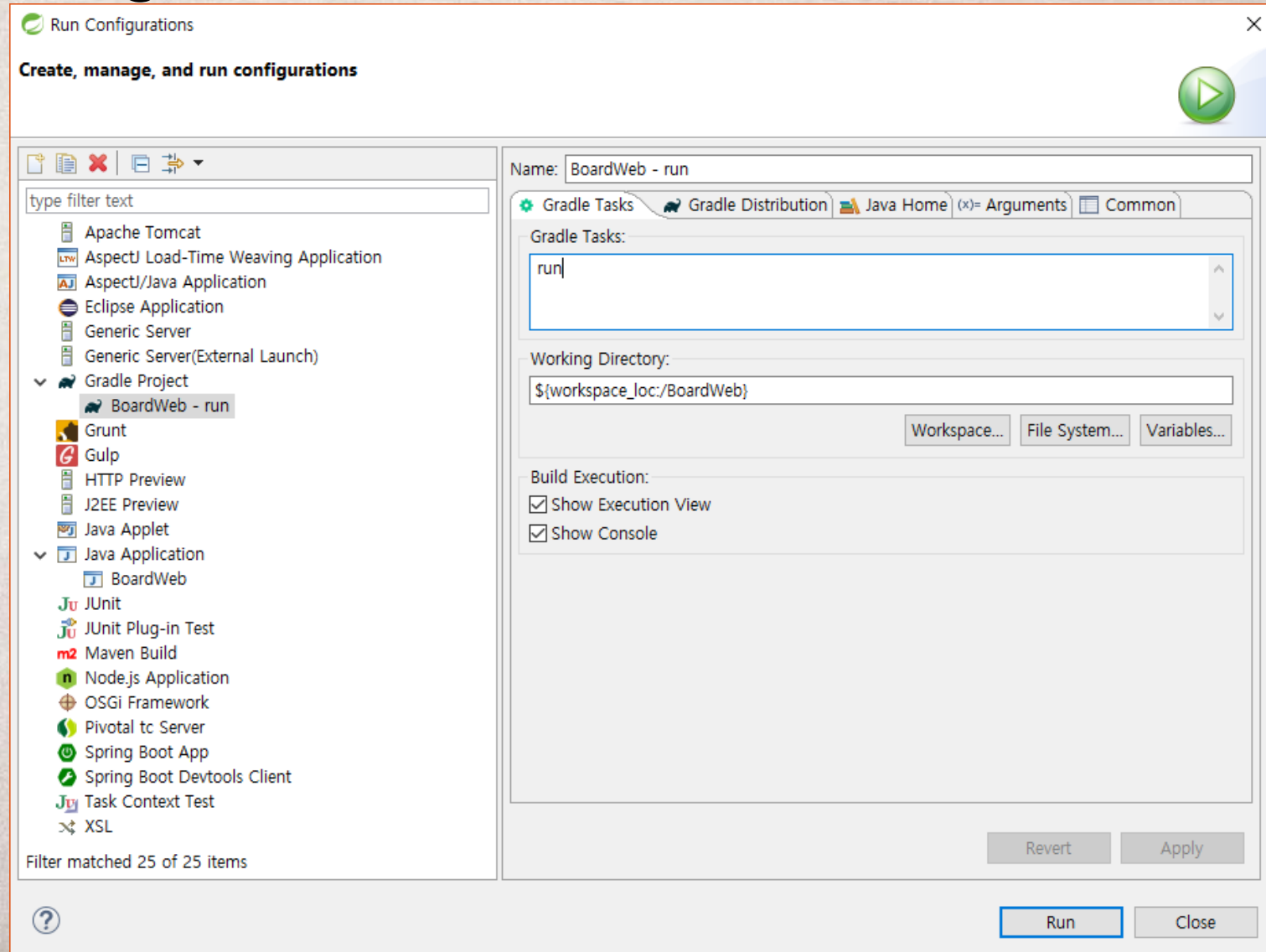
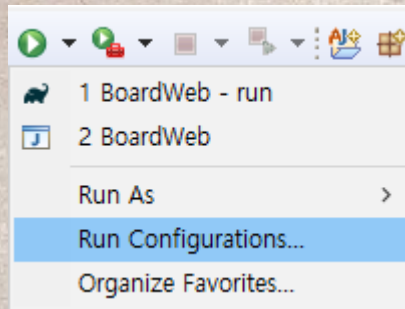
36 @Override
37 public void deleteBoard(BoardVO vo) {
38     log.printLog();           → 추가
39     boardDAO.deleteBoard(vo);
40 }
41
42 @Override
43 public void getBoard(BoardVO vo) {
44     log.printLog();           → 추가
45     boardDAO.getBoard(vo);
46 }
47
48 @Override
49 public List<BoardVO> getBoardList(BoardVO vo) {
50     log.printLog();           → 추가
51     return boardDAO.getBoardList(vo);
52 }
53
54 }
55

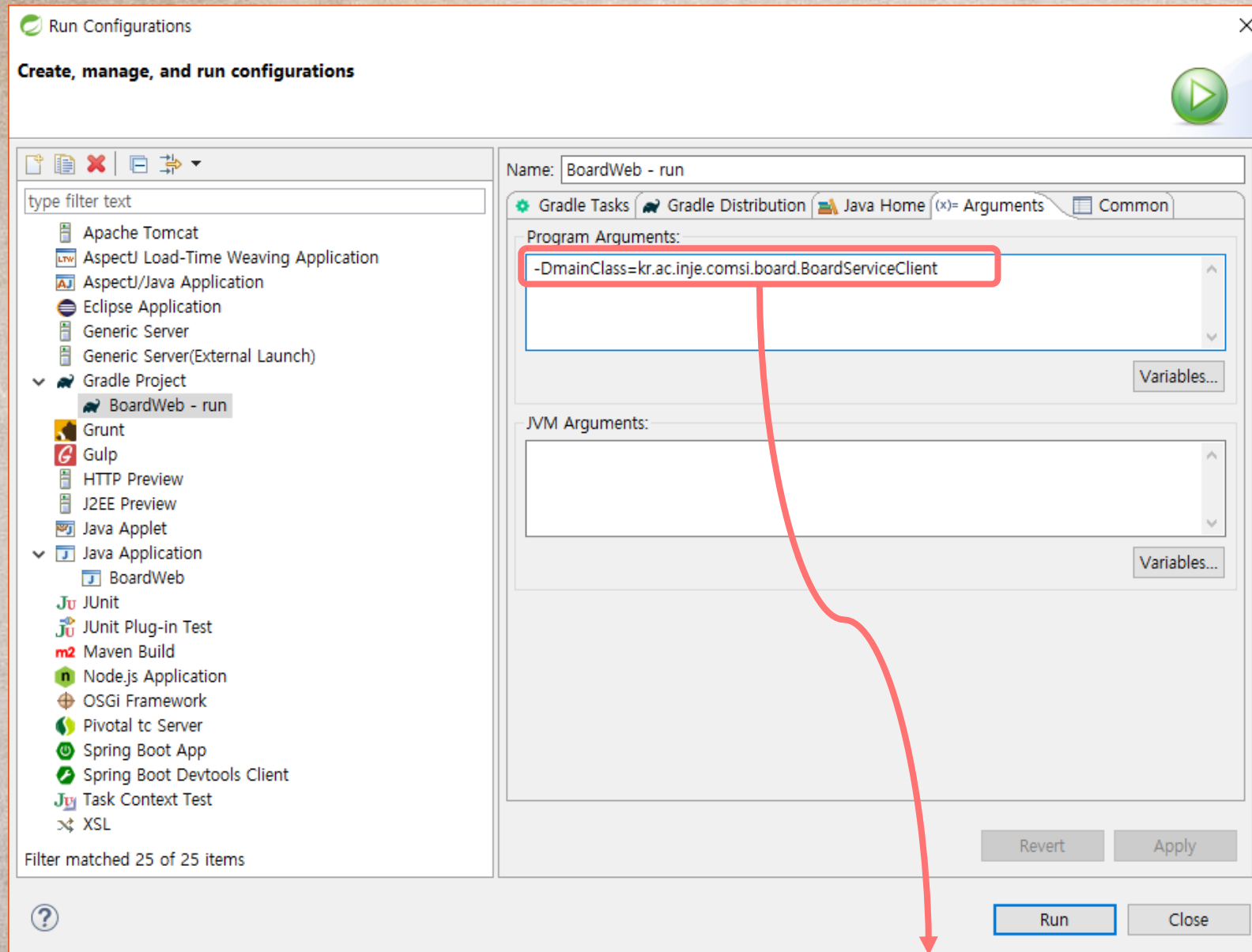
```

- LogAdvice 객체가 소스에 강하게 결합되어 있음
  - LogAdvice를 다른 클래스로 변경하거나 printLog() 메소드의 시그니처가 변경되는 상황에 유연하게 대처하기가 어렵다.

# 실행

- Run 아이콘 > Run Configurations... 실행





**-DmainClass=kr.ac.inje.comsi.board.BoardServiceClient**

BoardWeb - run [Gradle Project] run in D:\workspace\java\2017\_1\BoardWeb (2017. 9. 7 오후 9:30:06)  
Java Home: C:\Program Files\Java\jdk1.8.0\_144  
JVM Arguments: None  
Program Arguments: -DmainClass=kr.ac.inje.comsi.board.BoardServiceClient  
Gradle Tasks: run

:compileJava  
:processResources UP-TO-DATE  
:classes  
:run9월 07, 2017 9:30:06 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions  
정보: Loading XML bean definitions from class path resource [applicationContext.xml]  
9월 07, 2017 9:30:06 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh  
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Thu Sep 07 21:30:06 KST 2017]; root c

[공통 로그] 비즈니스 로직 수행 전 동작 → printLog() 실행  
==> JDBC로 insertBoard() 기능 처리  
[공통 로그] 비즈니스 로직 수행 전 동작 → printLog() 실행  
==> JDBC로 getBoardList() 기능 처리  
--> BoardVO [seq=3, title=AOP테스트, writer=홍길동, content=임시 내용 ....., regDate=2017-09-07, cnt=0]  
--> BoardVO [seq=2, title=임시 제목, writer=홍길동, content=임시 내용 ....., regDate=2017-09-07, cnt=0]  
--> BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다., regDate=2017-09-06, cnt=0]  
9월 07, 2017 9:30:07 오후 org.springframework.context.support.GenericXmlApplicationContext doClose  
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Thu Sep 07 21:30:06 KST 2017]; root c

BUILD SUCCESSFUL

Total time: 1.074 secs

## 만약?

- LogAdvice 클래스가 Log4jAdvice로 변경된다면?
  - printLog() → printLogging()으로 변경된다면

→ LogAdvice 클래스를 사용하는 BoardServiceImpl 클래스를 수정

- 생성자 수정
- 로그 출력 메소드 수정

# AOP 시작하기

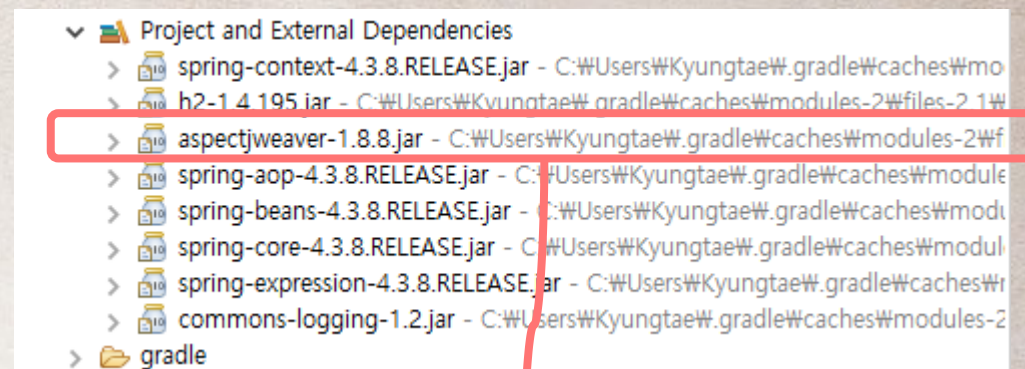
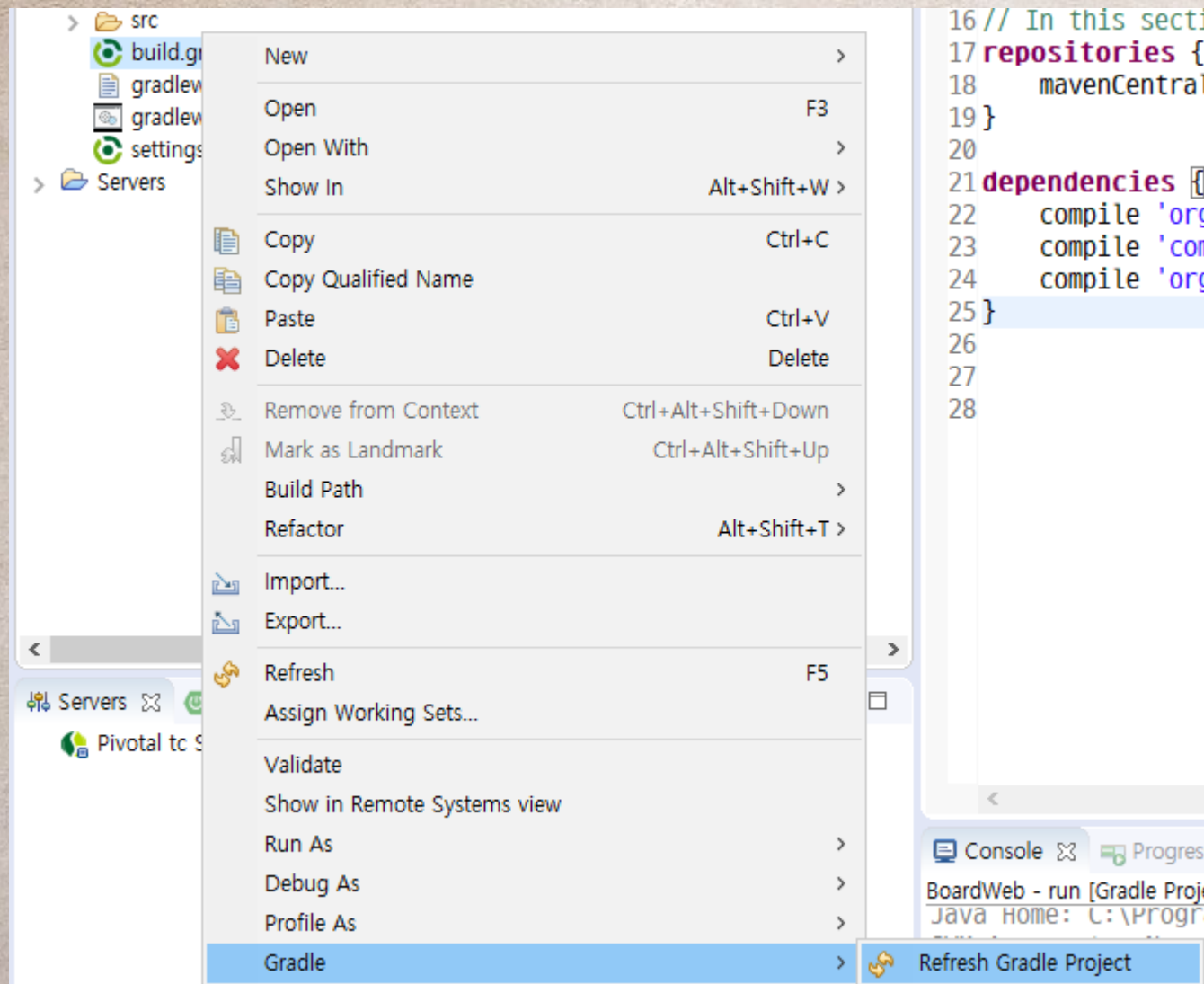
- 비즈니스 클래스 수정(BoardServiceImpl.java 클래스를 원상태로 복원)
- AOP 라이브러리 추가(gradle.build)
- 스프링 설정파일(applicationContext.xml) 파일에서 사용할 aop 네임 스페이스 추가

# build.gradle 수정 - AOP 라이브러리(aspectjweaver)

```
BoardServiceImpl.java BoardServiceClient.java build.gradle ✕
1/*
2 * This build file was generated by the Gradle 'init' task.
3 *
4 * This generated file contains a sample Java project to get you started.
5 * For more details take a look at the Java Quickstart chapter in the Gradle
6 * user guide available at https://docs.gradle.org/3.3/userguide/tutorial_java_projects.html
7 */
8
9// Apply the java plugin to add support for Java
10apply plugin: 'application'
11
12mainClassName = System.getProperty("mainClass");
13
14compileJava.options.encoding = 'UTF-8'
15
16// In this section you declare where to find the dependencies of your project
17repositories {
18    mavenCentral()
19}
20
21dependencies {
22    compile 'org.springframework:spring-context:4.3.8.RELEASE'
23    compile 'com.h2database:h2:1.4.195'
24    compile 'org.aspectj:aspectjweaver:1.8.8'
25}
26
27
```

→ 추가

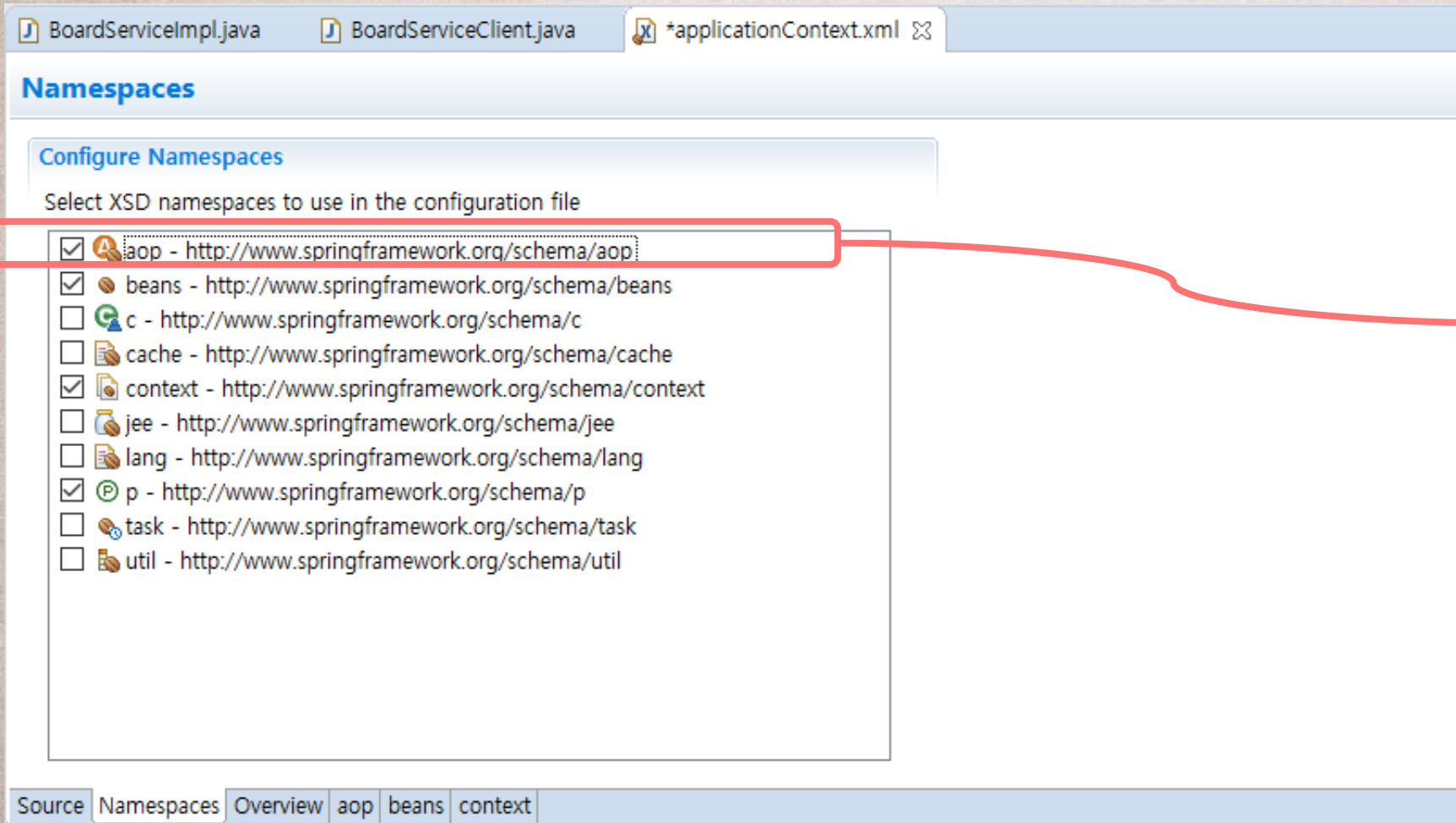
# Refresh gradle projec를 통한 라이브러리 추가



추가됨

# applicationContext.xml 수정 - aop 네임스페이스 추가

- applicationContext.xml을 열고 아래 탭 항목에서 “Namespaces” 선택



aop 항목을 체크(선택)

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:p="http://www.springframework.org/schema/p"
5     xmlns:context="http://www.springframework.org/schema/context"
6     xmlns:aop="http://www.springframework.org/schema/aop"
7     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9     http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12     <!--
13     <bean id="userService" class="kr.ac.inje.comsi.user.impl.UserServiceImpl">
14         <property name="userDAO" ref="userDAO"></property>
15     </bean>
16
17     <bean id="userDAO" class="kr.ac.inje.comsi.user.impl.UserDAO"></bean>
18     -->
19 </beans>
```

추가

추가

우선, LogAdvice 클래스를 <bean> 등록하고 AOP 관련 설정을 다음과 같이 추가한다.

# 변경된 applicationContext.xml - aop 관련 설정 추가

```
BoardServiceImpl.java BoardServiceClient.java applicationContext.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:p="http://www.springframework.org/schema/p"
5       xmlns:context="http://www.springframework.org/schema/context"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9                           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12
13     <!-- LogAdvice bean 등록 -->
14     <bean id="log" class="kr.ac.inje.comsi.common.LogAdvice"></bean>
15     <!-- aop 관련 설정 -->
16     <aop:config>
17         <aop:pointcut expression="execution(* kr.ac.inje.comsi..*Impl.*(..))" id="allPointcut"/>
18         <aop:aspect ref="log">
19             <aop:before method="printLog" pointcut-ref="allPointcut"/>
20         </aop:aspect>
21     </aop:config>
22 </beans>
23
24
25
```

# 실행

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 7 오후 10:09:55)
Gradle Version: 3.3
Java Home: C:\Program Files\Java\jdk1.8.0_144
JVM Arguments: None
Program Arguments: -DmainClass=kr.ac.inje.comsi.board.BoardServiceClient
Gradle Tasks: run

:compileJava
:processResources
:classes
:run9월 07, 2017 10:09:56 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 07, 2017 10:09:56 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Thu Sep 07 22:09:56 KST 2017]; root
[공통 로그] 비즈니스 로직 수행 전 동작 —————> printLog() 실행
==> JDBC로 insertBoard() 기능 처리
[공통 로그] 비즈니스 로직 수행 전 동작 —————> printLog() 실행
==> JDBC로 getBoardList() 기능 처리
---> BoardVO [seq=4, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-07, cnt=0]
---> BoardVO [seq=3, title=AOP테스트, writer=홍길동, content=임시 내용 ....., regDate=2017-09-07, cnt=0]
---> BoardVO [seq=2, title=임시 제목, writer=홍길동, content=임시 내용 ....., regDate=2017-09-07, cnt=0]
---> BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다...., regDate=2017-09-06, cnt=0]
9월 07, 2017 10:09:57 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Thu Sep 07 22:09:56 KST 2017]; root

BUILD SUCCESSFUL

Total time: 1.296 secs
```

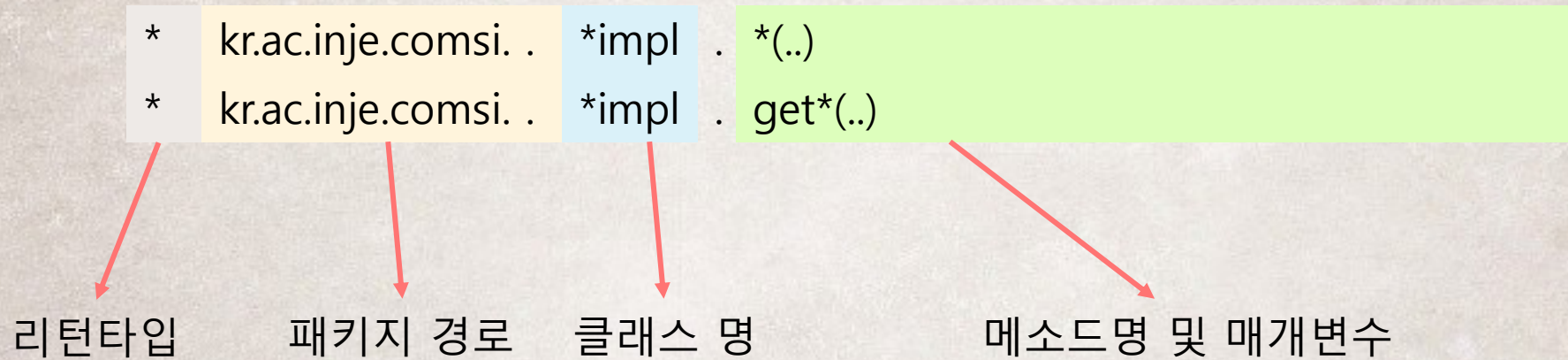
# 만약 LogAdvice 클래스를 Log4jAdvice로 변경한다면?

```
BoardServiceImpl.java BoardServiceClient.java applicationContext.xml
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:p="http://www.springframework.org/schema/p"
5       xmlns:context="http://www.springframework.org/schema/context"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9                           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12
13     <!-- LogAdvice bean 등록 -->
14     <bean id="log" class="kr.ac.inje.comsi.common.LogAdvice"></bean>
15     <!-- aop 관련 설정 -->
16     <aop:config>
17         <aop:pointcut expression="execution(* kr.ac.inje.comsi..*Impl.*(..))" id="allPointcut"/>
18         <aop:aspect ref="log">
19             <aop:before method="printLog" pointcut-ref="allPointcut"/>
20         </aop:aspect>
21     </aop:config>
22 </beans>
23
24
25
```

Log4jAdvice로 변경

printLogging으로 변경

# 포인트 컷 표현식의 구성과 의미



## ① 리턴타입 지정

- 리턴타입 지정에서 가장 기본적인 방법은 \* 캐릭터를 이용하는 것

표현식	설명
*	모든 리턴타입 허용
void	리턴타입이 void인 메소드 선택
!void	리턴타입이 void가 아닌 메소드 선택

## ② 패키지 지정

- 패키지 경로를 지정할 때는 '\*', '..' 캐릭터를 이용

표현식	설명
kr.ac.inje.comsi	정확하게 kr.ac.inje.comsi 패키지만 선택
kr.ac.inje.comsi..	kr.ac.inje.comsi 패키지로 시작하는 모든 패키지 선택
com.springbook..impl	kr.ac.inje.comsi 패키지로 시작하면서 마지막 패키지 이름이 impl로 끝나는 패키지 선택

### ③ 클래스 지정

- 클래스 이름을 지정할 때는 '\*', '+' 캐릭터를 이용

표현식	설명
BoardServiceImpl	정확하게 BoardServiceImpl 클래스만 선택
*Impl	클래스 이름이 Impl로 끝나는 클래스만 선택
BoardService+	클래스 이름 뒤에 '+'가 붙으면 해당 클래스로부터 파생된 모든 자식 클래스 선택. 인터페이스 뒤에 '+'가 붙으면 해당 인터페이스를 구현한 모든 클래스 선택

## ④ 메소드 지정

- 메소드를 지정할 때는 주로 '\*' 캐릭터를 사용하고 매개변수를 지정할 때는 '..'을 사용

표현식	설명
*(..)	가장 기본 설정으로 모든 메소드 선택
get*(..)	메소드 이름이 get으로 시작하는 모든 메소드 선택

## ⑤ 매개변수 지정

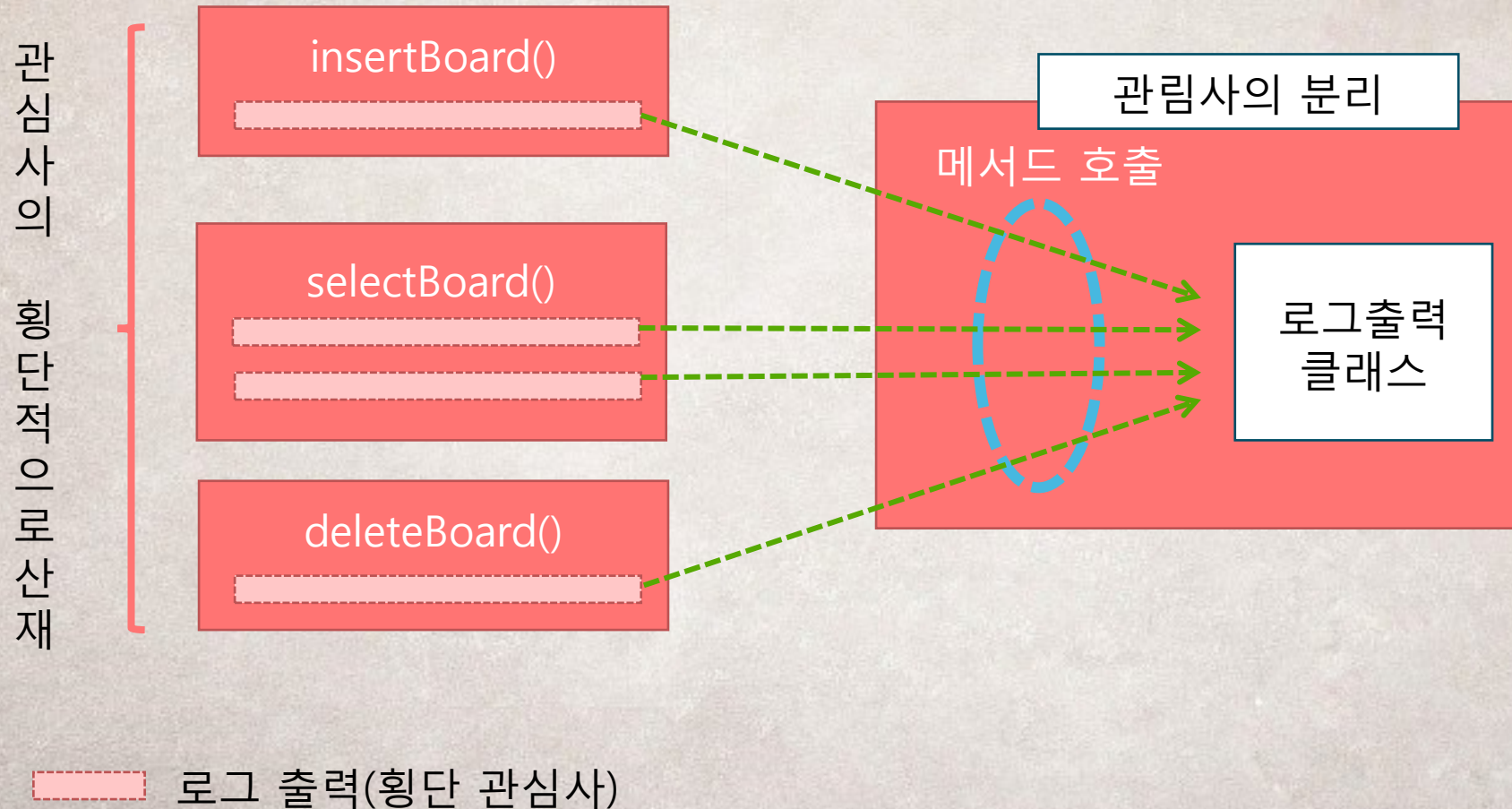
- 매개변수를 지정할 때는 '..', '\*'캐릭터를 사용하거나 정확한 타입을 지정

표현식	설명
(..)	가장 기본 설정으로 '..'은 모든 매개변수의 개수와 타입에 제약이 없음을 의미
(*)	반드시 1개의 매개변수를 가지는 메소드만 선택
(kr.ac.inje.comsi.user.UserVO)	매개변수로 UserVO를 가지는 메소드만 선택, 이때 클래스의 패키지 경로가 반드시 포함되어야 함
(!kr.ac.inje.comsiuser.UserVO)	매개변수로 UserVO를 가지지 않는 메소드만 선택
(Integer, ..)	한 개 이상의 매개변수를 가지되, 첫번째 매개변수의 타입이 Integer인 메소드만 선택
(Integer, *)	반드시 두 개의 매개변수를 가지되, 첫번째 매개변수의 타입이 Integer인 메소드만 선택

정리하면,

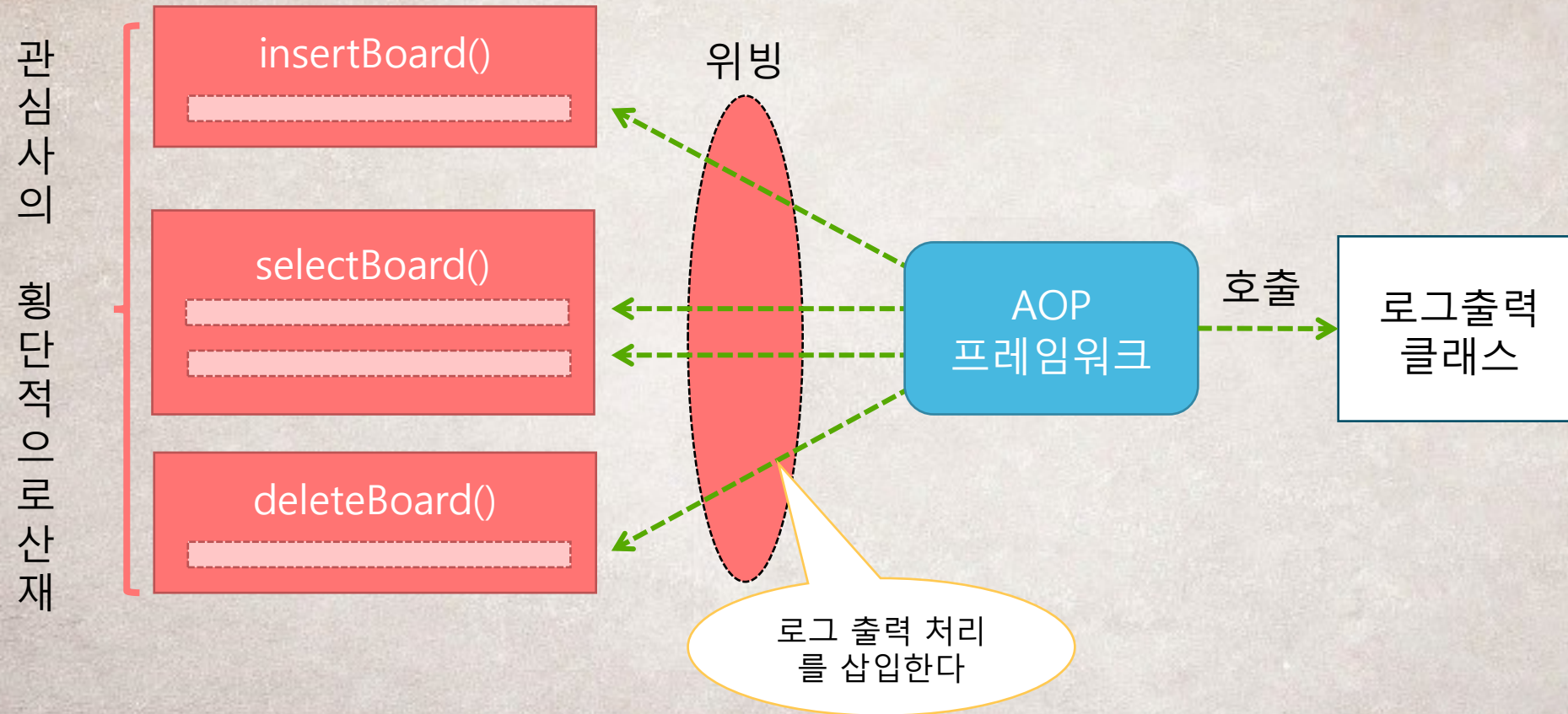
# 관점 지향과 횡단 관심사의 분리

- 기존 객체 지향에서 관심사 분리



# 관점 지향과 횡단 관심사의 분리

- AOP에서 횡단 관심사 분리와 위빙

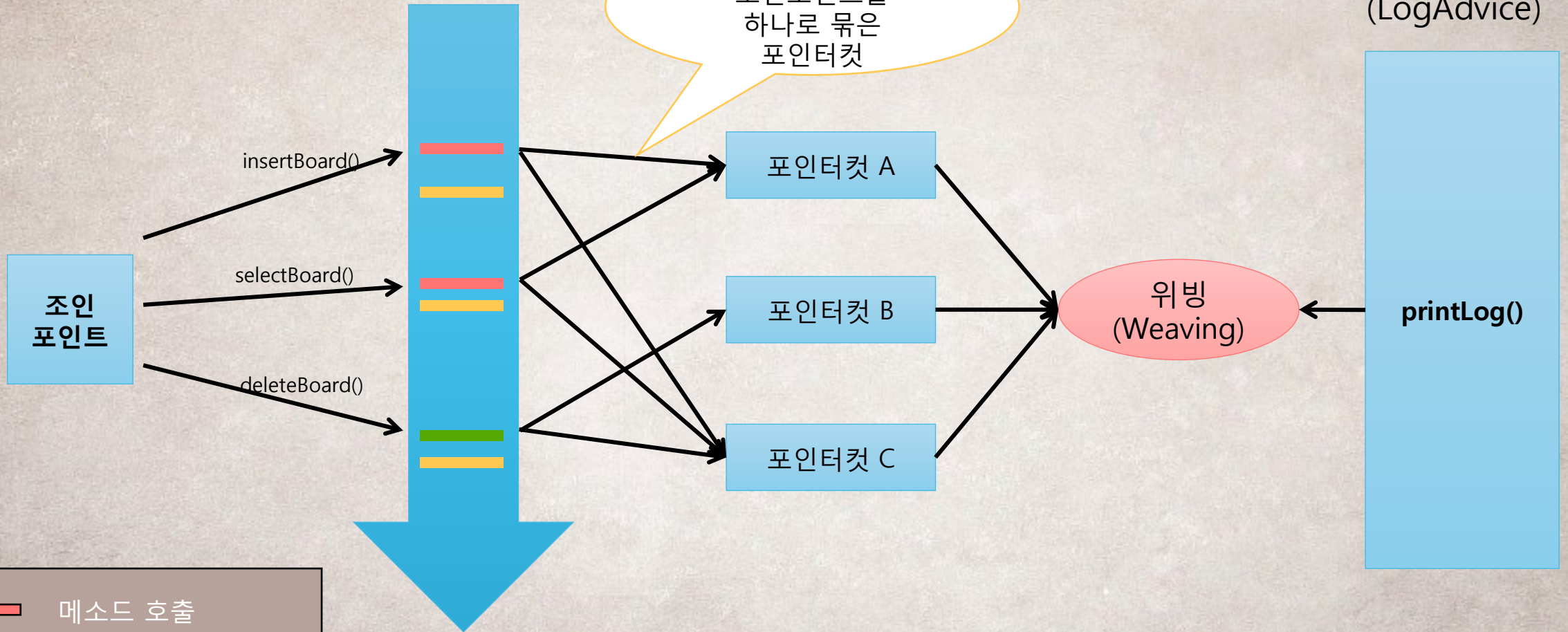


로그 출력(횡단 관심사)

# AOP 용어와 개념도

핵심관심  
(BoardServiceImpl)

횡단관심  
(LogAdvice)



- 메소드 호출
- 메소드 호출
- 로깅

# 용어 정리

# AOP(Aspect Oriented Programming) - 용어

용어	내용
어드바이스 (Advice)	실질적인 비즈니스 로직을 구현하고 있는 "핵심 비즈니스 로직 구현부"에 추가적인 기능을 제공하는 것( <b>횡단관심사</b> )
조인포인트 (Joinpoint)	클래스의 인스턴스 생성 시점, 메소드를 호출하는 시점, Exception이 발생하는 시점과 같이 어플리케이션을 실행할 때, 특정 작업이 실행되는 시점(비즈니스 매소드)을 의미한다.
포인트컷 (Pointcut)	분리된 기능(횡단관심사, 로깅 ...)들을 결합시키기 위한 규칙(패턴)이 필요한 데, 이와 같은 패턴을 포인트컷이라고 한다.
위빙 (Weaving)	"엮는다, 실로 짠다" 분리된 기능들을 하나로 결합시키는 것을 의미. 분리하여 개발된 기능들을 Weaving하는 작업을 도와주는 것이 AOP 툴이 하는 역할이다.
어스펙트 (Aspect)	어드바이스와 포인트컷을 합쳐서 하나의 Aspect라고 한다. 즉, 일정한 패턴을 가지는 클래스에 어드바이스를 적용하도록 지원할 수 있는 것을 Aspect라고 한다. 혹은 어드바이저(Advisor)라고 한다.
어드바이저 (Advisor)	어드바이스와 포인트컷을 하나로 묶어 다루는 것을 말한다.

# Advice 동작 시점 설정

- 어드바이스는 각 조인포인트에 삽입되어 동작할 횡단 관심에 해당하는 공통 기능이며, 동작 시점은 각 AOP 기술마다 다르며 5가지의 동작 시점을 제공

동작시점	설명
Before	비즈니스 메소드 실행 전 동작
After	<ul style="list-style-type: none"><li>- After Returning: 비즈니스 메소드가 성공적으로 리턴되면 동작</li><li>- After Throwing: 비즈니스 메소드 실행 중 예외가 발생하면 동작(마치 try~catch 블록에서 catch 블록에 해당)</li><li>- After: 비즈니스 메소드가 실행된 후, 무조건 실행(마치 try~catch~finally 블록에서 finally 블록에 해당)</li></ul>
Around	Around는 메소드 호출 자체를 가로채 비즈니스 메소드 실행 전 후에 처리할 로직을 삽입할 수 있음

# Before 어드바이스

- Before 어드바이스는 포인터컷으로 지정된 메소드 호출 시, 메소드가 실행되기 전에 처리될 내용들을 기술하기 위해 사용



Method 호출

Container

Before

```
insertBoard(){  
    비즈니스 로직 수행  
}
```

applicationContext.xml:

```
<aop:aspect ref="log">  
    <aop:before pointcut-ref="allPointcut" method="printLog"/>  
</aop:aspect>
```

# Before 어드바이스

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 13 오후 10:17:53)
9월 13, 2017 10:17:53 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:17:53 KST 2017]

[공통 로그] 비즈니스 로직 수행 전 동작
==> JDBC로 insertBoard() 기능 처리
[공통 로그] 비즈니스 로직 수행 전 동작
==> JDBC로 getBoardList() 기능 처리
---> BoardVO [seq=9, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=8, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=7, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=6, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=5, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=4, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=3, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=2, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다., regDate=2017-09-09, cnt=0]
9월 13, 2017 10:17:54 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:17:53 KST 2017]

BUILD SUCCESSFUL
```

- LogAdvice 클래스의 printLog() 메소드가 Before형태로 동작
  - 핵심 관심사 (insertBoard(), getBoard() 메소드) 수행 전에 횡단 관심사 (printLog() 메소드) 수행

# After Returning 어드바이스

- 포인트컷으로 지정된 메소드가 정상적으로 실행되고 나서, 메소드 수행 결과로 생성된 데이터를 리턴하는 시점에 동작



Method 호출

Container

```
insertBoard(){  
    비즈니스 로직 수행  
    return 데이터;  
}
```

After Returning

applicationContext.xml:

```
<aop:pointcut id="getPointcut" expression="execute(* kr.ac.inje.comsi..*impl.get*(..))" />  
<aop:aspect ref="log">  
    <aop:after-returning pointcut-ref="allPointcut" method="printLog"/>  
</aop:aspect>
```

# LogAdvice.java 수정

- LogAdvice의 printLog() 메소드 수정

```
JDBCUtil.java *LogAdvice.java applicationContext.xml
1 package kr.ac.inje.comsi.common;
2
3 public class LogAdvice {
4     public void printLog(){
5         System.out.println("[사후 처리] 비즈니스 로직 수행 후 동작");
6     }
7 }
8
```

# After Returning 어드바이스

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 13 오후 10:28:58)
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 13, 2017 10:28:58 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:28:58 KST 2017]; root of co

==> JDBC로 insertBoard() 기능 처리
==> JDBC로 getBoardList() 기능 처리
[사후 처리] 비즈니스 로직 수행 후 동작
---> BoardVO [seq=11, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=10, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=9, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=8, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=7, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=6, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=5, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=4, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=3, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=2, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다., regDate=2017-09-09, cnt=0]
9월 13, 2017 10:28:59 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:28:58 KST 2017]; root of conte

BUILD SUCCESSFUL

Total time: 1.199 secs
```

- LogAdvice 클래스의 printLog() 메소드가 After-Returning 형태로 동작
  - 포인트컷의 설정에 따라 핵심 관심사 (getBoard() 메소드) 수행 후에 횡단 관심사 (printLog() 메소드) 수행

# After Throwing 어드바이스

- 포인트컷으로 지정된 메소드가 실행되다가 예외가 발생하는 시점에 동작



Method 호출

Container

```
insertBoard(){  
    비즈니스 로직 수행  
    throw new Exception()  
}
```

After Throwing

applicationContext.xml:

```
<aop:pointcut id="allPointcut" expression="execute(* kr.ac.inje.comsi.*impl.*(..))" />  
<aop:aspect ref="log">  
    <aop:after-throwing pointcut-ref="allPointcut" method="printLog"/>  
</aop:aspect>
```

# BoardServiceImpl.java 수정

```
JDBCUtil.java  LogAdvice.java  applicationContext.xml  BoardDAO.java  BoardServiceImpl.java  BoardServiceClient.java

1 package kr.ac.inje.comsi.board.impl;
2
3 import java.util.List;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14     @Autowired
15     private BoardDAO boardDAO;
16
17     @Override
18     public void insertBoard(BoardVO vo) {
19         if (vo.getSeq() == 0) throw new IllegalArgumentException("0번 글은 등록할 수 없습니다.");
20         boardDAO.insertBoard(vo);
21     }
22
23     @Override
24     public void updateBoard(BoardVO vo) {
25         boardDAO.updateBoard(vo);
26     }
27
```

추가

# LogAdvice.java 수정

- LogAdvice의 printLog() 메소드 수정

```
JDBCUtil.java  LogAdvice.java  applicationContext.xml  BoardDAO.java  BoardServiceImpl.java  BoardServiceClient.java
1 package kr.ac.inje.comsi.common;
2
3 public class LogAdvice {
4     public void printLog(){
5         System.out.println("[예외 처리] 비즈니스 로직 수행 중 예외 발생");
6     }
7 }
8
```

# After Throwing 어드바이스

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 13 오후 10:40:23)
:run9월 13, 2017 10:40:23 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 13, 2017 10:40:24 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:40:24 KST 2017]; root of co

[예외 처리] 비즈니스 로직 수행 중 예외 발생
Exception in thread "main" java.lang.IllegalArgumentException: 0번 글은 등록할 수 없습니다.
    at kr.ac.inje.comsi.board.impl.BoardServiceImpl.insertBoard(BoardServiceImpl.java:19)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:333)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:190)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:157)
    at org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke(AspectJAfterThrowingAdvice.java:62)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
    at org.springframework.aop.interceptor.ExposeInvocationInterceptor.invoke(ExposeInvocationInterceptor.java:92)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
    at org.springframework.aop.framework.JdkDynamicAopProxy.invoke(JdkDynamicAopProxy.java:213)
    at com.sun.proxy.$Proxy9.insertBoard(Unknown Source)
    at kr.ac.inje.comsi.board.BoardServiceClient.main(BoardServiceClient.java:23)

:run FAILED
```

- LogAdvice 클래스의 printLog() 메소드가 After-throwing 형태로 동작
  - 포인트컷의 설정에 따라 핵심 관심사 수행 중 예외가 발생하여 횡단 관심사 (printLog() 메소드) 수행

# After 어드바이스

- try-catch-finally 구문에서 finally 블록처럼 예외 발생 여부에 상관없이 무조건 수행되는 어드바이스를 등록할 때



Method 호출

Container

```
insertBoard(){  
    비즈니스 로직 수행  
}
```

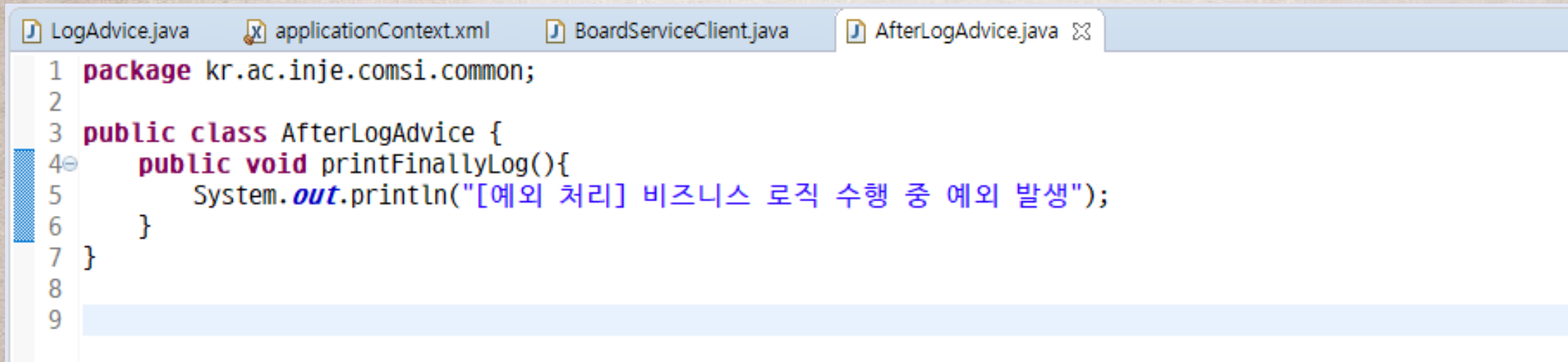
After

applicationContext.xml:

```
<aop:aspect ref="log">  
    <aop:after pointcut-ref="allPointcut" method="printLog"/>  
</aop:aspect>
```

# AfterLogAdvice.java 추가

- kr.ac.inje.comsi.common 패키지에 AfterLogAdvice 클래스 추가



```
LogAdvice.java applicationContext.xml BoardServiceClient.java AfterLogAdvice.java ✕
1 package kr.ac.inje.comsi.common;
2
3 public class AfterLogAdvice {
4     public void printFinallyLog(){
5         System.out.println("[예외 처리] 비즈니스 로직 수행 중 예외 발생");
6     }
7 }
8
9
```

# applicationContext.xml 파일 수정

LogAdvice.java applicationContext.xml BoardServiceClient.java AfterLogAdvice.java

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4     xmlns:p="http://www.springframework.org/schema/p"
5     xmlns:context="http://www.springframework.org/schema/context"
6     xmlns:aop="http://www.springframework.org/schema/aop"
7     xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8     http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9     http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12
13     <!-- LogAdvice bean 등록 -->
14     <bean id="log" class="kr.ac.inje.comsi.common.LogAdvice"></bean>
15     <bean id="afterLog" class="kr.ac.inje.comsi.common.AfterLogAdvice"></bean>
16
17     <!-- aop 관련 설정 -->
18     <aop:config>
19         <aop:pointcut expression="execution(* kr.ac.inje.comsi..*Impl.*(..))" id="allPointcut"/>
20         <aop:aspect ref="log">
21             <aop:after-throwing method="printLog" pointcut-ref="allPointcut"/>
22         </aop:aspect>
23
24         <aop:aspect ref="afterLog">
25             <aop:after method="printFinallyLog" pointcut-ref="allPointcut"/>
26         </aop:aspect>
27     </aop:config>
28 </beans>
29
```

추가

추가

# After 어드바이스

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 13 오후 10:49:27)
:processResources UP-TO-DATE
:classes
:run9월 13, 2017 10:49:28 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 13, 2017 10:49:28 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 22:49:28 KST 2017]; root of co

[사후 처리] 비즈니스 로직 수행 후 무조건 동작
Exception in thread "main" [예외 처리] 비즈니스 로직 수행 중 예외 발생
java.lang.IllegalArgumentException: 0번 글은 등록할 수 없습니다.
    at kr.ac.inje.comsi.board.impl.BoardServiceImpl.insertBoard(BoardServiceImpl.java:19)
    at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
    at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
    at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
    at java.lang.reflect.Method.invoke(Method.java:498)
    at org.springframework.aop.support.AopUtils.invokeJoinpointUsingReflection(AopUtils.java:333)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.invokeJoinpoint(ReflectiveMethodInvocation.java:190)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:157)
    at org.springframework.aop.aspectj.AspectJAfterAdvice.invoke(AspectJAfterAdvice.java:47)
    at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed(ReflectiveMethodInvocation.java:179)
    at org.springframework.aop.aspectj.AspectJAfterThrowingAdvice.invoke(AspectJAfterThrowingAdvice.java:62)
```

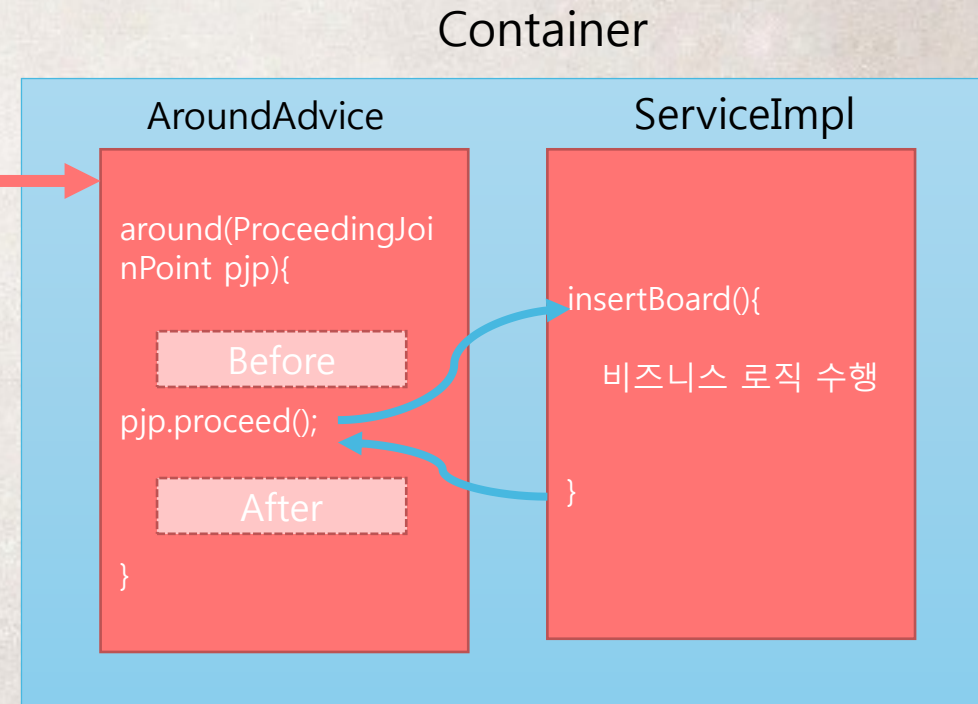
- AfterLogAdvice 클래스의 printFinallyLog () 메소드가 After 어드바이스 형태로 동작
  - 예외 발생후 printFinallyLog() 메소드가 실행

# Around 어드바이스

- Around 어드바이스는 클라이언트의 메소드 호출을 가로챈다.
- 어떤 상황에서는 하나의 어드바이스가 비즈니스 메소드 실행 전과 후에 모두 동작하여 로직을 처리하는 경우 있을 때 사용



Method 호출



applicationContext.xml:

```
<aop:aspect ref="log">  
    <aop:around pointcut-ref="allPointcut" method="printLog"/>  
</aop:aspect>
```

# AroundAdvice.java 추가

- kr.ac.inje.comsi.common 패키지에 AroundAdvice 클래스 추가

```
applicationContext.xml  AroundAdvice.java ✖
1 package kr.ac.inje.comsi.common;
2
3 import org.aspectj.lang.ProceedingJoinPoint;
4
5 public class AroundAdvice {
6
7     public Object aroundLog(ProceedingJoinPoint pjp) throws Throwable{
8         System.out.println("[Before]: 비즈니스 메소드 수행 전에 처리할 내용...");
9         Object returnObj = pjp.proceed();
10        System.out.println("[After]: 비즈니스 메소드 수행 후에 처리할 내용...");
11
12        return returnObj;
13    }
14 }
15
16
17 |
```

- 클라이언트의 요청을 가로챈 어드바이스는 클라이언트가 호출한 비즈니스 메소드를 호출하기 위해서 ProceedingJoinPoint 객체를 매개 변수로 받음
- ProceedingJoinPoint 객체의 proceed() 메소드를 통해서 비즈니스 메소드를 호출함

# BoardServiceImpl.java 수정

```
JDBCUtil.java  LogAdvice.java  applicationContext.xml  BoardDAO.java  BoardServiceImpl.java  BoardServiceClient.java

1 package kr.ac.inje.comsi.board.impl;
2
3 import java.util.List;
10
11 @Service("boardService")
12 public class BoardServiceImpl implements BoardService {
13
14     @Autowired
15     private BoardDAO boardDAO;
16
17     @Override
18     public void insertBoard(BoardVO vo) {
19         if (vo.getSeq() == 0) throw new IllegalArgumentException("0번 글은 등록할 수 없습니다.");
20         boardDAO.insertBoard(vo);
21     }
22
23     @Override
24     public void updateBoard(BoardVO vo) {
25         boardDAO.updateBoard(vo);
26     }
27
```

제거

# applicationContext.xml 수정

applicationContext.xml AroundAdvice.java BoardServiceImpl.java

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <beans xmlns="http://www.springframework.org/schema/beans"
3       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4       xmlns:p="http://www.springframework.org/schema/p"
5       xmlns:context="http://www.springframework.org/schema/context"
6       xmlns:aop="http://www.springframework.org/schema/aop"
7       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
8                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-4.3.xsd
9                           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd">
10
11     <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>
12
13     <!-- LogAdvice bean 등록 -->
14     <bean id="log" class="kr.ac.inje.comsi.common.LogAdvice"></bean>
15     <bean id="around" class="kr.ac.inje.comsi.common.AroundAdvice"></bean>
16     <!-- aop 관련 설정 -->
17     <aop:config>
18         <aop:pointcut expression="execution(* kr.ac.inje.comsi..*Impl.*(..))" id="allPointcut"/>
19         <aop:aspect ref="around">
20             <aop:around method="aroundLog" pointcut-ref="allPointcut"/>
21         </aop:aspect>
22     </aop:config>
23 </beans>
24
25
```

기존 Bean 변경

ref와 method 수정

# Around 어드바이스

```
BoardWeb - run [Gradle Project] run in D:\workspace\java\2017_1\BoardWeb (2017. 9. 13 오후 11:07:16)
Gradle Tasks: Full

:compileJava UP-TO-DATE
:processResources
:classes
:run9월 13, 2017 11:07:17 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
9월 13, 2017 11:07:17 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed Sep 13 23:07:17 KST 2017]; root of co

[Before]: 비즈니스 메소드 수행 전에 처리할 내용... ←
==> JDBC로 insertBoard() 기능 처리
[After]: 비즈니스 메소드 수행 후에 처리할 내용... ←
[Before]: 비즈니스 메소드 수행 전에 처리할 내용... ←
==> JDBC로 getBoardList() 기능 처리
[After]: 비즈니스 메소드 수행 후에 처리할 내용... ←
---> BoardVO [seq=12, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=11, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=10, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=9, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=8, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-13, cnt=0]
---> BoardVO [seq=7, title=AOP실행, writer=홍길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=6, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=5, title=AOP실행, writer=김길동2, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=4, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
---> BoardVO [seq=3, title=AOP실행, writer=김길동, content=임시 내용 ....., regDate=2017-09-09, cnt=0]
```

- AroundLogAdvice 클래스의 aroundLog () 메소드가 동작
  - 비즈니스 로직 호출 후후에 메시지 출력