# 20장 스프링 AOP 기능

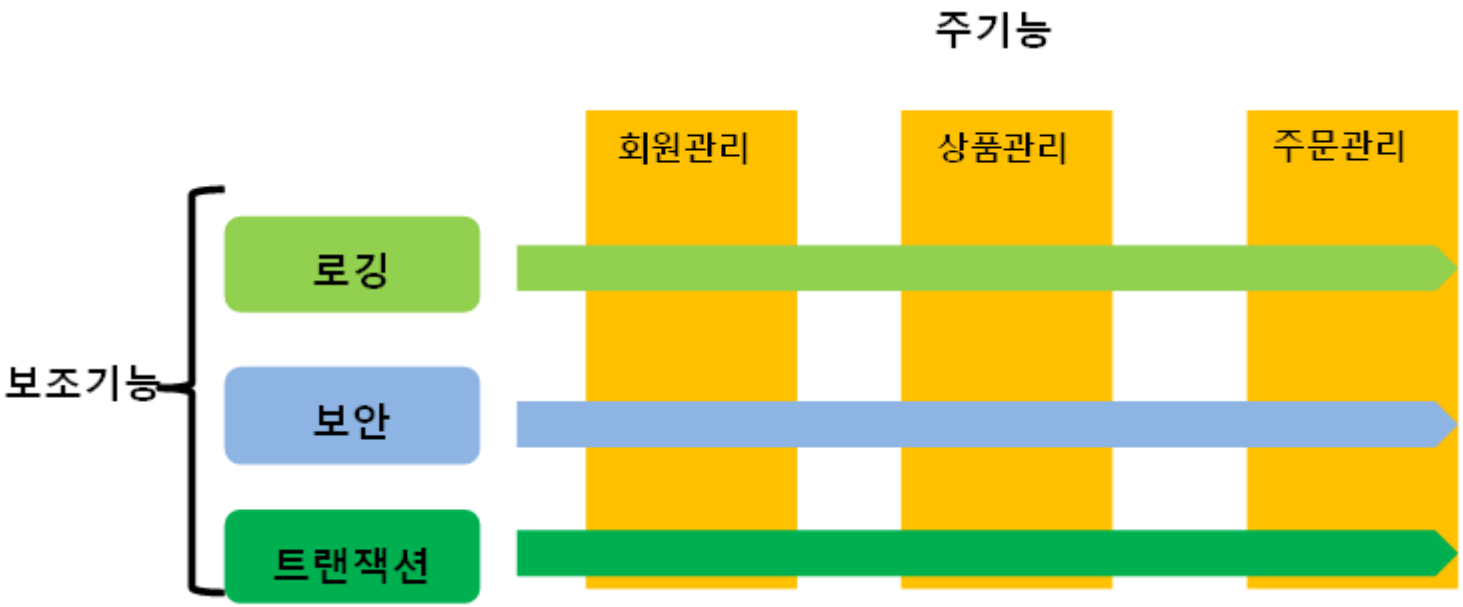
# 20.1 관점 지향 프로그래밍의 등장

- 주기능인 회원 등급 기능 구현 시 로깅 기능, 보안 기능, 트랜잭션 기능등의 보조 기능을 일일이 구현해야함.
- 규모가 있는 웹 애플리케이션일 경우 클래스의 메서드마다 이런 작업을 일일이 수작업으로 하기에는 시간도 많이 걸리고 소스 코드도 복잡해짐
- 즉, 유지관리에 문제가 생길 수 있음

- 관점 지향 프로그래밍(Aspect Oriented Programming, AOP)를 이용해서 주기능과 보조 기능을 분리해서 메서드에 적용함

# 20.1 관점 지향 프로그래밍의 등장

AOP를 적용한 쇼핑몰 구조도



# 20.2 스프링에서 AOP 기능 사용하기

**여러 가지 AOP 관련 용어**

| 용어 | 설명 |
|---|---|
| aspect | 구현하고자 하는 보조 기능을 의미합니다. |
| advice | aspect의 실제 구현체(클래스)를 의미합니다. 메서드 호출을 기준으로 여러 지점에서 실행됩니다. |
| joinpoint | advice를 적용하는 지점을 의미합니다. 스프링은 method 결합점만 제공합니다 |
| pointcut | advice가 적용되는 대상을 지정합니다. 패키지이름/클래스이름/메서드이름을 정규식으로 지정하여 사용합니다. |
| target | advice가 적용되는 클래스를 의미합니다. |
| weaving | advice를 주기능에 적용하는 것을 의미합니다. |

**스프링 프레임워크에서 AOP 기능을 구현하는 방법**

- 스프링 프레임워크에서 제공하는 API를 이용하는 방법
- @Aspect 애너테이션을 이용하는 방법

# 20.2 스프링에서 AOP 기능 사용하기

- ## 20.2.1 스프링 API를 이용한 AOP 기능 구현 과정

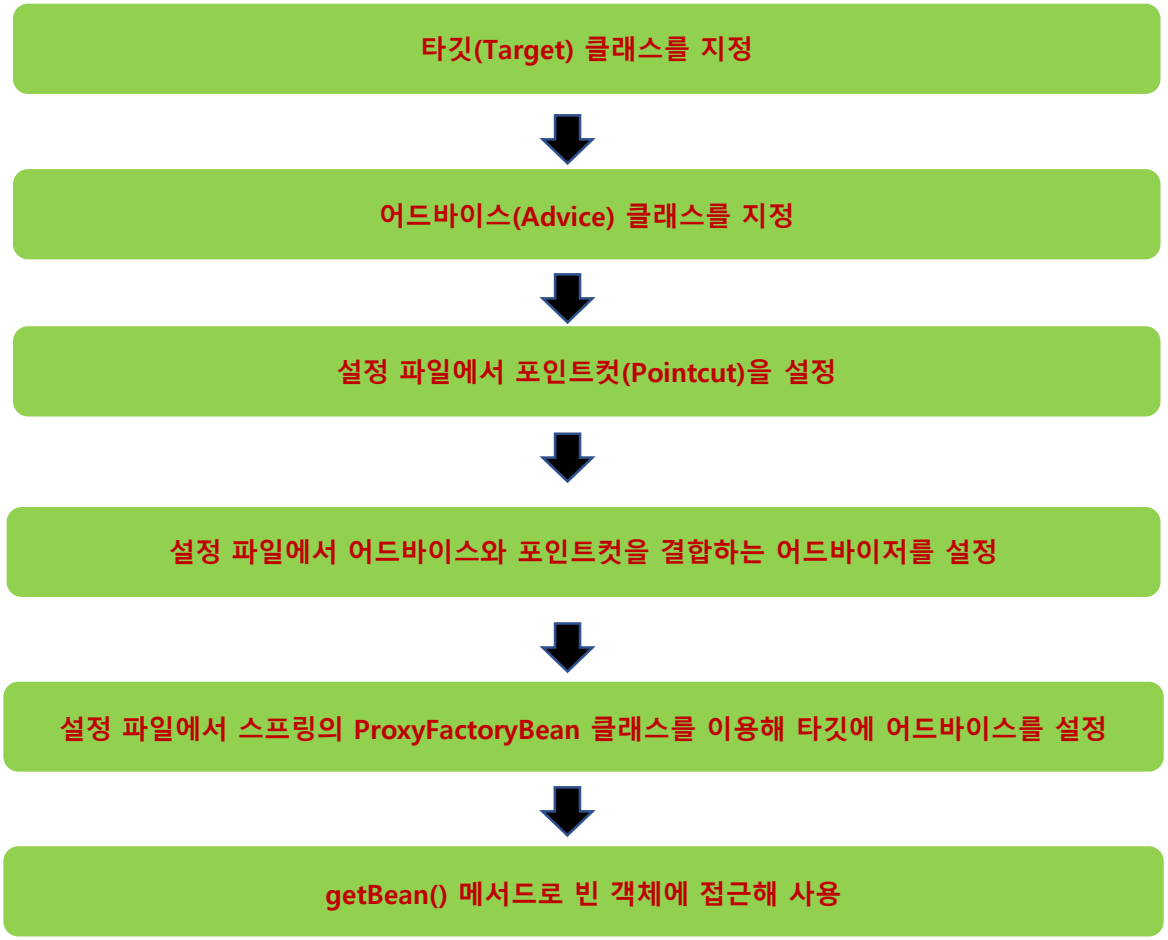
**AOP 기능 구현 과정**

① 타깃(Target) 클래스를 지정
② 어드바이스(Advice) 클래스를 지정
③ 설정 파일에서 포인트컷(Pointcut)을 설정
④ 설정 파일에서 어드바이스와 포인트컷을 결합하는 어드바이저를 설정
⑤ 설정 파일에서 스프링의 ProxyFactoryBean 클래스를 이용해 타깃에 어드바이스를 설정
⑥ getBean() 메서드로 빈 객체에 접근해 사용

# 20.2 스프링에서 AOP 기능 사용하기

- ## 20.2.1 스프링 API를 이용한 AOP 기능 구현 과정

**AOP 기능 구현 과정**



# 20.2 스프링에서 AOP 기능 사용하기

**스프링 API에서 제공하는 여러 가지 Advice 인터페이스**

| 인터페이스 | 추상 메서드 | 설명 |
|---|---|---|
| MethodBeforeAdvice | before(Method method,<br>Object[] args,<br>Object target)<br>throws Throwable | 해당 메서드가 실행되기 전 실행 |
| • Method method: 대상 객체에서 실행된 메서드를 나타내는 메서드 객체<br>• Object[] args: 메서드 인자 목록<br>• Object target: 대상 객체 | | |
| AfterReturningAdvice | void afterReturning(<br>Object returnValue,<br>Method method,<br>Object[] args,<br>Object target)<br>throws Throwable | 해당 메서드가 실행된 후 실행 |
| • Object returnValue : 대상 객체의 메서드가 반환하는 값<br>• Method method: 대상 객체에서 실행된 메서드를 나타내는 메서드 객체<br>• Object[] args: 메서드 인자 목록<br>• Object target: 대상 객체 | | |
| ThrowsAdvice | void afterThrowing(<br>Method method,<br>Object[] args,<br>Object target,<br>Exception ex) | 해당 메서드에서 예외 발생 시 실행 |
| • Method method: 대상 객체에서 메서드를 나타내는 메서드 객체<br>• Object[] args: 메서드 인자 목록<br>• Object target: 대상 객체<br>• Exception ex: 발생한 예외 타입 | | |

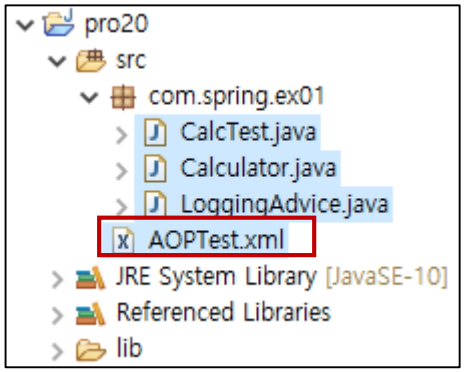
# 20.2 스프링에서 AOP 기능 사용하기

| 인터페이스 | 추상 메서드 | 설명 |
|---|---|---|
| MethodInterceptor | Object invoke(<br>MethodInvocation<br>invocation)<br>throws Throwable | 해당 메서드의 실행 전/후와 예외<br>발생 시 실행 |
| MethodInvocation invocation: 대상 객체의 모든 정보를 담고 있는 객체(호출된 메서드, 인자 등) | | |

# 20.2 스프링에서 AOP 기능 사용하기

- **20.2.2 스프링 API를 이용한 AOP 기능 실습**

**1.** 새 프로젝트 pro20을 만들고 lib 폴더를 만들어 라이브러리 클래스 패스를 설정합니다.(19장참고)
그리고 AOP 설정 파일인 AOPTest.xml를 src 패키지에 생성합니다.



# 20.2 스프링에서 AOP 기능 사용하기

**2.** AOP를 설정하는 AOPTest.xml을 다음과 같이 작성합니다.

**코드 20-2** pro20/src/AOPTest.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
  "http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans>
 <bean id="calcTarget" class="com.spring.ex01.Calculator" />
 <bean id="logAdvice" class="com.spring.ex01.LoggingAdvice" />
 <bean id="proxyCal" class="org.springframework.aop.framework.ProxyFactoryBean">
   <property name="target" ref="calcTarget"/>
   <property name="interceptorNames">
     <list>
       <value>logAdvice</value>
     </list>
   </property>
  </bean>
</beans>
```

타깃 클래스 빈을 지정합니다.

로그 기능을 하는 어드바이스 빈을 지정합니다.

스프링 프레임워크에서 제공하는 ProxyFactoryBean을 이용해 타깃과 어드바이스를 엮어줍니다.

타깃 빈을 calcTarget 빈으로 지정합니다.

스프링의 ProxyFactoryBean의 interceptorNames 속성에 logAdvice를 어드바이스 빈으로 설정하여 타깃 클래스의 메서드 호출 시 logAdvice를 실행합니다.

# 20.2 스프링에서 AOP 기능 사용하기

**3.** 이번에는 타깃 클래스인 Calculator 클래스를 작성합니다.

**코드 20-3** pro20/src/com.spring/ex01/Calculator.java

```
package com.spring.ex01;

public class Calculator
{
  public void add(int x, int y)
  {
    int result = x + y;
    System.out.println("결과:" + result);
  }

  public void subtract(int x, int y)
  {
    int result = x - y;
    System.out.println("결과:" + result);
  }

  public void multiply(int x, int y)
  {
    int result = x * y;
    System.out.println("결과:" + result);
  }
  public void divide(int x, int y)
  {
    int result = x / y;
    System.out.println("결과:" + result);
  }
}
```

# 20.2 스프링에서 AOP 기능 사용하기

**4.** 어드바이스 클래스인 LoggingAdvice를 다음과 같이 작성합니다.

**코드 20-4** pro20/src/com.spring/ex01/LoggingAdvice.java

```
package com.spring.ex01;

import org.aopalliance.intercept.MethodInterceptor;
import org.aopalliance.intercept.MethodInvocation;
public class LoggingAdvice implements MethodInterceptor
{
  public Object invoke(MethodInvocation invocation) throws Throwable
  {
    System.out.println("[메서드 호출 전 : LogginAdvice");
    System.out.println(invocation.getMethod() + "메서드 호출 전");

    Object object = invocation.proceed();

    System.out.println("[메서드 호출 후 : loggingAdvice");
    System.out.println(invocation.getMethod() + "메서드 호출 후");
    return object;
  }
}
```

인터페이스 MethodInterceptor를 구현해 어드바이스 클래스를 만듭니다.

메서드 호출 전에 수행하는 구문입니다.

invocation을 이용해 메서드를 호출합니다.

메서드 호출 후에 수행하는 구문입니다.

# 20.2 스프링에서 AOP 기능 사용하기

**5.** 실행 클래스인 CalcTest를 다음과 같이 작성합니다.

**코드 20-5** pro20/src/com.spring/ex01/CalcTest.java

```
package com.spring.ex01;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class CalcTest
{
  public static void main(String[] args)
  {
    ApplicationContext context = new ClassPathXmlApplicationContext("AOPTest.xml");
    Calculator cal = (Calculator) context.getBean("proxyCal");
    cal.add(100, 20);
    System.out.println();
    cal.subtract(100, 20);
    System.out.println();
    cal.multiply(100, 20);
    System.out.println();
    cal.divide(100, 20);
  }
}
```

AOPTest.xml을 읽어 들여 빈을 생성합니다.

id가 proxyCal인 빈에 접근합니다.

메서드 호출 전후에 어드바이스 빈이 적용됩니다.

# 20.2 스프링에서 AOP 기능 사용하기

**6.** main() 메서드가 있는 실행 클래스(CalcTest.java)가 보이는 상태에서 실행 버튼을 클릭해 실행합니다.

**add() 메서드로 결과 출력 전,후 add 메서드 정보를 출력합니다.**

```
[메서드 호출 전 : LogginAdvice
public void com.spring.ex01.Calculator.add(int,int)메서드 호출 전
결과:120
[메서드 호출 후 : loggingAdvice
public void com.spring.ex01.Calculator.add(int,int)메서드 호출 후

[메서드 호출 전 : LogginAdvice
public void com.spring.ex01.Calculator.subtract(int,int)메서드 호출 전
결과:80
[메서드 호출 후 : loggingAdvice
public void com.spring.ex01.Calculator.subtract(int,int)메서드 호출 후
```

# 20.2 스프링에서 AOP 기능 사용하기

**퍼스펙티브 변경하기**