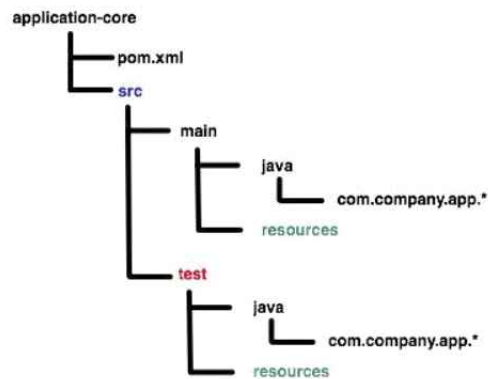


제 5 강 Maven 프로젝트 : AOP 심화

우리는 앞에서 DI, AOP를 공부하면서 Spring MVC 프로젝트를 사용하였다. 이것은 앞으로 우리가 웹 프로젝트를 주로 사용할 것이라는 것은 전제로 한 것이기 때문이다. 또 메이븐의 설정이 복잡하기 때문에 그러한 점을 생략하고 Spring 핵심 기능인 DI와 AOP 문법을 쉽게 접근하고자 하는 의도가 있었다. 이제 로컬 응용 어플리케이션을 개발할 때 사용하는 방식에 대해서 제대로 알아보자.

1. 메이븐 프로젝트의 구조

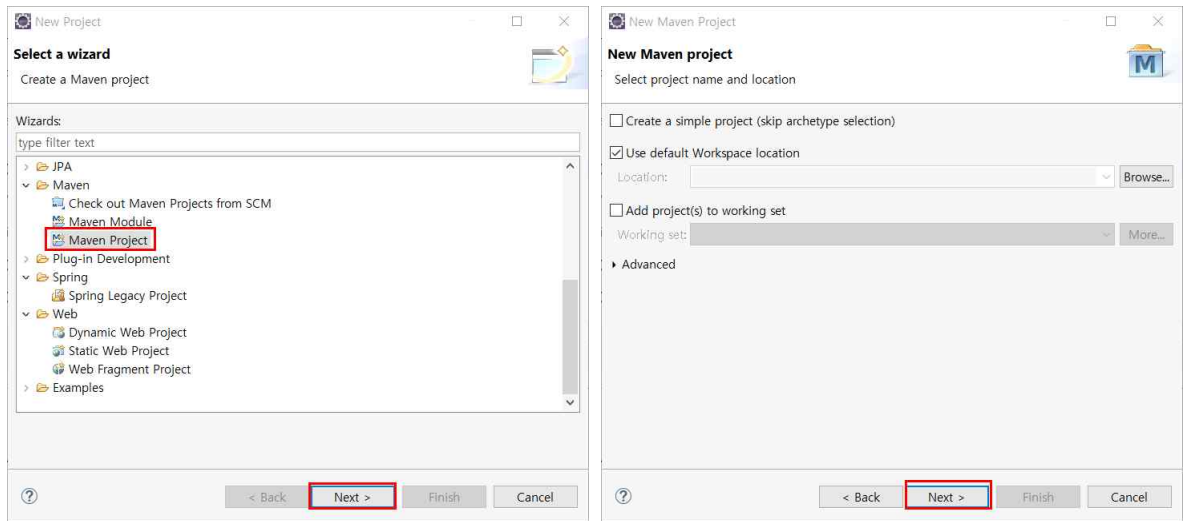


- ☐ 메이븐의 프로젝트 관리 디렉토리는 최상위에 프로젝트를 기준으로 pom.xml 이라는 메이븐 프로젝트 설정 파일이 존재하며 그것을 바탕으로 프로젝트와 관련된 정보를(라이브러리 및 빌드 정보 등) 기술하게 되어 있습니다.
- ☐ src 밑으로는 main 과 test 라는 디렉토리가 존재하며 각 하위에 java 와 resources 가 위치하게 됩니다.
- ☐ 기본적인 메이븐의 주요 디렉토리는 아래와 같습니다.
 - src/main/java : 자바 소스코드 파일을 위치시킵니다. 이 하위에 org.spring ... 와 같은 패키지를 배치합니다.
 - src/main/resources : 프로퍼티나 XML 등 리소스 파일이 위치합니다.
 - src/main/webapp : Web Project 일 경우 WEB-INF등 웹 어플리케이션 리소스를 위치시킵니다.
 - src/test/java : JUnit등의 테스트 파일이 위치하게 됩니다.
 - src/test/resources : 테스트 시 필요한 resource 파일이 위치하게 됩니다.

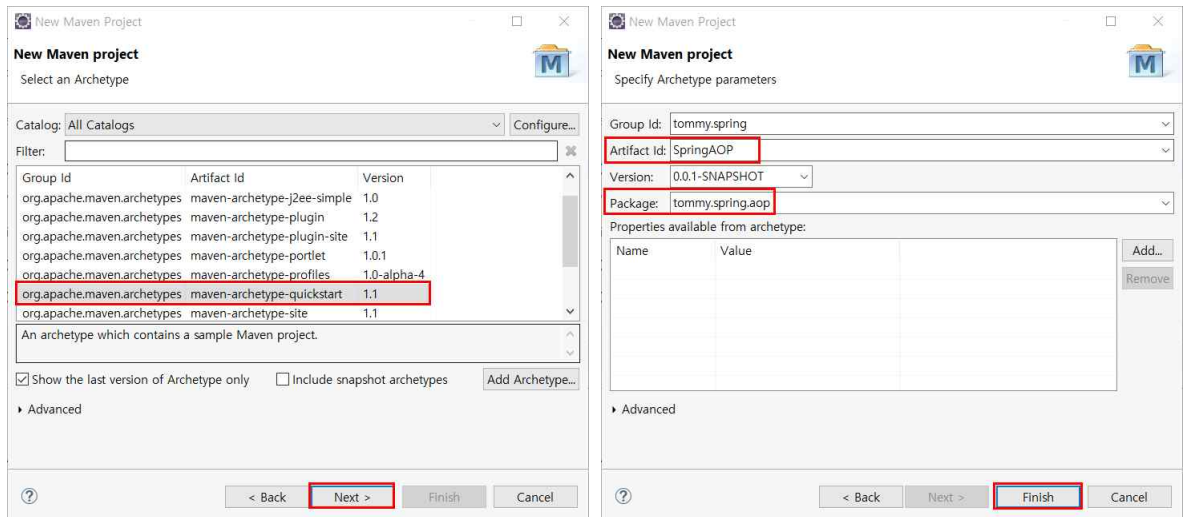
2. 메이븐 프로젝트 생성하기

① 메이븐 프로젝트 생성하기

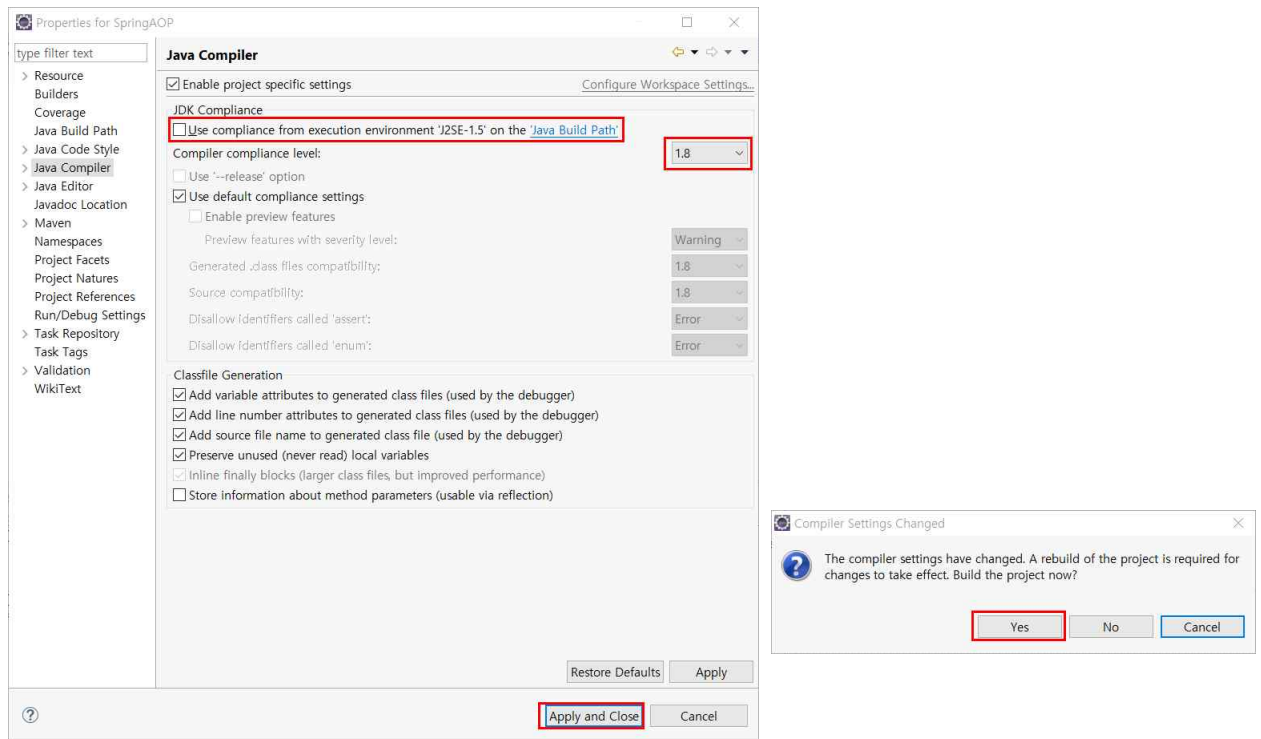
- [File]-[New]-[Project] 선택 후 아래와 같이 [Maven]-[Maven Project]를 선택하여 진행



- Artifact id에 SpringAOP라고 입력 후 아래 package를 tommy.spring.aop로 수정한다. 이때 Artifact id는 프로젝트 명이 됩니다. 그리고 Package 부분은 Artifact id를 입력 시 자동으로 입력되는데 본인이 원하는 패키지 명으로 수정하면 됩니다.



- Maven 프로젝트 설정 시 기본 Java의 사용은 1.5로 되어있습니다. java 1.5를 1.8 버전으로 변경해주는 작업이 필요합니다. Spring AOP 프로젝트 선택 후 우 클릭하여 [Properties] 메뉴 선택 후 아래와 같이 진행



② 메이븐 설정 및 스프링 의존관계 추가

: mvnrepository.com에서 spring-context 검색

□ pom.xml 파일을 아래와 같이 수정한다.

```

1      <!-- 상단 부분 생략 -->
2      <dependencies>
3          <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
4          <dependency>
5              <groupId>org.springframework</groupId>
6              <artifactId>spring-context</artifactId>
7              <version>5.1.9.RELEASE</version>
8          </dependency>
9          <dependency>
10             <groupId>junit</groupId>
11             <artifactId>junit</artifactId>
12             <version>4.12</version>
13             <scope>test</scope>
14         </dependency>
15     </dependencies>
16     <build>
17         <plugins>
18             <plugin>
19                 <groupId>org.apache.maven.plugins</groupId>
20                 <artifactId>maven-compiler-plugin</artifactId>
21                 <version>3.6.2</version>
22                 <configuration>
23                     <source>1.8</source>
24                     <target>1.8</target>
25                     <encoding>UTF-8</encoding>

```

25	<code></configuration></code>
26	<code></plugin></code>
27	<code></plugins></code>
28	<code></build></code>
29	<code></project></code>

- ☐ 이로써 가장 간단한 로컬 어플리케이션을 개발할 기본적인 준비가 끝났다. Spring의 DI 기능만을 테스트 한다면 지금의 설정으로도 충분히 가능하나 우리는 AOP 기능을 테스트 할 것이다.

③ AOP 관련 의존관계 추가

	<code><!-- 상단 부분 생략 --></code>
1	<code><!-- https://mvnrepository.com/artifact/aspectj/aspectjrt --></code>
2	<code><dependency></code>
3	<code><groupId>aspectj</groupId></code>
4	<code><artifactId>aspectjrt</artifactId></code>
5	<code><version>1.9.4</version></code>
6	<code></dependency></code>
7	<code><!-- https://mvnrepository.com/artifact/org.aspectj/aspectjweaver --></code>
8	<code><dependency></code>
9	<code><groupId>org.aspectj</groupId></code>
10	<code><artifactId>aspectjweaver</artifactId></code>
11	<code><version>1.9.4</version></code>
12	<code></dependency></code>
	<code><!-- 하단 부분 생략 --></code>

④ 로그 관련 라이브러리 추가

	<code><!-- 상단 부분 생략 --></code>
1	<code><!-- https://mvnrepository.com/artifact/commons-logging/commons-logging --></code>
2	<code><dependency></code>
3	<code><groupId>commons-logging</groupId></code>
4	<code><artifactId>commons-logging</artifactId></code>
5	<code><version>1.2</version></code>
6	<code></dependency></code>
	<code><!-- 하단 부분 생략 --></code>

- ☐ `src/main/java` 안의 모든 내용(패키지, 클래스)을 삭제한다. `src/test/java` 안의 모든 내용(패키지, 클래스)을 삭제한다. 이제 모든 기본 설정을 마무리하였다.

3. 실습을 위한 기본 구조 작성

- ① 기본적인 모델 VO 클래스와 인터페이스를 작성한다.
- ☐ 게시글을 나타내는 ArticleVO 클래스 작성

1	<code>package tommy.spring.board.vo;</code>
2	<code>public class ArticleVO {</code>
3	<code>private int id;</code>

4	public ArticleVO() {
5	}
6	public int getId() {
7	return id;
8	}
9	public void setId(int id) {
10	this.id = id;
11	}
12	}

☐ 데이터베이스 처리 ArticleDAO 인터페이스 작성

1	package tommy.spring.board.dao;
2	import tommy.spring.board.vo.ArticleVO;
3	public interface ArticleDAO {
4	void insert(ArticleVO article);
5	void updateReadCount(int id);
6	}

☐ 회원을 나타내는 MemberVO 클래스 작성

1	package tommy.spring.member.vo;
2	public class MemberVO {
3	}

② 데이터베이스 처리 구현 클래스 작성

☐ OracleDAO 인터페이스를 구현한 OracleArticleDAO 클래스 구현

1	package tommy.spring.board.dao;
2	import tommy.spring.board.vo.ArticleVO;
3	public class OracleArticleDAO implements ArticleDAO {
4	@Override
5	public void insert(ArticleVO article) {
6	System.out.println("MyOracleArticleDAO.insert() 실행");
7	}
8	@Override
9	public void updateReadCount(int id) {
10	System.out.println("MyOracleArticleDAO.updateReadCount() 실행");
11	}
12	}

③ 게시판 서비스를 담당할 서비스 클래스와 인터페이스를 작성한다.

☐ 게시판에 글쓰기 서비스를 제공할 WriteArticleService 인터페이스 작성

1	package tommy.spring.board.service;
2	import tommy.spring.board.vo.ArticleVO;
3	public interface WriteArticleService {
4	void write(ArticleVO article);

```
5 }
```

- ☐ WriteArticleService 인터페이스를 실제로 구현한 WriteArticleServiceImpl 클래스 작성

```
1 package tommy.spring.board.service;
2 import tommy.spring.board.dao.ArticleDAO;
3 import tommy.spring.board.vo.ArticleVO;
4 public class WriteArticleServiceImpl implements WriteArticleService {
5     private ArticleDAO articleDao;
6     public WriteArticleServiceImpl() {
7     }
8     public WriteArticleServiceImpl(ArticleDAO articleDao) {
9         this.articleDao = articleDao;
10    }
11    @Override
12    public void write(ArticleVO article) {
13        System.out.println("WriteArticleServiceImpl.write() 메서드 실행");
14        articleDao.insert(article);
15    }
16 }
```

④ 멤버서비스를 담당할 서비스 클래스와 인터페이스 작성

- ☐ 수정정보를 나타내줄 UpdateInfo 클래스 작성

```
1 package tommy.spring.member.vo;
2 public class UpdateInfo {
3 }
```

- ☐ 회원관리 서비스를 제공할 MemberService 인터페이스 작성

```
1 package tommy.spring.member.service;
2 import tommy.spring.member.vo.MemberVO;
3 import tommy.spring.member.vo.UpdateInfo;
4 public interface MemberService {
5     void regist(MemberVO member);
6     boolean update(String memberId, UpdateInfo info);
7 }
```

- ☐ MemberService 인터페이스를 구현한 MemberServiceImpl 클래스 작성

```
1 package tommy.spring.member.service;
2 import tommy.spring.member.vo.MemberVO;
3 import tommy.spring.member.vo.UpdateInfo;
4 public class MemberServiceImpl implements MemberService {
5     @Override
6     public void regist(MemberVO member) {
7         System.out.println("MemberServiceImpl.regist() 메서드 실행");
8     }
9 }
```

```

9      @Override
10     public boolean update(String memberId, UpdateInfo info) {
11         System.out.println("MemberServiceImpl.update() 메서드 실행");
12         return true;
13     }
14 }

```

☐ 이제 기본적인 구조 설계가 끝났다.

4. XML 설정을 활용한 AOP 실습

① XML 스키마를 이용해서 AOP를 구현하는 과정은 다음과 같다.

- ▣ 공통기능을 제공하는 **Advice 클래스를 구현**한다.
- ▣ XML 설정 파일에서 **<aop:config>** 태그를 이용해서 **Aspect를 설정**한다. **Advice를 어떤 Pointcut에 적용할지**를 결정한다.

② 공통기능을 제공할 Advice 클래스 작성

```

1  package tommy.spring.common;
2  import org.aspectj.lang.ProceedingJoinPoint;
3  public class ProfilingAdvice {
4      public Object trace(ProceedingJoinPoint joinPoint) throws Throwable {
5          String signatureString = joinPoint.getSignature().toShortString();
6          System.out.println(signatureString + " 시작");
7          long start = System.currentTimeMillis();
8          try {
9              Object result = joinPoint.proceed();
10             return result;
11         } finally {
12             long finish = System.currentTimeMillis();
13             System.out.println(signatureString + " 종료");
14             System.out.println(signatureString + " 실행 시간 : " + (finish - start) + "ms");
15         }
16     }
17 }

```

② 스프링 설정파일 작성

- ☐ Advice 클래스를 적용하려면 일단 XML 설정 파일에 **Advice클래스를 빈으로 등록**해야 한다.
- ☐ XML 스키마를 이용해서 AOP를 구현하려면 **aop 네임스페이스를 추가**해 주어야한다.
- ☐ **<aop:config>**, **<aop:aspect>**, **<aop:pointcut>**, **<aop:around>** 태그를 이용해서 AOP를 설정한다.
- ☐ Pointcut 표현식에 execution 명시자는 Advice를 적용할 패키지, 클래스 그리고 메서드를 표현할 때 사용한다.

☐ **src/main/resources/applicationContextOne.xml** : bean, aop 네임스페이스 체크

	<!-- 상단 부분 생략 -->
1	<bean id="performanceTraceAdvice" class="tommy.spring.common.ProfilingAdvice"></bean>
2	<aop:config>
3	<aop:aspect id="traceAspect1" ref="performanceTraceAdvice">
4	<aop:pointcut id="publicMethod"
5	expression="execution(public * tommy.spring.board.service..*(..))" />
6	<aop:around pointcut-ref="publicMethod" method="trace" />
7	</aop:aspect>
8	<aop:aspect id="traceAspect2" ref="performanceTraceAdvice">
9	<aop:around
10	pointcut="execution(public * tommy.spring.member.service..*(..))"
11	method="trace" />
12	</aop:aspect>
13	</aop:config>
14	<bean id="writeArticleService" class="tommy.spring.board.service.WriteArticleServiceImpl">
15	<constructor-arg><ref bean="articleDAO" /></constructor-arg>
16	</bean>
17	<bean id="articleDAO" class="tommy.spring.board.dao.OracleArticleDAO"></bean>
18	<bean id="memberService" class="tommy.spring.member.service.MemberServiceImpl"></bean>
19	</beans>

③ 테스트를 위한 메인 클래스 작성

1	package tommy.spring.board.controller;
2	import org.springframework.context.support.AbstractApplicationContext;
3	import org.springframework.context.support.ClassPathXmlApplicationContext;
4	import tommy.spring.board.service.WriteArticleService;
5	import tommy.spring.board.vo.ArticleVO;
6	import tommy.spring.member.service.MemberService;
7	import tommy.spring.member.vo.MemberVO;
8	public class MainOne {
9	public static void main(String[] args) {
10	String[] configLocations = new String[] { "applicationContextOne.xml" };
11	AbstractApplicationContext context =
	new ClassPathXmlApplicationContext(configLocations);
12	WriteArticleService articleService =
	(WriteArticleService) context.getBean("writeArticleService");
13	articleService.write(new ArticleVO());
14	MemberService memberService =
	context.getBean("memberService", MemberService.class);
15	memberService.regist(new MemberVO());
16	context.close();
17	}
18	}

④ MainOne 클래스 실행 및 결과확인



4. 다양한 AOP 실습

① 실습을 위하여 게시판 서비스에 Read 기능 추가하자.

☐ 사용자 정의 예외 클래스 작성 : 글이 없을 때 발생할 예외

```

1 package tommy.spring.board.vo;
2 public class ArticleNotFoundException extends Exception {
3     private static final long serialVersionUID = 1L;
4 }

```

☐ Read 서비스를 제공할 인터페이스 작성

```

1 package tommy.spring.board.service;
2 import tommy.spring.board.vo.ArticleNotFoundException;
3 import tommy.spring.board.vo.ArticleVO;
4 public interface ReadArticleService {
5     ArticleVO getArticleAndIncreaseReadCount(int id) throws ArticleNotFoundException;
6 }

```

☐ Read 서비스를 구현한 클래스 작성

```

1 package tommy.spring.board.service;
2 import tommy.spring.board.vo.ArticleNotFoundException;
3 import tommy.spring.board.vo.ArticleVO;
4 public class ReadArticleServiceImpl implements ReadArticleService {
5     @Override
6     public ArticleVO getArticleAndIncreaseReadCount(int id) throws ArticleNotFoundException {
7         if (id == 0)
8             throw new ArticleNotFoundException();
9         return new ArticleVO();
10    }
11 }

```

② Advice 클래스를 작성하자.

☐ Before, After Returning, After Throwing, After Advice 실습용 Logging 서비스를 제공할 Advice 클래스

```

1 package tommy.spring.common;
2 public class LoggingAdvice {

```

```

3      public void before() {
4          System.out.println("[LA] 메서드 실행 전 전처리 수행");
5      }
6      public void afterReturning(Object ret) {
7          System.out.println("[LA] 메서드 실행 후 후처리 수행, 리턴값=" + ret);
8      }
9      public void afterThrowing(Throwable ex) {
10         System.out.println("[LA] 메서드 실행 중 예외 발생, 예외=" +
                                ex.getClass().getName());
11     }
12     public void afterFinally() {
13         System.out.println("[LA] 메서드 실행 완료");
14     }
15 }

```

□ Around Advice 실습용 캐시 서비스를 제공할 Advice 클래스

```

1  package tommy.spring.common;
2  import java.util.HashMap;
3  import java.util.Map;
4  import org.aspectj.lang.ProceedingJoinPoint;
5  import tommy.spring.board.vo.ArticleVO;
6  public class ArticleCacheAdvice {
7      private Map<Integer, ArticleVO> cache = new HashMap<Integer, ArticleVO>();
8      public ArticleVO cache(ProceedingJoinPoint joinPoint) throws Throwable {
9          Integer id = (Integer) joinPoint.getArgs()[0];
10         ArticleVO article = cache.get(id);
11         if (article != null) {
12             System.out.println("[ACA] 캐시에서 Article[" + id + "] 구함");
13             return article;
14         }
15         ArticleVO ret = (ArticleVO) joinPoint.proceed();
16         if (ret != null) {
17             cache.put(id, ret);
18             System.out.println("[ACA] 캐시에 Article[" + id + "] 추가함");
19         }
20         return ret;
21     }
22 }

```

□ 파라미터 접근 실습용 추적정보를 제공할 Advice 클래스

```

1  package tommy.spring.common;
2  import org.aspectj.lang.JoinPoint;
3  import tommy.spring.member.vo.UpdateInfo;
4  public class UpdateMemberInfoTraceAdvice {
5      public void traceReturn(JoinPoint joinPoint, boolean result,
6                             String memberId, UpdateInfo info) {
7          System.out.println("[TA] 정보 수정: 결과=" + result + ",대상회원=" +

```

	memberId + ",수정정보=" + info);
7	}
8	}

③ 스프링 설정 파일 작성

□ src/main/resources/applicationContextTwo.xml : bean, aop 네임스페이스 체크

	<!-- 상단 부분 생략 -->
1	<bean id="loggingAdvice" class="tommy.spring.common.LoggingAdvice" />
2	<bean id="cacheAdvice" class="tommy.spring.common.ArticleCacheAdvice" />
3	<bean id="traceAdvice" class="tommy.spring.common.UpdateMemberInfoTraceAdvice" />
4	<aop:config>
5	<aop:aspect id="loggingAspect" ref="loggingAdvice" order="1">
6	<aop:pointcut id="publicMethod"
	expression="execution(public * tommy.spring..*(..))" />
7	<aop:before method="before" pointcut-ref="publicMethod" />
8	<aop:after-returning method="afterReturning"
	pointcut-ref="publicMethod" returning="ret" />
9	<aop:after-throwing method="afterThrowing"
	pointcut-ref="publicMethod" throwing="ex" />
10	<aop:after method="afterFinally"
	pointcut-ref="publicMethod" />
11	</aop:aspect>
12	<aop:aspect id="cacheAspect" ref="cacheAdvice" order="2">
13	<aop:around
	pointcut="execution(public * *..ReadArticleServiceImpl.*(..))"
	method="cache" />
14	</aop:aspect>
15	<aop:aspect id="traceAspect" ref="traceAdvice" order="3">
16	<aop:after-returning
	pointcut="args(memberId,info)" method="traceReturn"
	returning="result" arg-names="joinPoint,result,memberId,info" />
17	</aop:aspect>
18	</aop:config>
19	<bean id="readArticleService" class="tommy.spring.board.service.ReadArticleServiceImpl" />
20	<bean id="memberService" class="tommy.spring.member.service.MemberServiceImpl" />
21	</beans>

④ 테스트용 메인 클래스 작성

1	package tommy.spring.board.controller;
2	import org.springframework.context.support.AbstractApplicationContext;
3	import org.springframework.context.support.ClassPathXmlApplicationContext;
4	import tommy.spring.board.service.ReadArticleService;
5	import tommy.spring.board.vo.ArticleNotFoundException;
6	import tommy.spring.board.vo.ArticleVO;
7	import tommy.spring.member.service.MemberService;
8	import tommy.spring.member.vo.UpdateInfo;
9	public class MainTwo {

```

10 public static void main(String[] args) {
11     String[] configLocations = new String[] { "applicationContextTwo.xml" };
12     AbstractApplicationContext context =
13         new ClassPathXmlApplicationContext(configLocations);
14     ReadArticleService readArticleService =
15         context.getBean("readArticleService", ReadArticleService.class);
16     try {
17         ArticleVO article1 = readArticleService.getArticleAndIncreaseReadCount(1);
18         ArticleVO article2 = readArticleService.getArticleAndIncreaseReadCount(1);
19         System.out.println("article1 == article2 : " + (article1 == article2));
20         readArticleService.getArticleAndIncreaseReadCount(0);
21     } catch (ArticleNotFoundException e) {
22     }
23     MemberService memberService =
24         context.getBean("memberService", MemberService.class);
25     memberService.update("javaline", new UpdateInfo());
26     context.close();
27 }

```

⑤ MainTwo 클래스 실행 및 결과 확인

```

<terminated> MainTwo [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (2019. 9. 14. 오후 1:47)
[LA] 메서드 실행 전 전처리 수행
[ACA] 캐시에 Article[1] 추가함
[LA] 메서드 실행 후 후처리 수행, 리턴값=tommy.spring.board.vo.ArticleVO@6d2a209c
[LA] 메서드 실행 완료
[LA] 메서드 실행 전 전처리 수행
[ACA] 캐시에서 Article[1] 구함
[LA] 메서드 실행 후 후처리 수행, 리턴값=tommy.spring.board.vo.ArticleVO@6d2a209c
[LA] 메서드 실행 완료
article1 == article2 : true
[LA] 메서드 실행 전 전처리 수행
[LA] 메서드 실행 중 예외 발생, 예외=tommy.spring.board.vo.ArticleNotFoundException
[LA] 메서드 실행 완료
[LA] 메서드 실행 전 전처리 수행
MemberServiceImpl.update() 메서드 실행
[TA] 정보 수정: 결과=true,대상회원=javaline,수정정보=tommy.spring.member.vo.UpdateInfo@3cc2931c
[LA] 메서드 실행 후 후처리 수행, 리턴값=true
[LA] 메서드 실행 완료

```

- ☐ 지금까지 우리는 XML 설정을 이용한 AOP 설정을 살펴보았다. 이제 위에서 만든 예제를 어노테이션 설정으로 바꾸어 보자.

5. 어노테이션을 활용한 AOP 설정

① @Aspect 어노테이션을 이용한 AOP

- ☐ @Aspect 어노테이션을 사용하면 XML 파일에 Advice 및 Pointcut 등의 설정을 하지 않고도 자동으로 Advice를 적용할 수 있다.
- ☐ XML 스키마 기반의 AOP와 의 차이점
 - @Aspect 어노테이션을 이용해서 Aspect 클래스를 구현한다. 이때 Aspect 클래스는 Advice를 구현한 메서드와 Pointcut을 포함한다.
 - XML 설정에서 <aop:aspectj-autoproxy /> 태그를 설정 한다.

② Aspect 클래스 작성하기

```
1 package tommy.spring.common;
2 import org.aspectj.lang.ProceedingJoinPoint;
3 import org.aspectj.lang.annotation.Around;
4 import org.aspectj.lang.annotation.Aspect;
5 import org.aspectj.lang.annotation.Pointcut;
6 import org.springframework.core.annotation.Order;
7 @Aspect
8 @Order(3)
9 public class ProfilingAspect {
10     @Pointcut("execution(public * tommy.spring..*(..))")
11     private void profileTarget() {
12     }
13     @Around("profileTarget()")
14     public Object trace(ProceedingJoinPoint joinPoint) throws Throwable {
15         String signatureString = joinPoint.getSignature().toShortString();
16         System.out.println(signatureString + " 시작");
17         long start = System.currentTimeMillis();
18         try {
19             Object result = joinPoint.proceed();
20             return result;
21         } finally {
22             long finish = System.currentTimeMillis();
23             System.out.println(signatureString + " 종료");
24             System.out.println(signatureString + " 실행 시간 : " + (finish - start) + "ms");
25         }
26     }
27 }
```

③ 스프링 설정파일 작성하기

- ☐ src/main/resources/applicationContextThree.xml : bean, aop 네임스페이스 체크

```
<!-- 상단 부분 생략 -->
1 <aop:aspectj-autoproxy />
2 <bean id="performanceTraceAspect" class="tommy.spring.common.ProfilingAspect" />
3 <bean id="writeArticleService" class="tommy.spring.board.service.WriteArticleServiceImpl">
4     <constructor-arg><ref bean="articleDao" /></constructor-arg>
5 </bean>
6 <bean id="articleDao" class="tommy.spring.board.dao.OracleArticleDAO" />
7 <bean id="memberService" class="tommy.spring.member.service.MemberServiceImpl" />
8 </beans>
```

④ 테스트용 실행 파일 작성

```
1 package tommy.spring.board.controller;
2 import org.springframework.context.support.AbstractApplicationContext;
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```

4 import tommy.spring.board.service.WriteArticleService;
5 import tommy.spring.board.vo.ArticleVO;
6 import tommy.spring.member.service.MemberService;
7 import tommy.spring.member.vo.MemberVO;
8 public class MainThree {
9     public static void main(String[] args) {
10         String[] configLocations = new String[] { "applicationContextThree.xml" };
11         AbstractApplicationContext context =
12             new ClassPathXmlApplicationContext(configLocations);
13         WriteArticleService articleService =
14             (WriteArticleService) context.getBean("writeArticleService");
15         articleService.write(new ArticleVO());
16         MemberService memberService =
17             context.getBean("memberService", MemberService.class);
18         memberService.regist(new MemberVO());
19         context.close();
20     }
21 }

```

⑤ MainThree 클래스 실행 및 결과 확인

6. @Aspect 어노테이션을 활용하여 다양한 Advice 적용하기

① 포인트 컷 클래스 작성하기

- ☐ @Pointcut 어노테이션은 Pointcut을 정의하는 AspectJ 표현식 값을 갖는다.
- ☐ @Pointcut 어노테이션을 적용한 메서드는 리턴 타입이 void여야 한다.
- ☐ @Pointcut 어노테이션을 이용해서 Pointcut을 정의하면 Advice 관련 어노테이션에서 해당 메서드 이름을 이용해서 Pointcut을 사용할 수 있다.

```

1 package tommy.spring.common;
2 import org.aspectj.lang.annotation.Pointcut;
3 public class PublicPointcut {
4     @Pointcut("execution(public * tommy.spring..*(..))")
5     public void publicMethod() {
6     }
7 }

```

② 로그서비스를 제공할 Aspect 클래스 작성하기


```

16         if (article != null) {
17             System.out.println("[ACA] 캐시에서 Article[" + id + "] 구함");
18             return article;
19         }
20         ArticleVO ret = (ArticleVO) joinPoint.proceed();
21         if (ret != null) {
22             cache.put(id, ret);
23             System.out.println("[ACA] 캐시에 Article[" + id + "] 추가함");
24         }
25         return ret;
26     }
27     @Override
28     public int getOrder() {
29         return 2;
30     }
31 }

```

④ 회원 정보변경 추적 서비스를 제공할 Aspect 클래스 작성하기

```

1 package tommy.spring.common;
2 import org.aspectj.lang.JoinPoint;
3 import org.aspectj.lang.annotation.AfterReturning;
4 import org.aspectj.lang.annotation.Aspect;
5 import tommy.spring.member.vo.UpdateInfo;
6 @Aspect
7 @Order(3)
8 public class UpdateMemberInfoTraceAspect {
9     @AfterReturning(pointcut = "args(memberId,info)", returning = "result",
10                     argNames = "result,memberId,info")
11     public void traceReturn(JoinPoint joinPoint, boolean result, String memberId,
12                             UpdateInfo info) {
13         System.out.println("[TA] 정보 수정: 결과=" + result + ",대상회원=" + memberId +
14                             ",수정정보=" + info);
15     }
16 }

```

⑤ 참고 : 타입을 이용한 파라미터 접근

- ☐ JoinPoint의 getArgs() 메서드를 이용하면 대상 객체의 메서드를 호출할 때 사용한 인자에 접근할 수 있다고 했는데 JoinPoint를 사용하지 않고 Advice 메서드에서 직접 파라미터를 이용해서 메서드 호출 시 사용된 인자에 접근할 수 있다.
 - ☒ Advice 구현 메서드에 인자를 전달받을 파라미터를 명시한다.
 - ☒ Pointcut 표현식에 args() 명시자를 사용해서 인자 목록을 지정한다.
- ☐ Advice 구현 메서드에 사용할 파라미터를 명시한다.
 - ex) public void traceReturn(String memberId, UpdateInfo info){ ...
- ☐ XML 설정파일에서 args() 명시자를 이용해서 인자목록을 지정한다.

ex) <aop:after-returning pointcut="args(memberId, info)" method="traceReturn"/>

- ☐ 위 설정에서 args() 명시자가 의미하는 것은 대상 객체의 메서드 호출 시 인자가 두 개 전달되고, 이중 첫 번째 파라미터는 traceReturn()의 memberId 파라미터 타입이고 두 번째 인자는 info 파라미터 타입이다.
- ☐ args() 명시자의 경우 메서드 정의에 있는 타입이 아닌 실제 메서드 호출 시 전달되는 인자의 타입에 따라서 적용여부가 결정된다.
- ☐ @Aspect 어노테이션을 사용하는 경우에도 XML 스키마를 사용하는 경우와 마찬가지로 Pointcut 표현식에 args() 명시자를 사용하면 된다.

■ 인자의 이름 매핑 처리

- ☐ args() 명시자에 지정한 이름과 Advice 구현 메서드의 파라미터 이름이 일치하는지의 여부를 확인하는 순서
 - ① Advice 어노테이션 태그의 argNames 속성이나 Advice 설정 XML 스키마의 args-names 속성에서 명시한 파라미터 이름을 사용한다.
 - argName 속성은 Advice 구현 메서드의 파라미터 이름을 입력할 때 사용
ex) @AfterReturning(pointcut="args(memberId, info)", argNames="memberId,info")
 - 첫 번째 파라미터 타입이 JoinPoint나 ProceedingJoinPoint라면 첫 번째 파라미터를 제외한 나머지 파라미터의 이름을 argNames 속성에 입력해 주면 된다.
 - XML 스키마를 사용하는 경우에는 arg-names 속성을 이용해서 파라미터 이름을 지정하면 된다.
ex) <aop:after-returning pointcut=".." method=".." arg-names="result,memberId,info">
 - ② 디버그 옵션을 이용해서 컴파일할 때 생성되는 디버그 정보를 이용해서 파라미터 이름이 일치하는지의 여부를 확인한다.
 - ③ 디버그 옵션이 없을 경우 파라미터 개수를 이용해서 일치 여부를 확인한다.
 - ④ 위의 3가지 모두 일치하지 않을 경우 IllegalArgumentException 예외를 발생한다.

⑥ 스프링 설정파일 작성하기

- ☐ @Aspect 어노테이션을 이용해서 Aspect 클래스를 작성했다면 XML 설정파일에 Aspect 클래스를 빈으로 등록해야 한다.
- ☐ <aop:aspectj-autoproxy/> 태그를 사용하면 @Aspect 어노테이션이 적용된 빈 객체를 Aspect로 사용한다.

- ☐ src/main/resources/applicationContextFour.xml : bean, aop 네임스페이스 체크

	<!-- 상단 부분 생략 -->
1	<aop:aspectj-autoproxy />
2	<bean id="traceAspect" class="tommy.spring.common.UpdateMemberInfoTraceAspect" />
3	<bean id="cacheAspect" class="tommy.spring.common.ArticleCacheAspect" />
4	<bean id="loggingAspect" class="tommy.spring.common.LoggingAspect" />
5	<bean id="readArticleService" class="tommy.spring.board.service.ReadArticleServiceImpl" />
6	<bean id="memberService" class="tommy.spring.member.service.MemberServiceImpl" />
7	</beans>

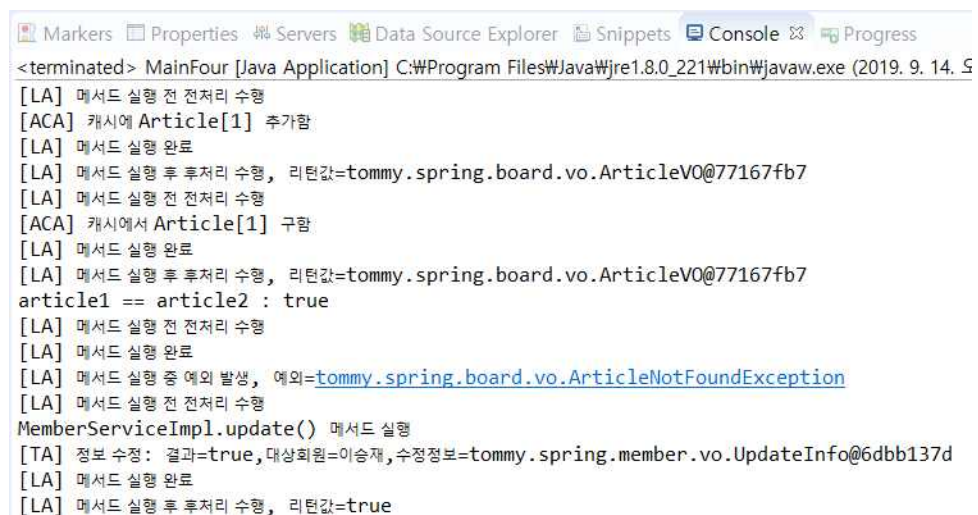
⑦ 테스트 클래스 작성

```

1 package tommy.spring.board.controller;
2 import org.springframework.context.support.AbstractApplicationContext;
3 import org.springframework.context.support.ClassPathXmlApplicationContext;
4 import tommy.spring.board.service.ReadArticleService;
5 import tommy.spring.board.vo.ArticleNotFoundException;
6 import tommy.spring.board.vo.ArticleVO;
7 import tommy.spring.member.service.MemberService;
8 import tommy.spring.member.vo.UpdateInfo;
9 public class MainFour {
10     public static void main(String[] args) {
11         String[] configLocations = new String[] { "applicationContextFour.xml" };
12         AbstractApplicationContext context =
13             new ClassPathXmlApplicationContext(configLocations);
14         ReadArticleService readArticleService = context.getBean("readArticleService",
15             ReadArticleService.class);
16
17         try {
18             ArticleVO article1 = readArticleService.getArticleAndIncreaseReadCount(1);
19             ArticleVO article2 = readArticleService.getArticleAndIncreaseReadCount(1);
20             System.out.println("article1 == article2 : " + (article1 == article2));
21             readArticleService.getArticleAndIncreaseReadCount(0);
22         } catch (ArticleNotFoundException e) {
23         }
24         MemberService memberService =
25             context.getBean("memberService", MemberService.class);
26         memberService.update("이승재", new UpdateInfo());
27         context.close();
28     }
29 }

```

⑧ MainFour 클래스 실행 및 결과 확인



```

<terminated> MainFour [Java Application] C:\Program Files\Java\jre1.8.0_221\bin\javaw.exe (2019. 9. 14. 5
[LA] 메서드 실행 전 전처리 수행
[ACA] 캐시에 Article[1] 추가함
[LA] 메서드 실행 완료
[LA] 메서드 실행 후 후처리 수행, 리턴값=tommy.spring.board.vo.ArticleVO@77167fb7
[LA] 메서드 실행 전 전처리 수행
[ACA] 캐시에서 Article[1] 구함
[LA] 메서드 실행 완료
[LA] 메서드 실행 후 후처리 수행, 리턴값=tommy.spring.board.vo.ArticleVO@77167fb7
article1 == article2 : true
[LA] 메서드 실행 전 전처리 수행
[LA] 메서드 실행 완료
[LA] 메서드 실행 중 예외 발생, 예외=tommy.spring.board.vo.ArticleNotFoundException
[LA] 메서드 실행 전 전처리 수행
MemberServiceImpl.update() 메서드 실행
[TA] 정보 수정: 결과=true,대상회원=이승재,수정정보=tommy.spring.member.vo.UpdateInfo@6dbb137d
[LA] 메서드 실행 완료
[LA] 메서드 실행 후 후처리 수행, 리턴값=true

```