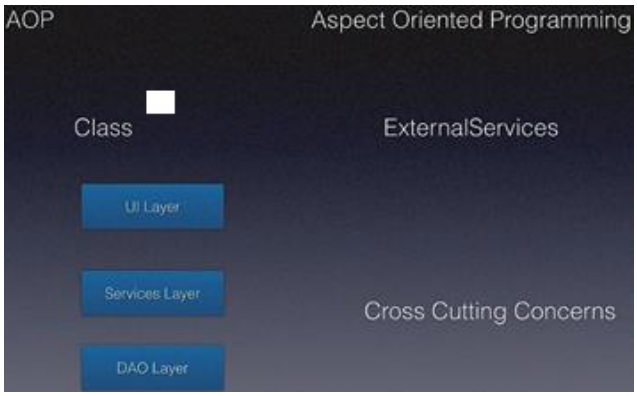
1. 소개: **A**spect **O**riented **P**rogramming

가. 개요

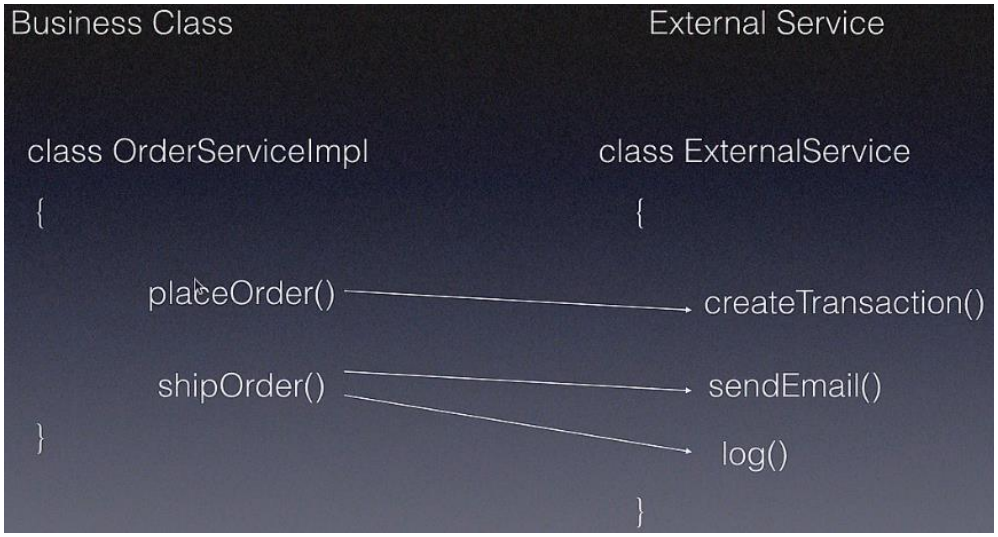
- 트랜잭션 관리나 로깅, 시큐리티 등등 같은 서비스나 외부서비스를 **클래스내의 코드나 메소드를 변경하지 않고** 우리의 어플리케이션 프로세스에 적용하는 과정
- 외부서비스들은 그들이 자바EE 웹 어플리케이션 전체 걸쳐 적용되기 때문에, CROSS-CUTTING CONCERNS(범 분야적 관심==공통관심) 이라고도 불린다.
- 우리의 어플리케이션 내에서 일어나는 것을 로그파일에 로깅한다.
- 트랜잭션 관리는 전형적으로 서비스 레이어에 적용된다.
- 이러한 전체 과정을 Aspect Oriented Programming이라 부름

나. 이미지



다. 예제

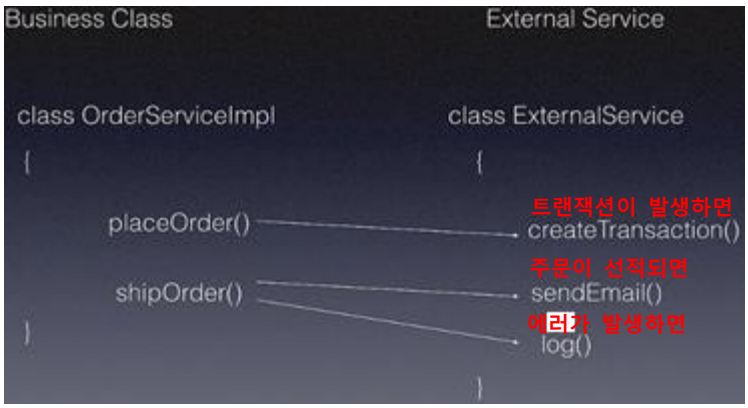
- 트랜잭션 생성하기
- 주문에서 배에 적재한 다음, 이메일을 보내기를 원함
- 서비스 클래스에서 일어나고 있는 일을 로그 파일에 로깅하기를 원함



- 외부서비스를 서비스 클래스에 Aspect Oriented Programming 이용해서 적용될 수 있음

2. AOP 관련 용어들

가. AOP에서의 Key terms



나. ASPECT: **외부 서비스(External Service) 클래스**

- 모든 외부 서비스들이 aspect에 포함될 필요는 없음

다. Advice

- **Aspect 클래스 안에 정의된 메소드**
- Aspect의 구현체라고 부를 수 있음

라. PointCut

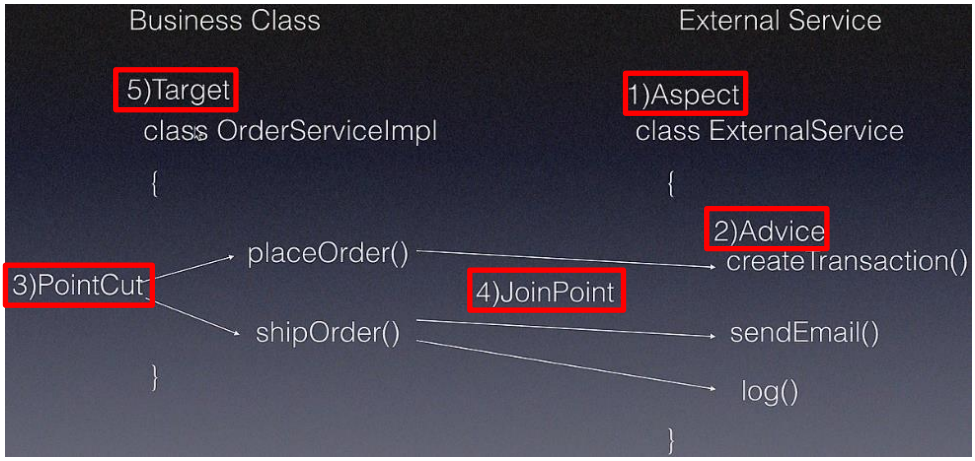
- **표현언어**(Expression language) 제공
- 어플리케이션 내, **어떤 메소드가 Advisier를 필요로 하는지** 알려주는 표현(expression)

마. JointPoint

- 하지만, PointCut은 어플리케이션 내 어떤 문제들이 어드바이저를 필요로 하는지 표현하는 표현식일 뿐 결정하지는 않음
- 어떤 문제에 어떤 어드바이저를 필요한지 결정
- **Advice와 PointCut의 Combination**

바. Target: **Advise를 필요로 하는 비즈니스 오브젝트**

사. 개념도

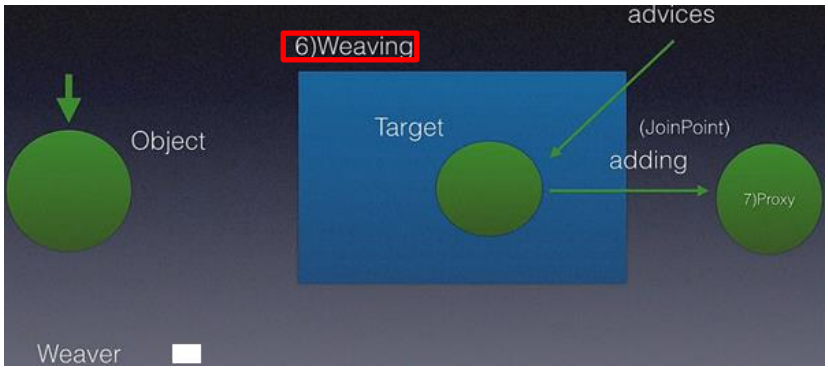


아. 위빙(Weaving)

- Target 오브젝트는 Aspect 오브젝트를 필요로 함
- 조인트포인트(JointPoint) 기반으로 어드바이즈와 타겟오브젝트 섞어주는 프로세스
- 즉, **조인트포인트**를 사용해 **Advice**를 **Target 오브젝트의 메소드**에 적용하는 프로세스

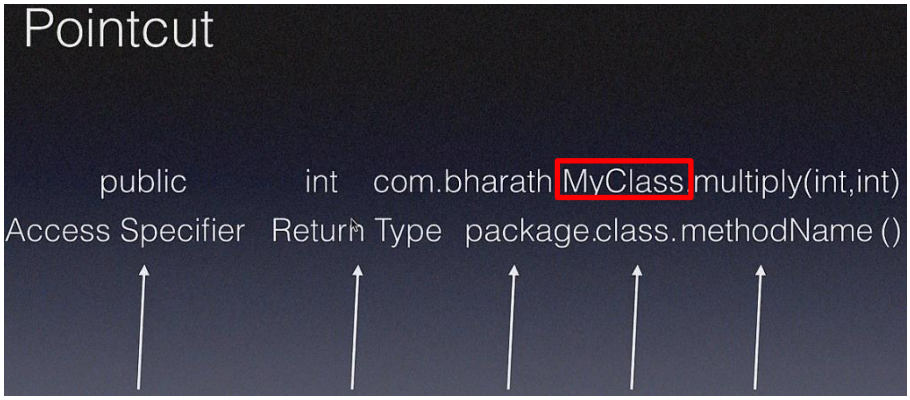
자. 프록시(Proxy)

- 프록시란 **Weaing** 프로세스의 결과로 생성되는 클래스
- 프록시는 Advice 로직뿐 아니라 business 로직 메소드를 모두 포함하고 있음
- 프록시 클래스를 적용하는데 필요한 어드바이스와 비즈니스 로직의 조합
- **Weaving**은 **Weaver**라는 특별한 컴포넌트를 통해 이루어짐



3. 포인트컷(PointCut) 신택스(Syntax): PointCut expressions

- 완전한 퀄리파이어(qualifier): Access Specifier Return Type package.class.methodName()
- 사용예: public int com.springaop.MyClass.multiply(int, int)



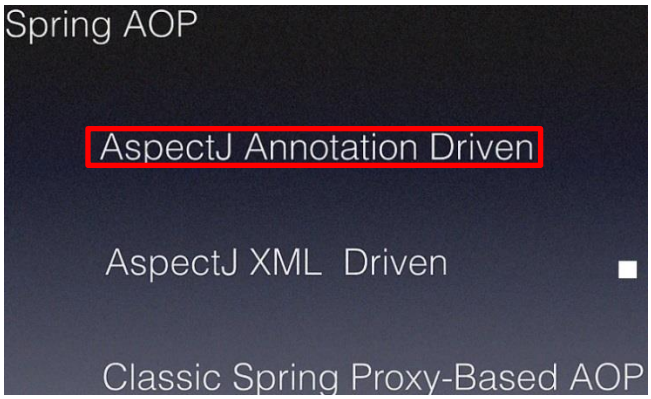
- AS(Access specifier), RT(Return Type), PACK(Package), CLASS, MN(Method Name)
- pack(패키지 레벨, 현재, 서브 패키지), 파라미터 레벨(Any Parameter)
- public void *Id(): 모든 패키지, 클래스내의 리턴없는 임의의 파라미터를 가진 이름이 Id로 끝나는 파라미터가 없는 메소드
- public int *e*(..): 모든 패키지/클래스내의 int 리턴하는 e가 포함된 임의의 수의 파라미터를 가진 메소드
- public int get(..): 접근이 public이면서, 리턴타입이 int이면서, 임의의 수의 파라미터를 가진 정확히 이름이 get인 메소드
- public * *(): 임의의 리턴타입, 파라미터가 없는 임의의 메소드
- public int *(..): 리턴타입이 int이면서, 임의의 파라미터 수를 가진 임의의 메소드
- public * com.app..*.get*(): com.app패키지아래의 임의의 클래스 중 임의의 리턴타입을 가지고, get으로 시작하는 파라미터 없는 임의의 메소드
- public * *(..): 임의의 리턴타입을 가지고, 임의의 수의 파라미터를 가지는 임의의 패키지 아래에 있는 모든 메소드

4. AOP 지원하는 대표적인 프레임워크

가. AspectJ

나. Spring AOP

다. JBoss AOP



5. AspectJ 애노테이션

- @Before("${pattern}")

말 그대로 지정한 패턴에 해당하는 메소드가 실행되기 전에, interceptor와 같이 동작

이 어노테이션이 붙은 메소드의 반환 값은 void여야 함

- @After("${pattern}")

지정한 패턴에 해당하는 메소드가 실행된 후에 동작

이 어노테이션이 붙은 메소드의 반환 값은 void여야 함

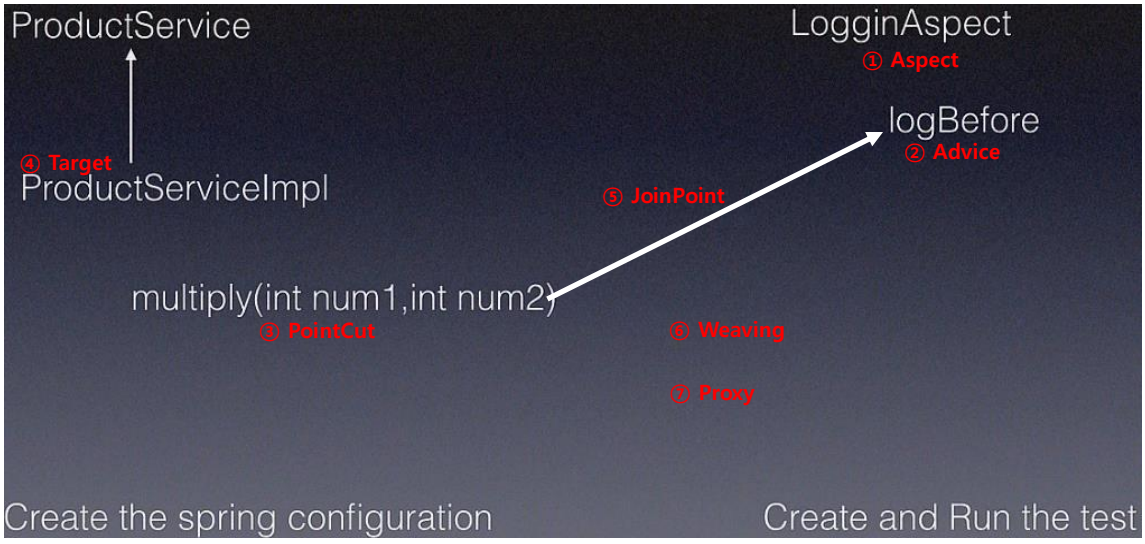
- @Around("${pattern}"}

지정된 패턴에 해당하는 메소드의 실행되기 전, 실행된 후 모두에서 동작

이 어노테이션이 붙은 메소드의 반환 값은 Object여야 함: 지정된 패턴에 해당하는 메소드의 실행 결과를 반환해야 하므로

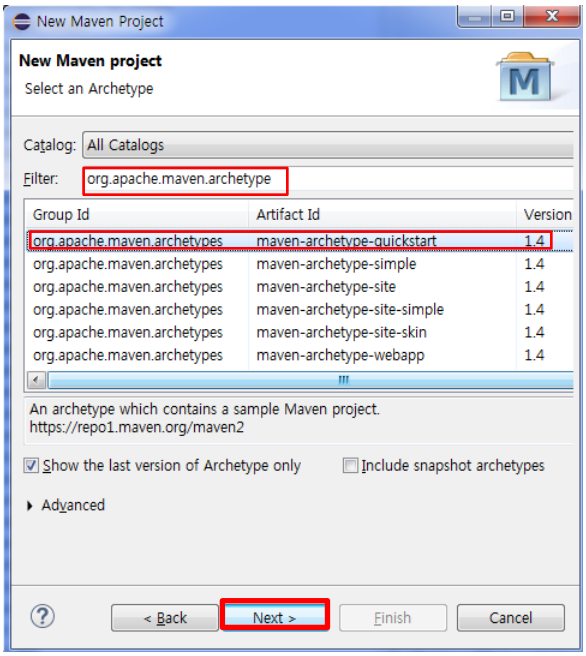
- @AfterReturning: AOP가 적용될 메소드가 에러없이 성공적으로 실행된 이후에 동작
- @AfterThrowing: AOP가 적용될 메소드가 에러발생하여 Exception을 던지는 시점에 동작

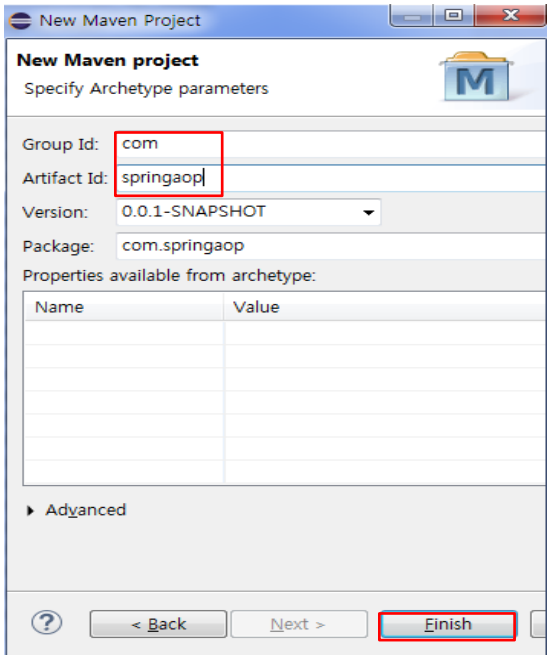
6. AOP 유스케이스

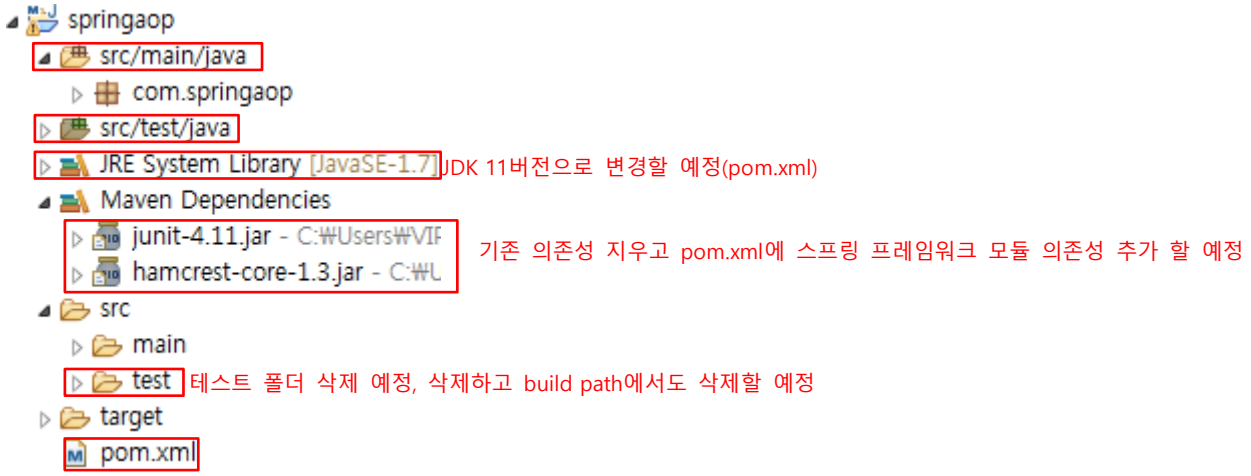


7. AOP 메이븐 프로젝트 만들기

가. 기본 프로젝트 만들기


New Maven Project
New Maven project
Select an Archetype
Catalog: All Catalogs
Filter: org.apache.maven.archetype
Group Id | Artifact Id | Version
org.apache.maven.archetypes | maven-archetype-quickstart | 1.4
org.apache.maven.archetypes | maven-archetype-simple | 1.4
org.apache.maven.archetypes | maven-archetype-site | 1.4
org.apache.maven.archetypes | maven-archetype-site-simple | 1.4
org.apache.maven.archetypes | maven-archetype-site-skin | 1.4
org.apache.maven.archetypes | maven-archetype-webapp | 1.4
An archetype which contains a sample Maven project.
https://repo1.maven.org/maven2
Show the last version of Archetype only
Include snapshot archetypes
Advanced
< Back
Next >
Finish
Cancel

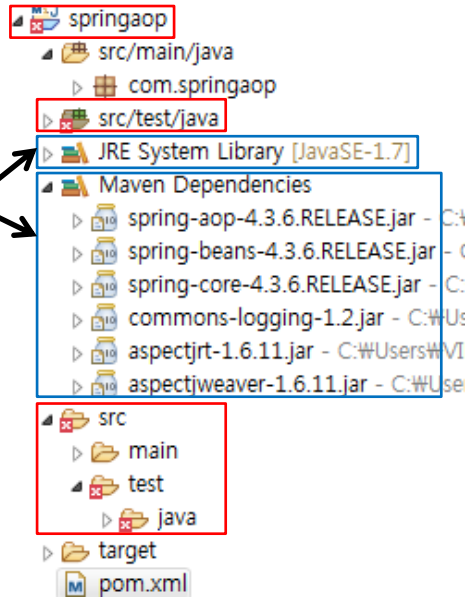

New Maven Project
New Maven project
Specify Archetype parameters
Group Id: com
Artifact Id: springaop
Version: 0.0.1-SNAPSHOT
Package: com.springaop
Properties available from archetype:
Name
Value
Advanced
< Back
Next >
Finish


springaop
src/main/java
com.springaop
src/test/java
JRE System Library [JavaSE-1.7]
JDK 11버전으로 변경할 예정(pom.xml)
Maven Dependencies
junit-4.11.jar - C:₩Users₩VIF
hamcrest-core-1.3.jar - C:₩L
기존 의존성 지우고 pom.xml에 스프링 프레임워크 모듈 의존성 추가 할 예정
src
main
test
테스트 폴더 삭제 예정, 삭제하고 build path에서도 삭제할 예정
target
pom.xml

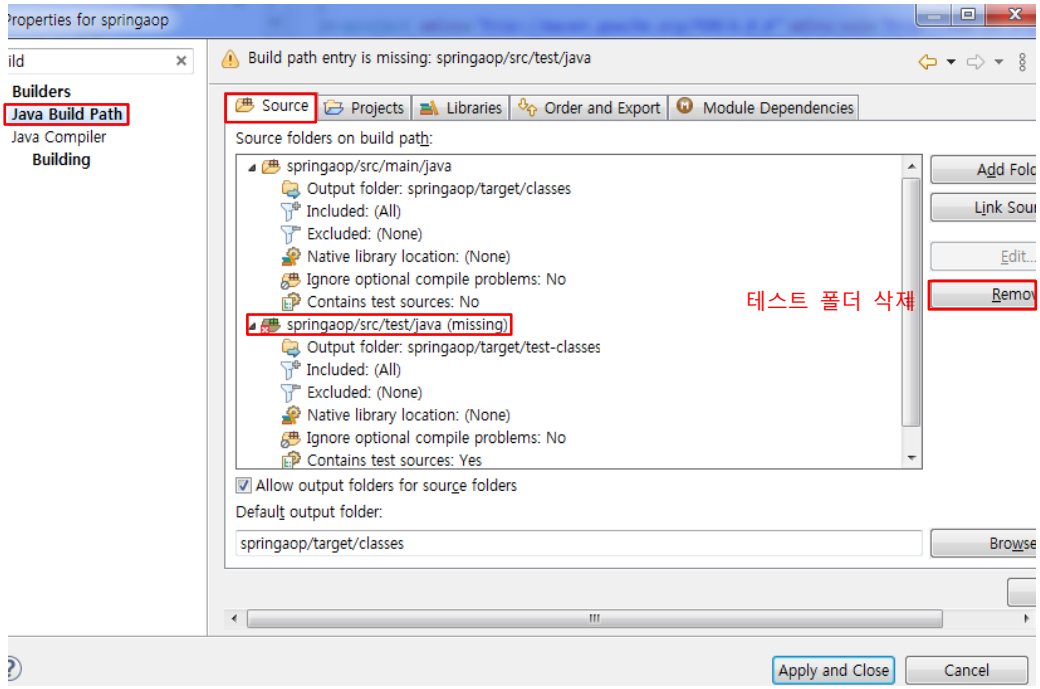
나. AOP, AspectJ 의존성 추가 및 JDK 버전 11로 변경: pom.xml

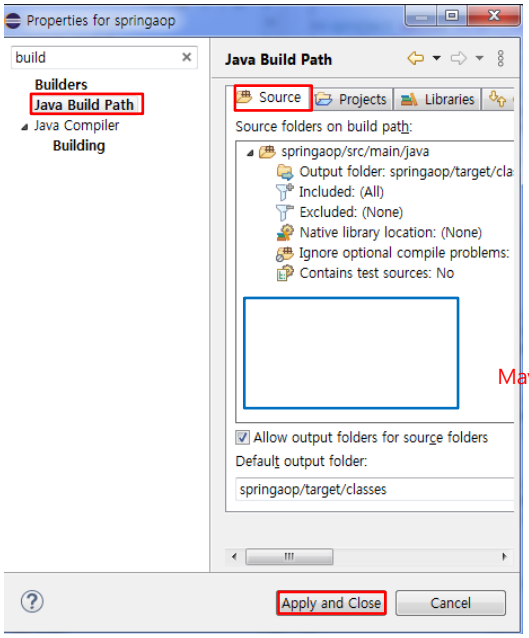
springaop/pom.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLS
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven
    <modelVersion>4.0.0</modelVersion>

    <groupId>com</groupId>
    <artifactId>springaop</artifactId>
    <version>0.0.1-SNAPSHOT</version>

    <name>springaop</name>
    <url>http://www.example.com</url>

    <properties>
        <springframework.version>4.3.6.RELEASE</springframework.version>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
            <version>${springframework.version}</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjrt</artifactId>
            <version>1.6.11</version>
        </dependency>
        <dependency>
            <groupId>org.aspectj</groupId>
            <artifactId>aspectjweaver</artifactId>
            <version>1.6.11</version>
        </dependency>
    </dependencies>
    <build>
        <pluginManagement>
            <plugins>
                <plugin>
                    <groupId>org.apache.maven.plugins</groupId>
                    <artifactId>maven-compiler-plugin</artifactId>
                    <version>3.8.0</version>
                    <configuration>
                        <source>11</source>
                        <target>11</target>
                    </configuration>
                </plugin>
            </plugins>
        </pluginManagement>
    </build>
</project>
```

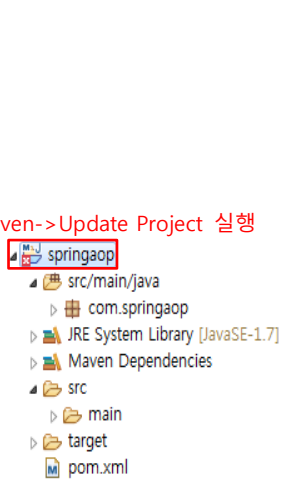


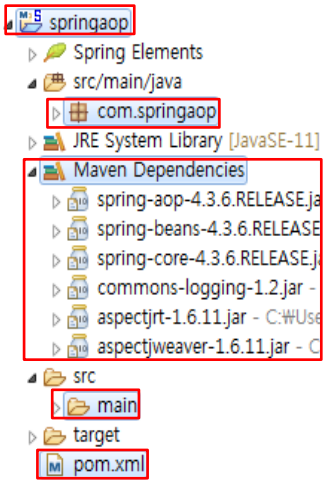
다. 작업 폴더 생성 정리: test 폴더 삭제 및 빌드패스 수정)



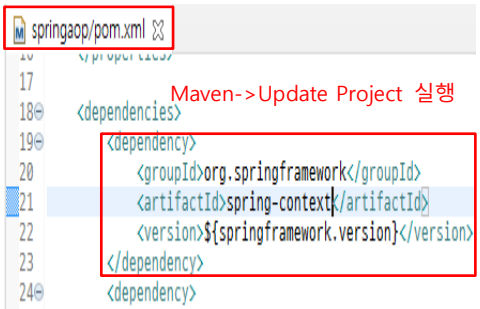


Maven->Update Project 실행

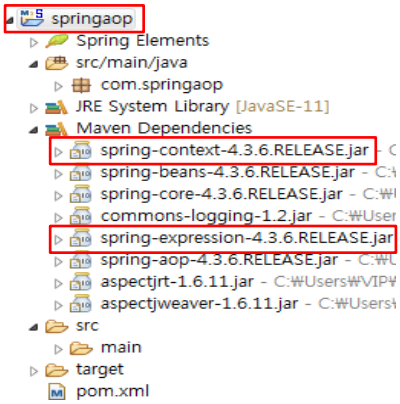




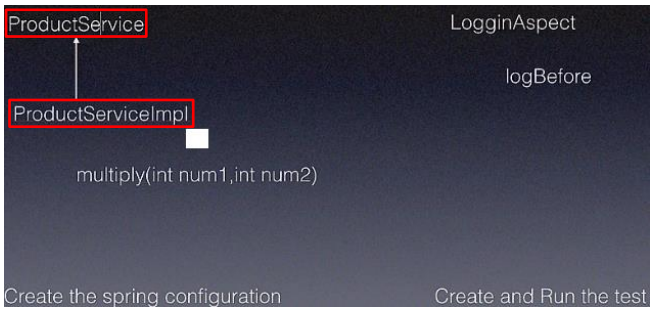
라. 스프링 애노테이션 사용: 검색할 패키지 위치를 지정 -> context 의존성추가



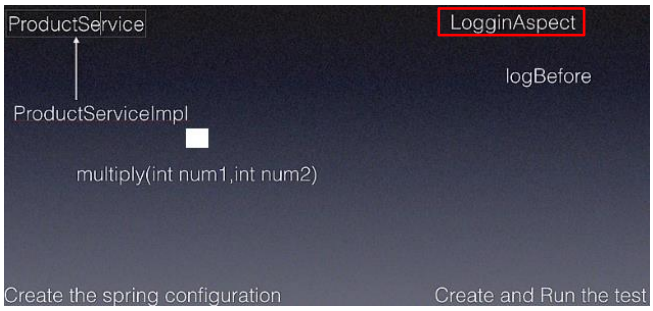
Maven->Update Project 실행



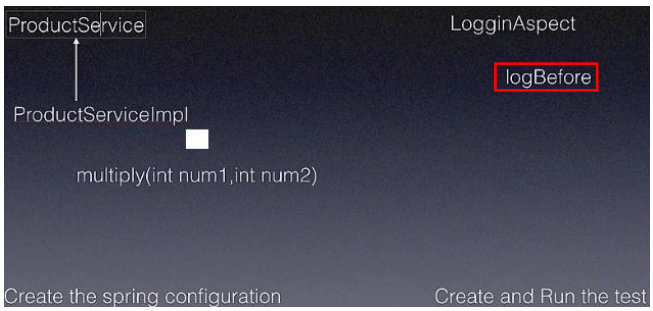
8. POJOs(타겟) 클래스 만들기



9. 로깅 Aspect 클래스 만들기

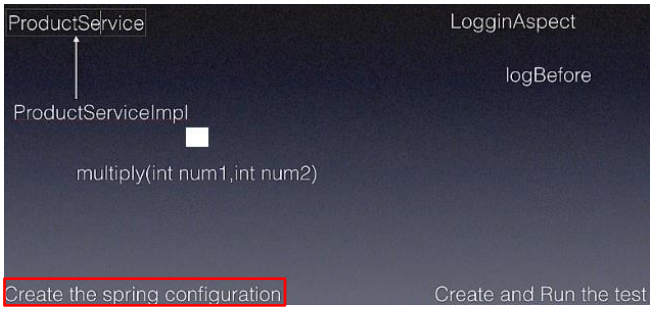


10. Advice 만들기(Aspect내 메소드)

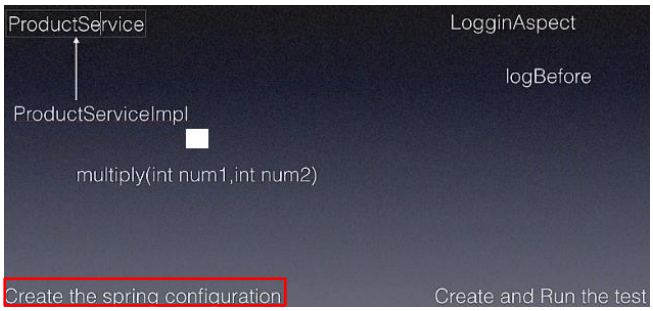


11. **PointCut 표현식** 만들기

12. 스프링 설정파일 만들기



13. Bean과 Aspect 설정

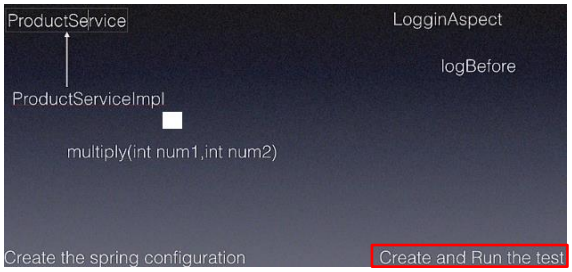


config.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<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:p="http://www.springframework.org/schema/p"
    xmlns:c="http://www.springframework.org/schema/c"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd">

    <aop:aspectj-autoproxy />

    <bean name="productService" class="com.springaop.ProductServiceImpl" />
    <bean name="loggingAspect" class="com.springaop.aspects.LoggingAspect" />

</beans>
```

Runtime 때 스프링 컨테이너가 자동으로 생성시켜주는 빈 지정

14. 테스트 클래스 만들기



15. 테스트 실행