AOP (Aspect Oriented Programming) 실습1

**I. AOP 개념**

**(1) 개념**

지금까지 공통 기능을 모든 모듈에 적용하기 위한 방법으로 상속을 통한 방법이 있었지만, 이러한 한계를 극복하기 위해서 AOP가 등장하게 되었다.

**상속의 문제점 : ① 다중 상속이 불가하므로 다양한 모듈에 상속기법을 통한 공통 기능 부여는 한계.**

**② 기능 구현부분에 핵심 기능 코드와 공통 기능 코드가 섞여 있어 효율성 저하.**

⇒ AOP방법 : 핵심기능과 공통기능(**Aspect**)을 분리 시켜놓고, 공통 기능을 필요로 하는 핵심기능들에서

사용하는 방식

- Aspect : 공통 기능

- Advice : Aspect의 기능 자체

- Jointpoint : Advice를 적용해야 되는 부분(필드, 메소드) (스프링에서는 메소드만 해당)

- Pointcut : Jointpoint의 부분으로 실제로 Advice가 적용된 부분

- Weaving : Advice를 핵심 기능에 적용 하는 행위

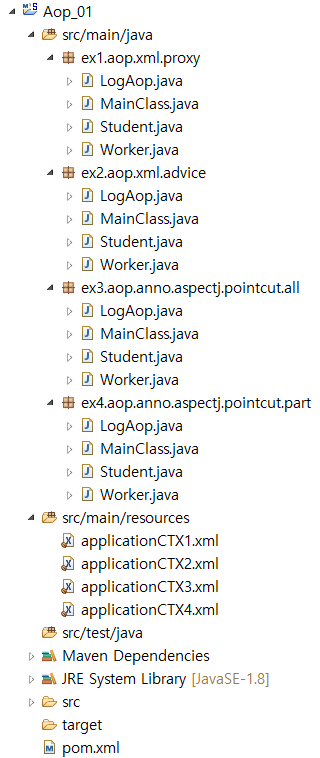
(2) 스프핑에서 AOP 구현 방법: proxy 사용



(3) 스프링에서 AOP 구현 방식

① XML 스키마 기반의 AOP구현

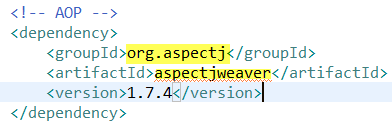
② @Aspect 어노테이션 기반의 AOP 구현



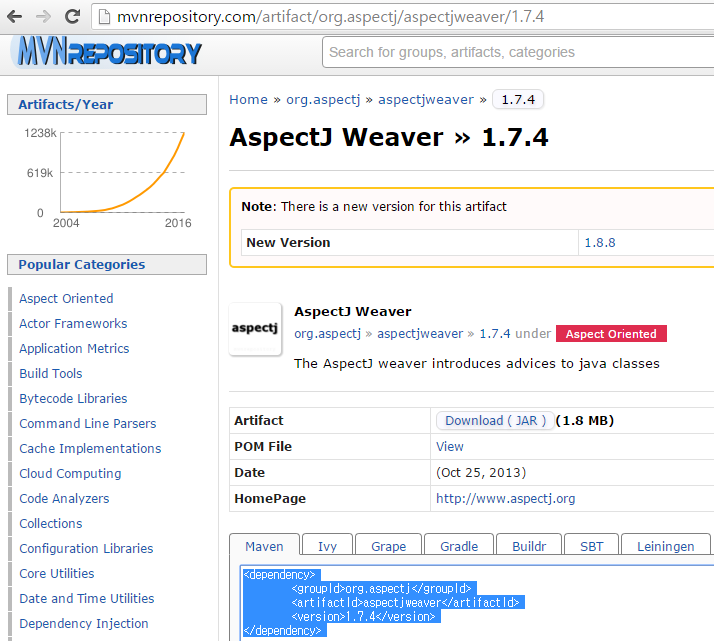
**II. XML기반의 AOP구현**

**(1) aspectj 의존설정**

[pom.xml]



※ 반드시 1.7.4 이상이어야 함 (구글검색: "aspectjweaver maven")



**(2) 공통 기능의 클래스 제작 – Advice 역할 클래스**

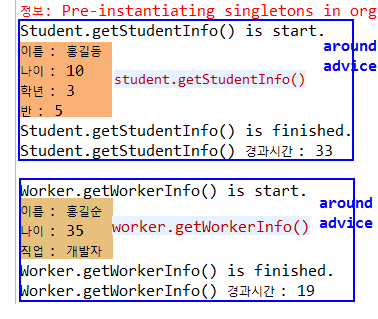
Advice 종류

|  |
| --- |
| <**aop:before> : 메소드 실행 전에 advice실행**  **<aop:after-returning> : 정상적으로 메소드 실행 후에 advice실행**  **<aop:after-throwing> : 메소드 실행중 exception 발생시 advice실행**  **<aop:after> : 메소드 실행중 exception 이 발생하여도 advice실행**  **<aop:around> : 메서드 실행 전/후 및 exception 발생시 advice실행** |

**(3) XML설정 파일에 Aspect 설정**

**[실습1]**

[결과]



[applicationCTX1.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.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd"*>      <aop:config>  <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex1.aop.xml.proxy.\*)"* />  <aop:around pointcut-ref=*"publicM"* method=*"loggerAop"* />  </aop:aspect>  </aop:config>        </beans> |

[MainClass.java ]

|  |
| --- |
| **package** ex1.aop.xml.proxy;  **import** org.springframework.context.support.AbstractApplicationContext;  **import** org.springframework.context.support.GenericXmlApplicationContext;  **public** **class** MainClass {  **public** **static** **void** main(String[] args) {      ctx.close();  }  } |

[LogAop.java ] :: 공통기능 클래스

|  |
| --- |
| **package** ex1.aop.xml.proxy;  **import** org.aspectj.lang.ProceedingJoinPoint;  **public** **class** LogAop {  **public** Object loggerAop(ProceedingJoinPoint joinpoint) **throws** Throwable {  String signatureStr = joinpoint.getSignature().toShortString();    }  } |

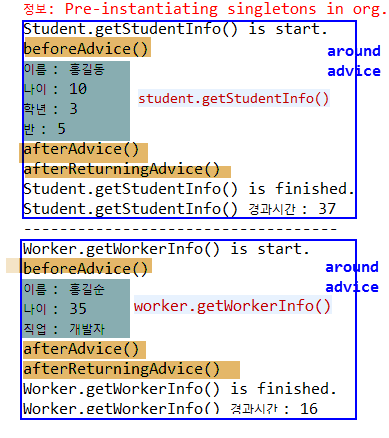
[Student.java ]

|  |
| --- |
| **package** ex1.aop.xml.proxy;  **public** **class** Student {  **private** String name;  **private** **int** age;  **private** **int** gradeNum;  **private** **int** classNum;      **public** **void** getStudentInfo() {    }  } |

[Worker.java ]

|  |
| --- |
| **package** ex1.aop.xml.proxy;  **public** **class** Worker {  **private** String name;  **private** **int** age;  **private** String job;      **public** **void** getWorkerInfo() {    }  } |

**[실습2]**



[ applicationCTX2.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.xsd*  *http://www.springframework.org/schema/aop*  *http://www.springframework.org/schema/aop/spring-aop-3.2.xsd"*>    <aop:config>    <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex2.aop.xml.advice.\*)"* />  <aop:around pointcut-ref=*"publicM"* method=*"loggerAop"* />  </aop:aspect>    <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex2.aop.xml.advice.\*)"* />  <aop:before pointcut-ref=*"publicM"* method=*"beforeAdvice"* />  </aop:aspect>    <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex2.aop.xml.advice.\*)"* />  <aop:after-returning pointcut-ref=*"publicM"* method=*"afterReturningAdvice"* />  </aop:aspect>    <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex2.aop.xml.advice.\*)"* />  <aop:after-throwing pointcut-ref=*"publicM"* method=*"afterThrowingAdvice"* />  </aop:aspect>    <aop:aspect id=*"logger"* ref=*"logAop"*>  <aop:pointcut id=*"publicM"* expression=*"within(ex2.aop.xml.advice.\*)"* />  <aop:after pointcut-ref=*"publicM"* method=*"afterAdvice"* />  </aop:aspect>    </aop:config>      </beans> |

[MainClass.java ]

|  |
| --- |
| **package** ex2.aop.xml.advice;  **import** org.springframework.context.support.AbstractApplicationContext;  **import** org.springframework.context.support.GenericXmlApplicationContext;  **public** **class** MainClass {  **public** **static** **void** main(String[] args) {        }  } |

[LogAop.java ]

|  |
| --- |
| **package** ex2.aop.xml.advice;  **import** org.aspectj.lang.JoinPoint;  **import** org.aspectj.lang.ProceedingJoinPoint;  **public** **class** LogAop {  **public** Object loggerAop(ProceedingJoinPoint joinpoint) **throws** Throwable {  String signatureStr = joinpoint.getSignature().toShortString();  System.***out***.println( signatureStr + " is start.");  **long** st = System.*currentTimeMillis*();    **try** {  } **finally** {  **long** et = System.*currentTimeMillis*();  System.***out***.println( signatureStr + " is finished.");  System.***out***.println( signatureStr + " 경과시간 : " + (et - st));  }    }    **public** **void** beforeAdvice(JoinPoint joinPoint) {  System.***out***.println("beforeAdvice()");  }    **public** **void** afterReturningAdvice() {  System.***out***.println("afterReturningAdvice()");  }    **public** **void** afterThrowingAdvice() {  System.***out***.println("afterThrowingAdvice()");  }    **public** **void** afterAdvice() {  System.***out***.println("afterAdvice()");  }    } |