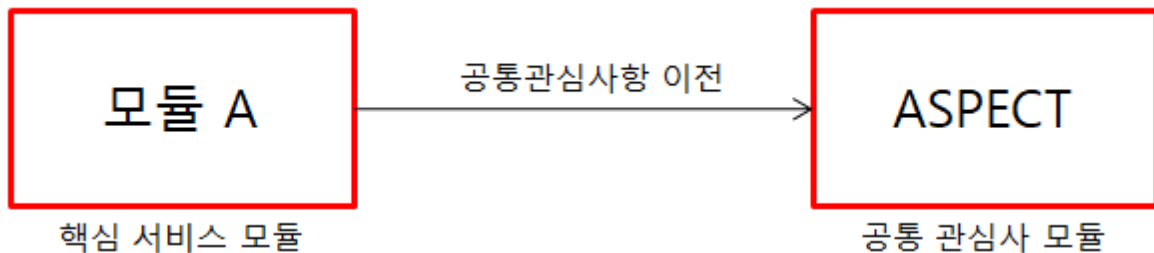


AOP (Aspect Oriented Programming)

OPP에서 Aspect만 바뀌었다.

특정 한 타이밍에 어느 위치에 어떤 메소드를 실행 시킬 것인지에 대한 프로그래밍 기법

Aspect : 기존 객체 지향 프로그램에서 공통 관심사를 모듈화 기법으로 추가된 중복코드(공통 처리 부분)를 별도의 독립된 클래스로 만들어 놓은 것



핵심 모듈 : 반드시 구현할 기능

ex) 로그인 여부 확인 (회원 수정, 삭제 등에 필요함)

위와 같은 기능을 할 Code들을 따로 Class로 만들어 준다. 이것을 **공통 관심사 모듈**이라고 한다.

공통 관심사 모듈

ex) 로그인 처리부분, 보안, 공통으로 사용할 수 없는 코딩

=> Spring에서는 Aspect를 Advice라고 한다. Angular에서는 Service라고 한다.

AOP는 언제 불러다 사용할 것이고, 언제 실행할 것인지가 중요하다.

이렇게 실행 위치를 지정해 주는 것을

JoinPoint, PointCut이라고 한다.

JoinPoint : 실행할 위치를 지정 (실행 위치만 지정한다.)

ex) before, after

PointCut : 어떤 메소드 앞, 뒤에서 실행을 지정.

어느 메소드를 사용할 것인지도 같이 설정해준다.

ex) before에 startMethod 실행지정.

이러한 지정 법은 Weaving이라고 한다.

Weaving : 핵심 Class의 특정 위치에 넣어주는 것

이러한 모든 것은 통틀어 Aspect라고 한다.

또한 Aspect 와 Advice 를 합쳐진 것을 보고 Advisor 라고도 한다.

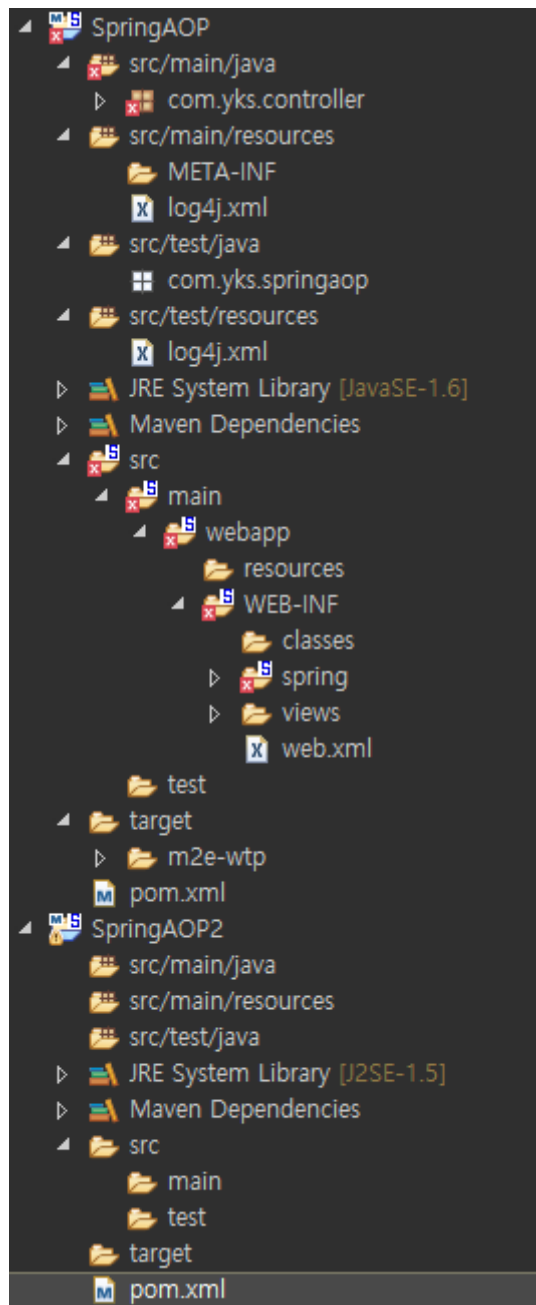
AOP 를 실행시키는 방법

1. 프록시 기반의 AOP를 지원(Aop 객체를 생성해주는 Class)
default (Method 중심) -> 빈즈 Class 로 구성
2. [aop:config](#) 태그를 이용
3. AspectJ 이용 (@Aspect Annotation 을 이용하는 방법)

프로젝트 생성

AOP -> springFrame(스프링 프레임워크가 필요하고, 스프링-익스프레션, 스프링-빈즈 라는 라이브러리가 필요하다.)


- 세 가지 라이브러리를 사용하기 위해 반드시 필요한 것 : Spring framework



프로젝트 생성시 Simple Spring Maven 을 선택하면 구조가 훨씬 간단하다.

Simple Spring Maven : 필요한 기능만 가져다 쓴다고 하면 이것 사용함

Spring MVC Project : MVC전체를 사용하고자 할때

 New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☐ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

```
/SpringAOP2/src/main/java/sp/aop/BeforeLogAdvice.java
```

```
package sp.aop;
```

```
import java.lang.reflect.Method;
```

```
import org.springframework.aop.MethodBeforeAdvice;
```

```
// 추가
```

```
import org.apache.commons.logging.Log;
```

```
import org.apache.commons.logging.LogFactory;
```

```
// Advice : 모든 Class 에 공통으로 사용할 기능 (소스코드 -> method 중심)
```

```
// interface 대신 (OOP방식)에 Advice Class 를 작성해서 구현(AOP 방식) -> Aspect  
Oriented Programming
```

```
// 실행 위치를 특정 Method 의 앞에서 실행 -> MethodBeforeAdvice를 구현
```

```
public class BeforeLogAdvice implements MethodBeforeAdvice {
```

```
    private Log log = LogFactory.getLog(getClass());    // Log 객체를 얻어오는 문장
```

```
    // 1. Spring의 AOP(Method 중심) -> 핵심 Class 의 Method
```

```
// 2. 생성된 객체를 배열로 받아온다.
// 3. target class(핵심 class 의 객체를 얻어온다.)

@Override
public void before(Method method, Object[] args, Object target) throws
Throwable {
    // TODO Auto-generated method stub
    log.info(method.toString() + "Method : " + target + "에서 호출 전!");
}
}
```

인터페이스 생성

New Java Interface

Create a new Java interface.

Source folder: SpringAOP2/src/main/java Browse...

Package: sp.aop Browse...

☐ Enclosing type: Browse...

Name: TestService

Modifiers: ☐ public ☐ package ☐ private ☐ protected

Extended interfaces: Add...
Remove

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

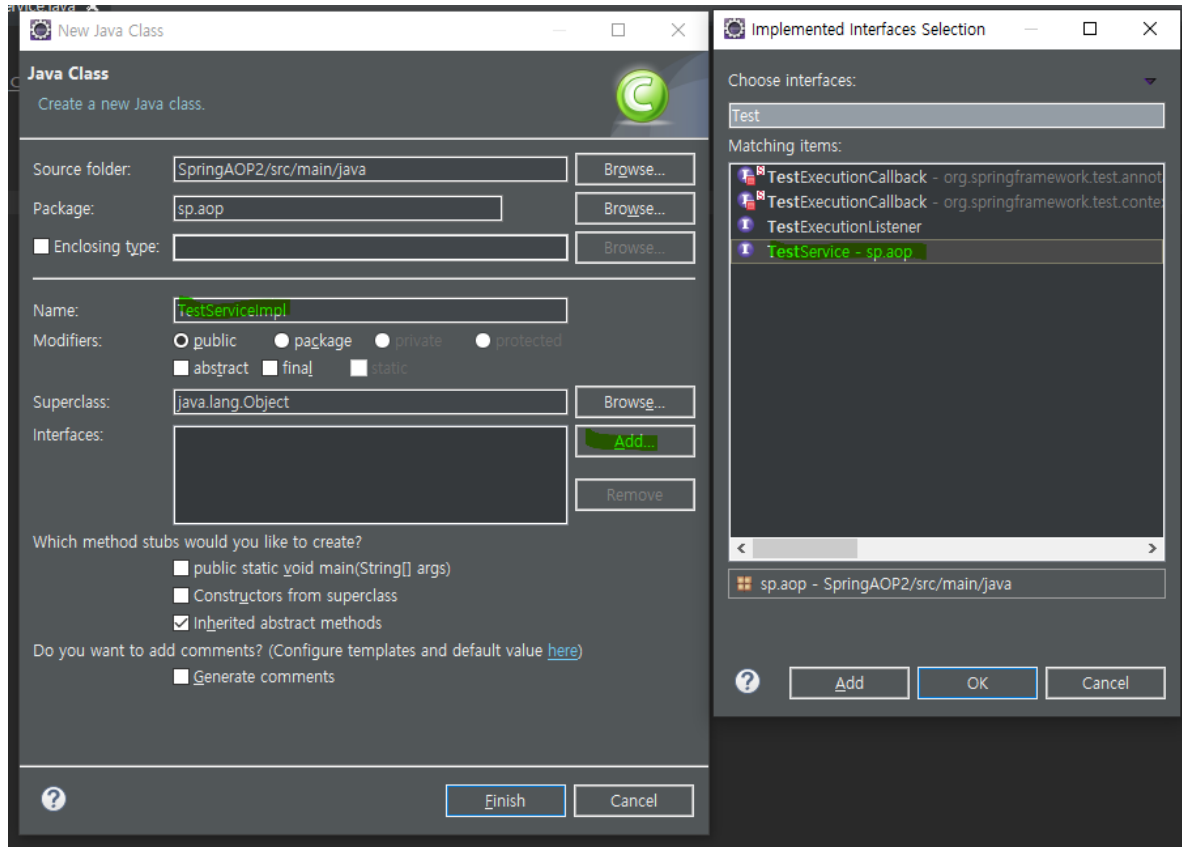
? Finish Cancel

```
/SpringAOP2/src/main/java/sp/aop/TestService.java

package sp.aop;

// 모든 핵심 class에서 공통으로 사용할 목적으로의 Method 작성
public interface TestService {
    public void save(String msg); // 입력
    public void write(); // 출력
}
```

구현받을 클래스 생성



/SpringAOP2/src/main/java/sp/aop/TestServiceImpl.java

```
package sp.aop;
```

```
// 2. 핵심 모듈 Class (=target Class)
```

```
// save(), write() -> 핵심 Method
```

```
public class TestServiceImpl implements TestService {
```

```
    private String msg = "before AOP 연습";
```

```
    @Override
```

```
    public void save(String msg) {          // 회원수정
```

```
        // TODO Auto-generated method stub
```

```
        this.msg = msg;
```

```
        System.out.println("save() Method 호출");
```

```
    }
```

```
    @Override
```

```
    public void write() {
```

```
        // TODO Auto-generated method stub
```

```
        System.out.println("save() Method 호출" + this.msg);
```

```
    }
```

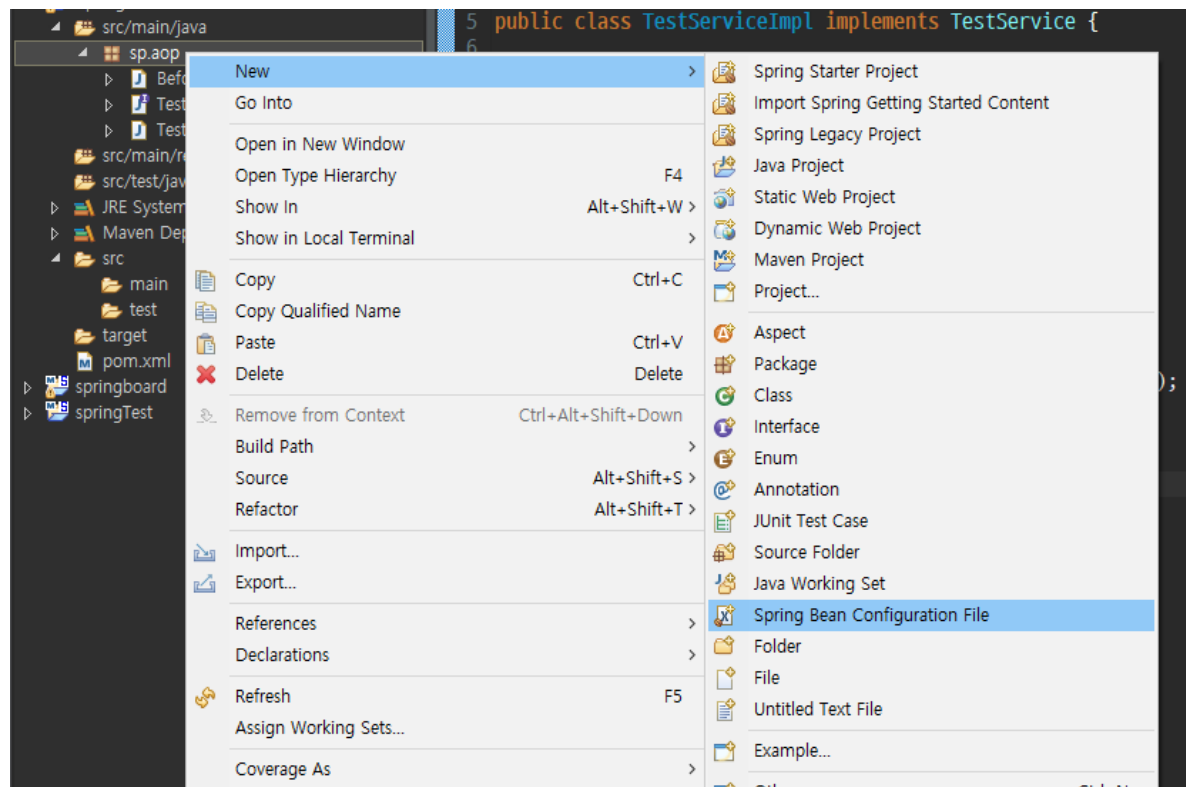
```
}
```

AOP의 Advice를 이해하는 부분이다.

원래는 위에 코딩을 했어야 했지만, Advice를 사용함으로써 코딩하지 않고 실행될 위치만 지정해주면 된다.

실행 위치는 xml에서 지정한다.


xml 생성



Create a new Spring Bean Definition file

New Spring Bean Definition file

Select the location and give a name for the Spring Bean Definition file



Enter or select the parent folder:

SpringAOP2/src/main/java/sp/aop

SpringAOP2

.settings

src

main

java

sp

aop

resources

test

target

springboard

springTest

File name:

Advanced >>

☒ Add Spring project nature if required

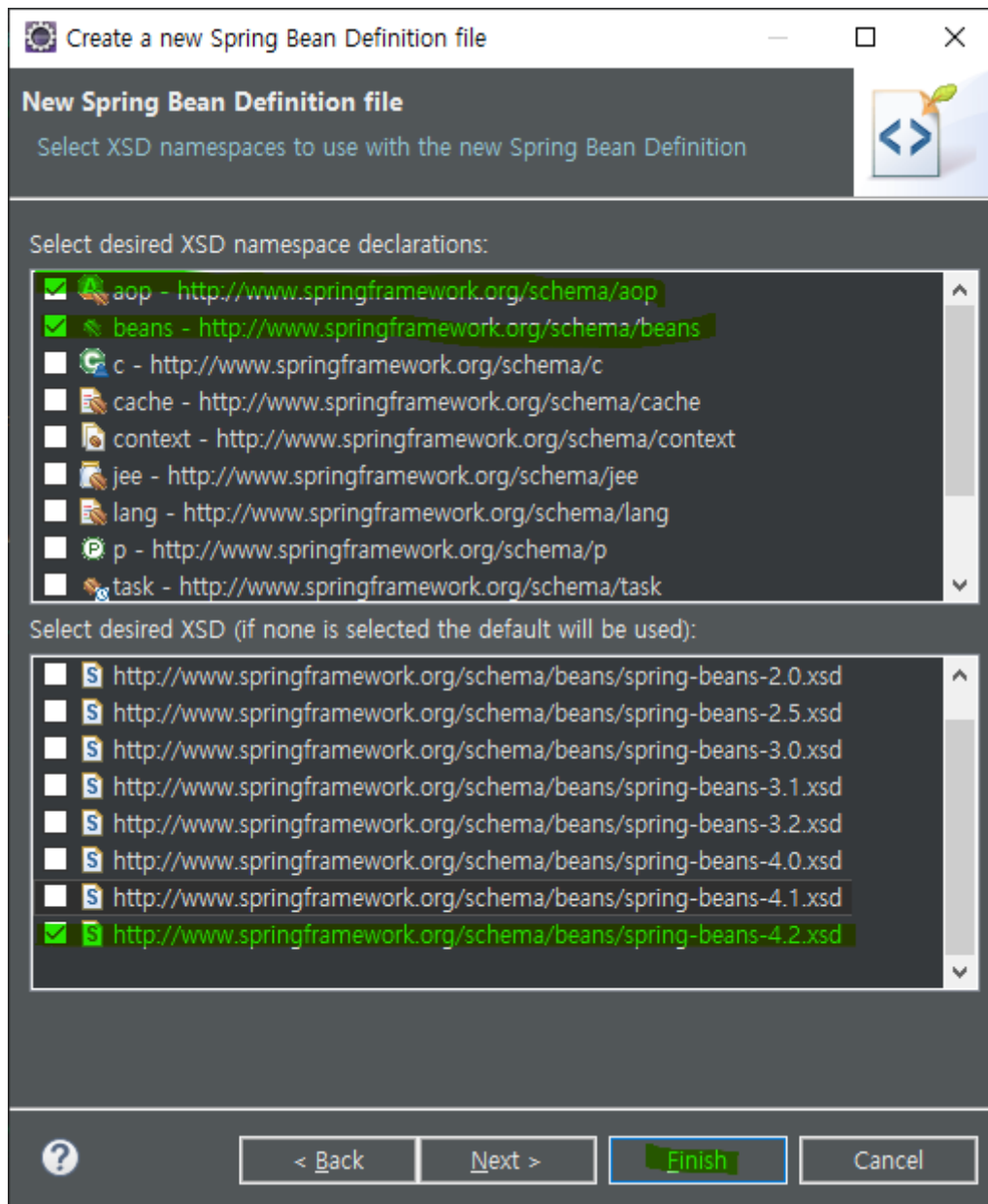
?

< Back

Next >

Finish

Cancel



아래처럼 나오면 성공

```
<?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:aop="http://www.springframework.org/schema/aop"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">

</beans>
```

```

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:aop="http://www.springframework.org/schema/aop"
5       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
6                           http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">
7
8
9 <!-- 1. 핵심 Class 빈즈 등록 -->
10 <bean id="testServiceImpl" class="sp.aop.TestServiceImpl" />
11
12 <!-- 2. Advice Class 빈즈 등록 -->
13 <bean id="beforeLog" class="sp.aop.BeforeLogAdvice" />
14
15
16 <!-- 3.PointCut 생성 -> 어느 위치에서 AOP Method 를 지정해서 실행
17     value="접근 지정 반환명 package 명... Class 명 하위 package 명(..) 특정 Method명 0개 이상"
18     (*) 매개변수 한 개 표시, (*,*) 매개변수 2개 표시 -->
19
20 <bean id="writePointcut" class="org.springframework.aop.support.JdkRegexpMethodPointcut">
21     <property name="pattern" value=".*write.*" />
22 </bean>
23
24
25 <!-- 4. Advice + PointCut(Advisor) 설정 -->
26 <bean id="testAdvisor" class="org.springframework.aop.support.DefaultPointcutAdvisor">
27     <property name="advice" ref="beforeLog" />
28     <property name="pointcut" ref="writePointcut" />
29 </bean>
30
31 <!-- 5. AOP를 적용(ProxyFactoryBean 객체들 생성) target(핵심 Class) -->
32 <bean id="testService" class="org.springframework.aop.framework.ProxyFactoryBean">
33     <property name="target" ref="testServiceImpl" />
34     <property name="interceptorNames">
35         <list>
36             <value>testAdvisor</value>
37         </list>
38     </property>
39 </bean>
40 <!-- ///////////END AOP 환경설정 부분 -->
41
42 </beans>

```

```

<?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:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
                           http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">

<!-- 1. 핵심 Class 빈즈 등록 -->
<bean id="testServiceImpl" class="sp.aop.TestServiceImpl" />

<!-- 2. Advice Class 빈즈 등록 -->
<bean id="beforeLog" class="sp.aop.BeforeLogAdvice" />

<!-- 3.PointCut 생성 -> 어느 위치에서 AOP Method 를 지정해서 실행
     value="접근 지정 반환명 package 명... Class 명 하위 package 명(..) 특정 Method
명 0개 이상"
     (*) 매개변수 한 개 표시, (*,*) 매개변수 2개 표시 -->

<bean id="writePointcut"[객체명]
       class="org.springframework.aop.support.JdkRegexpMethodPointcut">[네임속성은 수정
절대 불가]
     <property name="pattern" value=".*write.*" />[패턴에 들어갈 값은 Method의 위치
를 써준다.] --> 모든 패키지 내부에 있는 write메소드 라고 써준 것임
</bean>

<!-- 4. Advice + PointCut(Advisor) 설정 -->
<bean id="testAdvisor" [객체명]
       class="org.springframework.aop.support.DefaultPointcutAdvisor"> [폴패키지 네임]

```


```

    <property name="advice" ref="beforeLog" /> [네임속성 변경 불가, advice가 들어가는 것은 객체명이 들어가야 해서 ref속성을 넣음.]
    <property name="pointcut" ref="writePointcut" />
</bean>

<!-- 5. AOP를 적용(ProxyFactoryBean 객체를 생성) target(핵심 class) -->AOP 객체 생성
<bean id="testService"[객체명]
    class="org.springframework.aop.framework.ProxyFactoryBean"> [객체를 만들기 위한 패키지 네임]
    <property name="target" [원래 클래스에서 제공하는 것이기 때문에 수정 불가능, 멤버변수]
        ref="testServiceImpl"[핵심 class 빈즈 등록] />
    <property name="interceptorNames"> [advisor를 등록시켜줄 때 필요로 하는 멤버 변수]-기본이 리스트 형태로 받는다.
        <list> [그래서 값을 넣어주려면 list태그에 value 속성이 필요하다]
            <value>testAdvisor</value> [값을 더 넣고 싶다면, 이 부분에 밸류들 추가]
        </list>
    </property>
</bean>
<!-- ///////////END AOP 환경설정 부분 -->

</beans>

```

 New Java Class

Java Class
Create a new Java class.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☒ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static

Superclass:

Interfaces:

Which method stubs would you like to create?
☒ `public static void main(String[] args)`
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

ResultMain.java BeforeLogAdvice.java TestService.java TestServiceImpl.java SpringAop2/pom.xml

```

1 package sp.aop;
2
3 import org.springframework.context.ApplicationContext;
4 import org.springframework.context.support.ClassPathXmlApplicationContext; // xml문서를 찾을 수 있게 하는 lib
5
6 public class ResultMain {
7
8     public static void main(String[] args) {
9         String path="sp/aop/app.xml";
10        ApplicationContext context = new ClassPathXmlApplicationContext(path);
11
12        //TestService service= (TestService)context.getBean("testServiceImpl");
13        //원래는 위의 것이 일반적이지만, AOP 객체를 얻어오기 위해서는, 밑에처럼 AOP객체를 얻는다.
14
15        //AOP객체를 생성 -> Advisor 작동 -> Advice + pointcut 실행
16        TestService service = (TestService)context.getBean("testService");
17        service.save("AOP 적용 연습");
18        //before advice(before() 작동 실행) -> 실행상태에서 처리
19        service.write();
20
21    } //main() END
22
23 } //CLASS END

```

메인메서드 클래스 작성

/SpringAOP2/src/main/java/sp/aop/ResultMain.java

```
package sp.aop;
```

```

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext; //
xml문서를 찾을 수 있게 하는 lib

public class ResultMain {

    public static void main(String[] args) {
        String path="sp/aop/app.xml";
        ApplicationContext context = new ClassPathXmlApplicationContext(path);

        //TestService service= (TestService)context.getBean("testServiceImpl");
        //원래는 위의 것이 일반적이지만, AOP 객체를 얻어오기 위해서는, 밑에처럼 AOP객체를
        얻는다.

        //AOP객체를 생성 -> Advisor 작동 -> Advice + pointcut 실행
        TestService service = (TestService)context.getBean("testService");
        service.save("AOP 적용 연습");
        //before advice(before() 작동 실행) -> 실행상태에서 처리
        service.write();

    } //main() END

} //CLASS END


```

```


<terminated> ResultMain [Java Application] C:\Program Files\Java\jre1.8.0_251\bin\javaw.exe (2020. 5. 20. 오후 2:48:33)
5월 20, 2020 2:48:33 오후 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5010be6: startup date [Wed May 20 14:48:33 KST 2020]; root of context hierarchy
5월 20, 2020 2:48:33 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [sp/aop/app.xml]
save() Method 호출
save() Method 호출AOP 적용 연습
5월 20, 2020 2:48:34 오후 sp.aop.BeforeLogAdvice before
정보: public abstract void sp.aop.TestService.write()Method : sp.aop.TestServiceImpl@1f57539에서 호출 전!

```

save 메서드가 호출되고 before가 작동한 후에 write가 호출되는 것을 확인할 수 있다.

 New Java Class

Java Class

 Type already exists.

Source folder:

Package:

☐ Enclosing type:

Name:

Modifiers: ☐ public ☐ package ☐ private ☐ protected
☐ abstract ☐ final ☐ static


Superclass:

Interfaces:

Which method stubs would you like to create?

☐ public static void main(String[] args)
☐ Constructors from superclass
☒ Inherited abstract methods

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments



```
/SpringAOP2/src/main/java/sp/aop/AfterLogAdvice.java
```

```
package sp.aop;
```

```
import java.lang.reflect.Method;
```

```
import org.springframework.aop.AfterReturningAdvice;
```

```
public class AfterLogAdvice implements AfterReturningAdvice {
```

```
    /*
     * 1. 추가된 객체
     * 2. 핵심 클래스의 method명
     * 3. 생성된 객체들
     * 4. target Class의 객체
     */
```

```
@Override
```

```
    public void afterReturning(Object returnValue, Method method, Object[] args,
    Object target) throws Throwable {
        // TODO Auto-generated method stub
    }
}
```

```

        System.out.println(method.toString() + "Method" + target + "에서 호출
후!");
    }
}

```

app.xml 수정

```

/SpringAOP2/src/main/java/sp/aop/app.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:aop="http://www.springframework.org/schema/aop"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans-4.2.xsd
    http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop-4.2.xsd">

    <!-- 1. 핵심 class 빈즈 등록 -->
    <bean id="testServiceImpl" class="sp.aop.TestServiceImpl" />

    <!-- 2. Advice Class 빈즈 등록 -->
    <bean id="beforeLog" class="sp.aop.BeforeLogAdvice" />
    <bean id="AfterLog" class="sp.aop.AfterLogAdvice" />

    <!-- 3.PointCut 생성 -> 어느 위치에서 AOP Method 를 지정해서 실행
        value="접근 지정 반환명 package 명... Class 명 하위 package 명(..) 특정 Method명
0개 이상"
        (*) 매개변수 한 개 표시, (*,*) 매개변수 2개 표시 -->

    <bean id="writePointcut"
class="org.springframework.aop.support.JdkRegexpMethodPointcut">
        <property name="pattern" value=".*write.*" />
    </bean>

    <!-- 4. Advice + PointCut(Advisor) 설정 -->
    <bean id="testAdvisor"
class="org.springframework.aop.support.DefaultPointcutAdvisor">
        <property name="advice" ref="beforeLog" />
        <property name="pointcut" ref="writePointcut" />
    </bean>

    <bean id="testAfterAdvisor"
class="org.springframework.aop.support.DefaultPointcutAdvisor">
        <property name="advice" ref="AfterLog" />
        <property name="pointcut" ref="writePointcut" />
    </bean>

    <!-- 5. AOP를 적용(ProxyFactoryBean 객체를 생성) target(핵심 Class) -->
    <bean id="testService"
class="org.springframework.aop.framework.ProxyFactoryBean">
        <property name="target" ref="testServiceImpl" />

```

```

<property name="interceptorNames">
  <list>
    <value>testAdvisor</value>
    <value>testAfterAdvisor</value>
  </list>
</property>
</bean>

```

```

terminated: Reason: java.lang.OutOfMemoryError: Java heap space (2020.5.20. 4:15:57)
5월 20, 2020 4:15:57 오후 org.springframework.context.support.ClassPathXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.ClassPathXmlApplicationContext@5010be6: startup date [Wed May 20 16:15:57 KST 2020]; root of context hierarchy
5월 20, 2020 4:15:57 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [sp/aop/app.xml]
save() Method 호출
5월 20, 2020 4:15:58 오후 sp.aop.BeforeLogAdvice before
정보: public abstract void sp.aop.TestService.write()Method : sp.aop.TestServiceImpl@6ee52dcd에서 호출 전!
save() Method 호출AOP 적용 연습
public abstract void sp.aop.TestService.write()Methodsp.aop.TestServiceImpl@6ee52dcd에서 호출 후!

```

app.xml 에 추가

```

<!-- 3.PointCut 생성 --> 어느 위치에서 AOP Method 를 지정해서 실행
value="접근 지정 반환명 package 명... Class 명 하위 package 명(..) 특정 Method명
0개 이상"
(* ) 매개변수 한 개 표시, (*,*) 매개변수 2개 표시 -->

```

```

<bean id="writePointcut"
class="org.springframework.aop.support.JdkRegexpMethodPointcut">
  <property name="pattern" value=".*write.*" />
</bean>

```

```

<!-- 아래 내용 추가 -->
<bean id="savePointcut"
class="org.springframework.aop.support.JdkRegexpMethodPointcut">
  <property name="pattern" value=".*save.*" />
</bean>

```