

4장. 스프링 AOP



AOP(Aspect Oriented Programming)



◆ AOP(관점 지향 프로그래밍)란?

스프링의 DI(의존성 주입)가 결합도와 관련된 기능이라면 AOP(Aspect Oriented Programming)는 **응집도**와 관련된 기능이라 할 수 있음

애플리케이션의 메소드들은 다음과 같이 **핵심 비즈니스 로직**은 몇 줄 안되고, 주로 로깅이나 예외, 트랜잭션 처리 같은 **부가적인 코드**가 대부분이다.

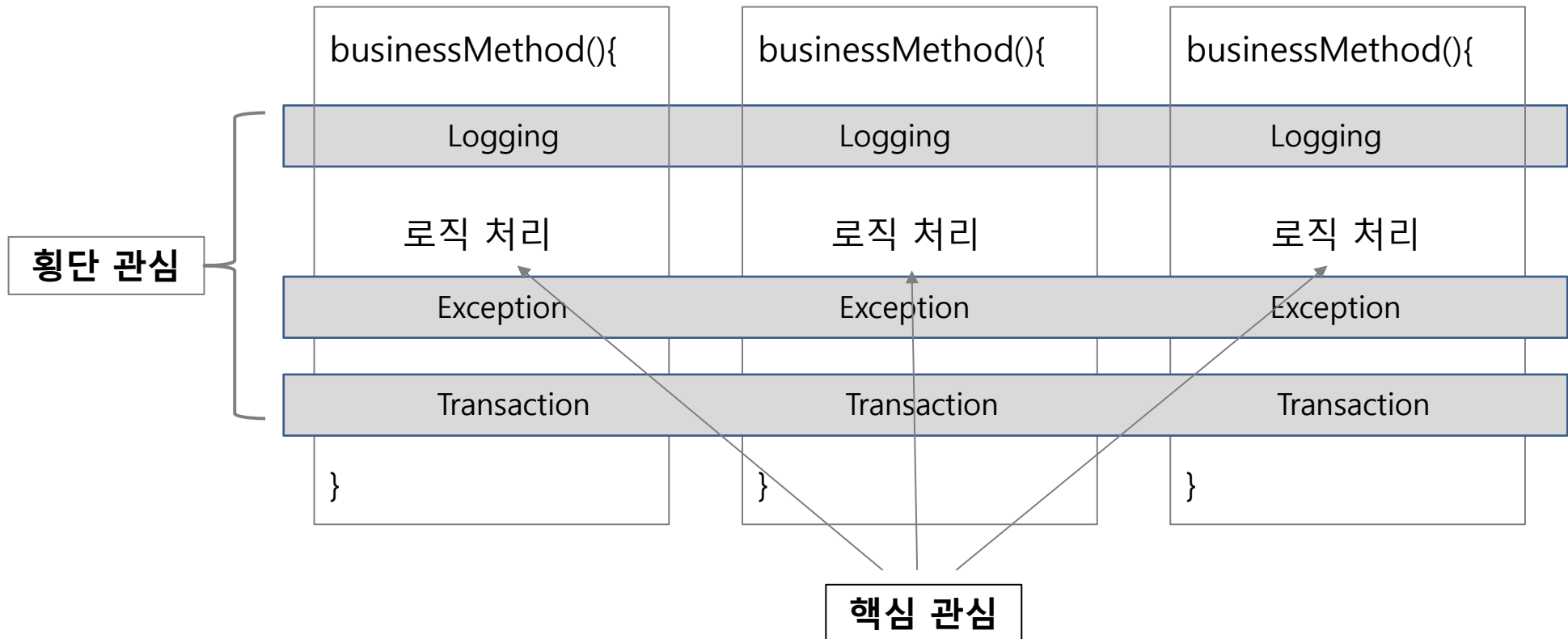
AOP에서 메소드 마다 등장하는 로깅이나 예외, 트랜잭션 처리 같은 코드들을 **횡단 관심** (Crosscutting Concerns)이라 하고, 실제로 수행되는 핵심 비즈니스 로직을 **핵심 관심** (Core Concerns)라고 한다.

```
businessMethod(){  
    Logging...  
    비즈니스 로직(3~5라인)  
    Exception Handle...  
    Transaction Handle...  
}
```



스프링 AOP

◆ AOP란?



◆ 로깅 실습 – SpringAOP 프로젝트

servlet-context.xml 스캔 범위 조정

```
<!-- Resolves views selected for rendering by @Controllers to  
<beans:bean class="org.springframework.web.servlet.view.Intern  
    <beans:property name="prefix" value="/WEB-INF/views/" />  
    <beans:property name="suffix" value=".jsp" />  
</beans:bean>  
  
<context:component-scan base-package="com.spring" />
```



스프링 AOP

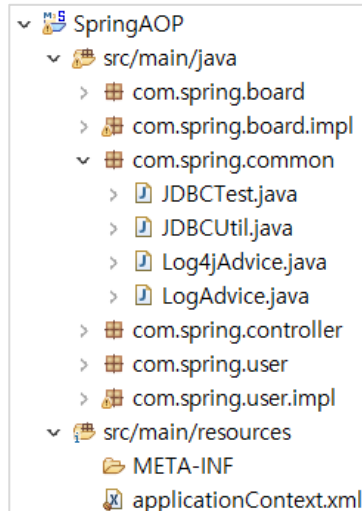
◆ 로깅 실습 – SpringAOP 프로젝트

applicationContext.xml 스캔 범위 등록

```
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xmlns:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/context http://www.springframework.org/schema/context
    http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop" >

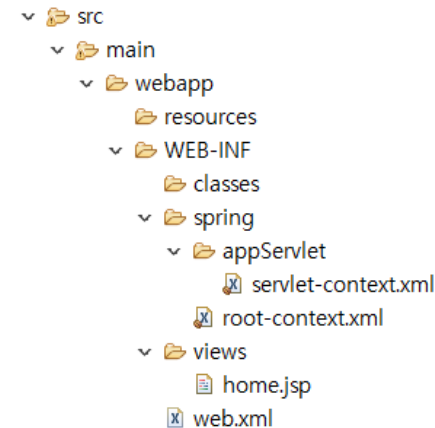
  <context:component-scan base-package="com.spring" />

</beans>
```



SpringAOP

- src/main/java
 - com.spring.board
 - com.spring.board.impl
 - com.spring.common
 - JDBCTest.java
 - JDBCUtil.java
 - Log4jAdvice.java
 - LogAdvice.java
 - com.spring.controller
 - com.spring.user
 - com.spring.user.impl
- src/main/resources
 - META-INF
 - applicationContext.xml



src

- main
 - webapp
 - resources
 - WEB-INF
 - classes
 - spring
 - appServlet
 - servlet-context.xml
 - root-context.xml
 - views
 - home.jsp
 - web.xml



스프링 AOP

◆ 로깅 실습 – SpringAOP 프로젝트

servlet-context.xml 스캔 범위 조정

```
package com.spring.common;

public class LogAdvice {
    public void printLog() {
        System.out.println("[공통 로그] 비즈니스 로직 수행 전 동작");
    }
}
```



스프링 AOP

```
@Service("boardService")
public class BoardServiceImpl implements BoardService{

    @Autowired
    private BoardDAO boardDAO;

    private LogAdvice log;

    public BoardServiceImpl() {
        log = new LogAdvice();
    }

    @Override
    public void insertBoard(BoardVO vo) {
        log.printLog();
        boardDAO.insertBoard(vo);
    }

    @Override
    public List<BoardVO> getBoardList() {
        log.printLog();
        return boardDAO.getBoardList();
    }
}
```

BoardServiceImpl 객체가 생성될때 생성
자에서 LogAdvice 객체도 같이 생성한다.



◆ 로깅 실습 – SpringAOP 프로젝트

```
public class BoardServiceClient {

    public static void main(String[] args) {
        //1. spring 컨테이너 구동
        AbstractApplicationContext container =
            new GenericXmlApplicationContext("applicationContext.xml");

        //2. BoardServiceImpl 객체를 Lookup
        BoardService boardService = (BoardService) container.getBean("boardService");

        //3. 글 등록 기능 테스트
        BoardVO vo = new BoardVO();
        vo.setTitle("안녕하세요");
        vo.setWriter("김경희");
        vo.setContent("지인 추천으로 가입했습니다.");
        boardService.insertBoard(vo);

        //4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList();
        for(BoardVO board : boardList) {
            System.out.println("---> " + board.toString());
        }

        //5. spring 컨테이너 종료
        container.close();
    }
}
```



◆ 로깅 실습 – SpringAOP 프로젝트

```
[공통 로그] 비즈니스 로직 수행 전 동작
==> insertBoard()
[공통 로그] 비즈니스 로직 수행 전 동작
==> getBoardList()
---> BoardVO [bno=4, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,
---> BoardVO [bno=3, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,
---> BoardVO [bno=2, title=안녕하세요, writer=하이미디어, content=지인 추천으로 가입했습니다
---> BoardVO [bno=1, title=가입인사, writer=관리자, content=잘 부탁드립니다..., regDate:
```



스프링 AOP

```
@Service("boardService")
public class BoardServiceImpl implements BoardService{

    @Autowired
    private BoardDAO boardDAO;

    //private LogAdvice log;
    private Log4jAdvice log;

    public BoardServiceImpl() {
        //log = new LogAdvice();
        log = new Log4jAdvice();
    }

    @Override
    public void insertBoard(BoardVO vo) {
        //log.printLog();
        log.printLogging();
        boardDAO.insertBoard(vo);
    }

    @Override
    public List<BoardVO> getBoardList() {
        //log.printLog();
        log.printLogging();
        return boardDAO.getBoardList();
    }
}
```



스프링 AOP

Advice 클래스가 LogAdvice에서 Log4jAdvice로 바뀌는 순간 BoardServiceImpl 클래스의 생성자를 수정해야 한다. 그리고 printLog()가 printLogging() 메소드로 변경되었으므로 모두 수정해야 한다.

```
[공통 로그-Log4j] 비즈니스 로직 수행 전 동작
```

```
==> insertBoard()
```

```
[공통 로그-Log4j] 비즈니스 로직 수행 전 동작
```

```
==> getBoardList()
```

```
---> BoardVO [bno=5, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,  
---> BoardVO [bno=4, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,  
---> BoardVO [bno=3, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,  
---> BoardVO [bno=2, title=안녕하세요, writer=하이미디어, content=지인 추천으로 가입했습니다,  
---> BoardVO [bno=1, title=가입인사, writer=관리자, content=잘 부탁드립니다..., regDate:
```



스프링 AOP

◆ AOP 적용하기 - AOP 라이브러리 추가

```
<!-- AspectJ -->
<dependency>
  <groupId>org.aspectj</groupId>
  <artifactId>aspectjrt</artifactId>
  <version>${org.aspectj-version}</version>
</dependency>
<!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver -->
<dependency>
  <groupId>org.aspectj</groupId>
  <artifactId>aspectjweaver</artifactId>
  <version>1.9.2</version>
  <scope>runtime</scope>
</dependency>
```

▼ Maven Dependencies

- > aspectjrt-1.6.10.jar - C:\Users\Wkiyon\m2\repository\org\aspectj\aspectjrt\1.6.10
- > aspectjweaver-1.9.2.jar - C:\Users\Wkiyon\m2\repository\org\aspectj\aspectjweaver\1.9.2














스프링 AOP

◆ 네임스페이스 추가 – aop 체크

Namespaces

Configure Namespaces

Select XSD namespaces to use in the configuration file

<input checked="" type="checkbox"/>		aop - http://www.springframework.org/schema/aop
<input checked="" type="checkbox"/>		beans - http://www.springframework.org/schema/beans
<input type="checkbox"/>		c - http://www.springframework.org/schema/c
<input type="checkbox"/>		cache - http://www.springframework.org/schema/cache
<input checked="" type="checkbox"/>		context - http://www.springframework.org/schema/context
<input type="checkbox"/>		jee - http://www.springframework.org/schema/jee
<input type="checkbox"/>		lang - http://www.springframework.org/schema/lang
<input type="checkbox"/>		mvc - http://www.springframework.org/schema/mvc
<input type="checkbox"/>		p - http://www.springframework.org/schema/p
<input type="checkbox"/>		task - http://www.springframework.org/schema/task
<input type="checkbox"/>		util - http://www.springframework.org/schema/util

Source

Namespaces

Overview

aop

beans

context



스프링 AOP

◆ AOP 적용하기

```
<context:component-scan base-package="com.spring" />

<bean id="log" class="com.spring.common.LogAdvice"></bean>
<aop:config>
    <aop:pointcut expression="execution(* com.spring..*Impl.*(..))" id="allPointcut"/>
    <aop:aspect ref="log">
        <aop:before method="printLog" pointcut-ref="allPointcut"/>
    </aop:aspect>
</aop:config>
```



스프링 AOP

◆ AOP 적용하기

```
@Service("boardService")
public class BoardServiceImpl implements BoardService{

    @Autowired
    private BoardDAO boardDAO;

    //private LogAdvice log;
    //private Log4jAdvice log;

    public BoardServiceImpl() {
        //log = new LogAdvice();
        //log = new Log4jAdvice();
    }

    @Override
    public void insertBoard(BoardVO vo) {
        //log.printLog();
        //log.printLogging();
        boardDAO.insertBoard(vo);
    }
}
```



스프링 AOP

◆ AOP 적용하기

```
[공통 로그] 비즈니스 로직 수행 전 동작
```

```
==> insertBoard()
```

```
[공통 로그] 비즈니스 로직 수행 전 동작
```

```
==> getBoardList()
```

```
---> BoardVO [bno=7, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,
```

```
---> BoardVO [bno=6, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,
```

```
---> BoardVO [bno=5, title=안녕하세요, writer=김경희, content=지인 추천으로 가입했습니다.,
```



스프링 AOP

◆ LogAdvice -> Log4jAdvice로 변경

```
<!-- <bean id="log" class="com.spring.common.LogAdvice"></bean> -->
<bean id="log" class="com.spring.common.Log4jAdvice"></bean>
<aop:config>
  <aop:pointcut expression="execution(* com.spring..*Impl.*(..))" id="allPointcut"/>
  <aop:aspect ref="log">
    <!-- <aop:before method="printLog" pointcut-ref="allPointcut"/> -->
    <aop:before method="printLogging" pointcut-ref="allPointcut"/>
  </aop:aspect>
</aop:config>
```

[공통 로그-Log4j] 비즈니스 로직 수행 전 동작

==> insertBoard()

[공통 로그-Log4j] 비즈니스 로직 수행 전 동작

==> getBoardList()

---> BoardVO [bno=8, title=안녕하세요, writer=김경희, content=지인 추천으로

---> BoardVO [bno=7, title=안녕하세요, writer=김경희, content=지인 추천으로

---> BoardVO [bno=6, title=안녕하세요, writer=김경희, content=지인 추천으로

