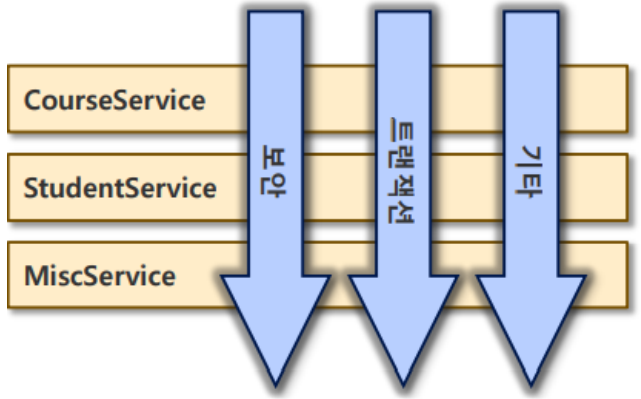
제8장 스프링 AOP 프레임워크

- **AOP** 개념
- 중요용어정리
- AOP 구현

AOP 개념 (1)

- 스프링 AOP(Aspect Oriented Programming)의 개요
  - 여러객체에서 공통으로 적용되는 공통관심사항(Cross-cutting Concern)
  - 주요 업무 처리만 수행하는 핵심관심사항(Core Concern)
  - 애플리케이션의 다양한 관점(point)에 영향을 미치는 기능



각각의 모듈의 주요 관심사는 특정 도메인에 대해서 서비스를 제공하는 것임
하지만, 이들 각각의 모듈들은 보안이나 트랜잭션 관리와 같은 유사한 보조적인(ancillary) 기능을 필요로 함

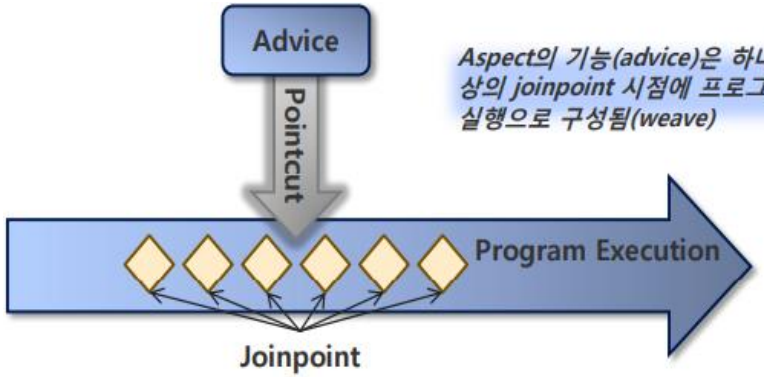
- 기존 OOP 에서는 공통관심사항을 여러 모듈에서 적용하는데 중복된 코드를 양산하는 한계가 존재
  - 이를 극복하기 위해 AOP 가 등장
- Aspect Oriented Programming은 문제를 해결하기 위한 핵심 관심 사항과 전체에 적용되는 공통 관심 사항을 기준으로 프로그래밍함으로써 공통 모듈을 손쉽게 적용할 수 있게 해준다.

- 핵심(Core)관심사항 : CourseService, StudentService, MiscService
- 공통(Cross-cutting)관심사항 : Security, Transactions, Other
- 핵심관심사항에 공통관심사항을 어떻게 적용시킬 것인가...

AOP 개념 (2)

- **Aspect**
  - 객체지향 개념의 상속과 위임
    - 상속은 애플리케이션을 통해 사용되는 동일한 기반 클래스라는 깨지기 쉬운 객체 계층 구조를 사용하게 됨
    - 위임은 위임 객체로의 복잡한 호출이 필요할 수 있기 때문에 다루기 어려움
  - Aspect-oriented
    - 많은 환경에서 더 명확해질 수 있는 상속과 위임의 대안으로 사용됨
    - AOP를 사용해서 한 곳에 공통된 기능을 정의하며, 새로운 특성을 적용하려는 클래스를 변경하지 않고 해당 기능을 어떻게 어디에 적용할지를 선언적으로 정의함
    - Cross-cutting concern은 aspect라고 하는 특별한 객체로 모듈화가 가능함
    - 장점
      - 각각의 관심사(concern)에 대한 로직은 코드 여기저기에 모두 흩어져있는 것과는 반대로 한 군데에 모여 있음
      - 서비스 모듈들은 주요 관심사(concern) (혹은 핵심 기능)에 대한 코드만을 포함하고 부수적인 관심사(concern)는 aspect로 옮겨지기 때문에 더 명확해짐

AOP 개념 (3)



- 스프링 AOP 용어 정의
  - **Advice**
    - 전기 검침 기사
      - 가정의 전기 사용량(KW)을 점검하여 전기 회사에 통보해 줌
      - 방문할 가정에 대한 리스트를 작성하고, 보고할 정보도 중요하지만, 이들의 실질적인 주요 업무(job)는 전기 사용량을 기록하는 것임
    - **Aspect의 업무(job)을 advice라고 함**
    - Aspect가 무엇인가와 언제인가 모두를 정의함
      - Aspect가 수행할 업무(job)에 대해서 설명하는 것 뿐만 아니라, 해당 업무(job)을 언제 수행할지에 대한 내용도 정의함
      - 메소드가 호출되기 전에 적용할 것인가? 메소드 호출 후에 적용할 것인가? 메소드 호출 전후 모두에 적용할 것인가? 혹은 메소드가 예외를 발생할 때에만 적용할 것인가?
    - 정리하면.. 어느 시점(메소드 수행전과 후, 예외발생 후 등)에 어떤 공통 관심 기능(Aspect)을 적용할지 정의하는 것(Before Advice, After returning Advice, After throwing Advice, Around Advice)

스프링 AOP구현 방법

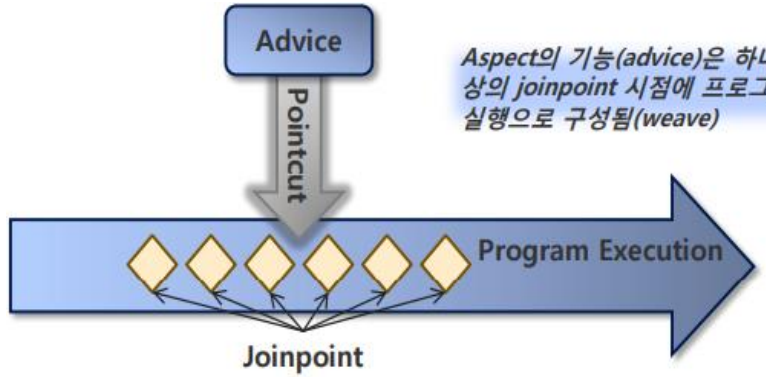
```
class 클래스명 implements   MethodInterceptor      {
        @Override
        public Object invoke(MethodInvocation invocation) throws Throwable{
                ... 보조업무
                Object result = invocation.proceed();
                ... 보조업무
                return result;
        }
}
```
// Around Advice

```
class 클래스명 implements  MethodBeforeAdvice  {
        @Override
        public void before(Method method, object[] args, Object target) throws Throwable{ }
}
```
// Before Advice

```
class 클래스명 implements  AfterReturningAdvice   {
        @Override
        public void afterReturning(Object returnValue, Method method, Object[] args, Object target) throws  Throwable{ }
}
```
// After returning Advice

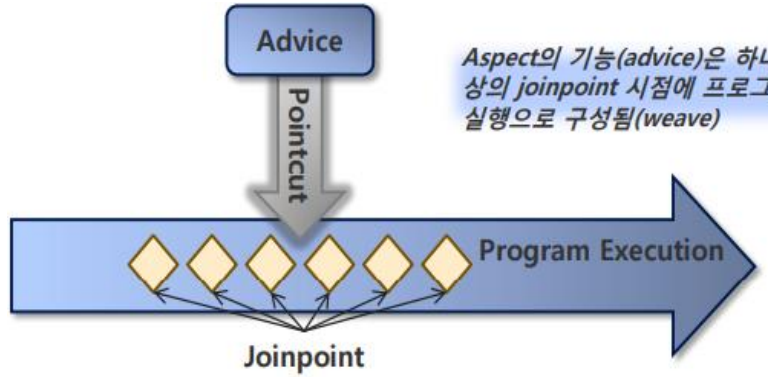
```
class 클래스명 implements   ThrowsAdvice  {
        @Override
        public void afterThrowing(예외처리 e) throws Throwable{ }
}
```
// After throwing Advice

AOP 개념 (4)



- Aspect 용어 정의
  - **Joinpoint**
    - 전기 검침 기사
      - 각각의 가정은 전기 검침기를 가지고 있으며 이들 가정이 전기 검침 기사의 잠재적인 대상이 됨
      - 전기 검침 기사는 모든 종류의 검침기를 읽는 능력을 가지고 있지만, 본인의 업무를 수행하려면 가정에 설치된 전기 검침기를 읽어야 함
    - 애플리케이션은 advice가 적용할 수많은 기회(opportunity)들을 가지고 있는데 이러한 기회를 jointpoint 라고 함
    - Aspect가 플러그인 될 수 있는 애플리케이션 실행에서의 지점(point) 임
      - 이러한 지점은(point) 호출되는 **메소드가 될 수도 있으며, 발생되는 예외, 혹은 변경하려는 필드**도 될 수 있음
      - Aspect의 코드가 새로운 행위를 추가하기 위해서 애플리케이션의 일반적인 흐름으로 삽입되는 지점임
    - 정리하면.. Aspect가 적용될 수 있는 지점(메소드)

AOP 개념 (5)



- Aspect 용어 정의
  - **Pointcut**
    - 전기 검침 기사
      - 한 명의 전기 검침 기사가 전기 회사에서 서비스를 제공하는 모든 가정을 방문할 수 없음
      - 방문할 모든 가정을 나누어서 할당함
    - 하나의 aspect는 애플리케이션의 모든 joinpoint들에 대해서 advice를 수행할 필요는 없음
    - Pointcut 은 aspect가 advice를 수행할 joinpoint의 **범위**를 줄이는 데에 도움을 줌
      - Advice가 aspect의 역할(what)과 시점(when)을 정의한다면, pointcut은 지점(where)을 정의함
      - Pointcut 정의는 적용되어야 하는 advice의 하나 이상의 joinpoint와 일치함
      - 구체적인 클래스와 메소드 명 혹은 클래스와 메소드 이름 형태(pattern)와 일치하도록 정의하는 정규 표현식(regular expression)을 사용해서 pointcut을 지정함
      - 어떤 AOP 프레임워크에서는 메소드 파라미터의 값과 같은 실행시 결정되는 것을 기반으로 하는 advice를 적용할지 안할지를 결정하는 동적인 pointcut 생성이 가능함
    - 정리하면.. 공통 관심 사항이 적용될 Joinpoint 적용

Pointcut을 사용하려면..

- org.springframework.aop.support.NameMatchMethodPointcut
- org.springframework.aop.support.DefaultPointcut

⇒ org.springframework.aop.support.NameMatchMethodPointcutAdvisor

⇒ org.springframework.aop.support.RegexMethodPointcutAdvisor // 정규표현식가능

AOP 개념 (6)

- Aspect 용어 정의
  - **Aspect**
    - 전기 검침 기사
      - 작업 시작 시, 어떤 일을 할 것인지(전기 사용량 측정)와 어떤 가정에서 정보를 수집할 것인지를 알고 있음
      - 자신의 작업(job)을 수행하는데 필요한 모든 것을 알고 있음
    - Aspect는 **advice와 pointcut의 통합하는 것**(Advisor)
      - Advice와 pointcut은 aspect에 대해서 알고 있는 모든 것을 정의함
      - 수행할 것이 무엇이며, 어디서 언제 수행할지
    - 정리하면.. 여러객체에서 공통으로 적용되는 공통 관심 사항(트랜잭션, 로깅, 보안 등) 처리

AOP 개념 (7)

- Aspect 용어 정의
  - **Target**
    - Advice가 수행될 객체를 의미함
    - 특정 행위를 추가하기 원하는 객체임
      - AOP를 사용하지 않으면 해당 객체는 주요 로직과 cross-cutting 에 대한 **로직 모두를 포함**해야 함
      - **AOP를 사용하면** 대상 객체는 **주요 관심사에만 초점**을 두고, 적용할 advice에 대해서 모르는 채로 있음
  - **Proxy**
    - 대상(target) 객체에 advice를 적용한 후에 생성되는 객체임
    - 클라이언트 객체에 한정해서 대상 객체 (pre-AOP)와 proxy 객체(post-AOP)는 동일함
      - 애플리케이션의 나머지는 proxy 객체를 지원하는 변경을 수행하지 못함

자바 Proxy클래스 설정

```
                                    // Proxy.newProxyInstance(loader, interfaces, h) ;
인터페이스명  proxy객체변수명 = (인터페이스명)Proxy.newProxyInstance(
          실제적용클래스명.class.getClassLoader()
          ,new Class[] {인터페이스명.class}
          ,new InvocationHandler() { // 익명클래스 작성
           @Override
          public Object invoke(Object proxy, Method method, Object[] args) throws Throwable
         {              ... 보조업무
                    Object   m =  method.invoke(업무인터페이스명, args);
                    ... 보조업무
                    return    m;
          }
     }
 );
```

AOP 개념 (8)

- Aspect 용어 정의
  - **Weaving**
    - proxy되는 객체를 생성하는 대상(target) 객체에게 aspect를 적용하는 과정(process)임
      - Aspect는 특정 joinpoint 에 대상 (target) 객체로 구성됨
    - 대상 객체의 생명주기 내에서 여러 지점에 발생될 수 있음
      - 컴파일 시점 – aspect는 대상(target) 클래스가 컴파일 될 때 구성됨. 특별한 컴파일러를 필요로 함. AspectJ의 weaving 컴파일러
      - 클래스 로드 시점 – 대상(target) 클래스가 JVM으로 로딩될 때 aspect를 구성함. 클래스가 애플리케이션으로 소개되기 전에 대상(target) 클래스의 바이트코드를 처리하는 특별한 ClassLoader를 필요로 함.
      - 실행 시점 – 애플리케이션의 실행 중간에 구성됨. AOP 컨테이너는 aspect에서 구성하는(weaving) 동안에 대상(target) 객체에게 위임하는 proxy 객체를 동적으로 생성함. 이 방법이 Spring AOP의 aspect가 구성되는 방법임
    - 정리하면.. 어떤Advice를 어떤 Pointcut(핵심사항)에 적용시킬 것인지에 대한 설정(Advisor)

AOP 개념 (9)

- Spring의 AOP 구현 방법
  - AOP 프레임워크
    - AspectJ (eclipse.org/aspectj)
    - Jboss AOP (labs.jboss.com/portal/jbossaop/index.html)
    - Spring AOP (www.springframework.org)
  - AOP 구현의 세가지 방법
    - POJO Class를 이용한 AOP구현
    - 스프링 API를 이용한 AOP구현
    - 애노테이션(Annotation) 을 이용한 AOP 구현

스프링 AOP와 AspectJ비교

구분	스프링 AOP	AspectJ
목표	간단한 AOP기능 제공	완벽한 AOP기능 제공
Joinpoint	메소드만 지원	생성자, 필드, 메소드 지원
Weaving	런타임시에만 가능 (RTW:Runtime Weaving)	런타임은 제공하지 않음. 컴파일시 위빙(CTW:Compile Time Weaving, 자바컴파일러를 확장한 형태의 컴파일러), 클래스 로드시 위빙(LTW:Load Time Weaving) 제공
대상	스프링컨테이너가 관리하는 빈에서만 가능	모든 자바 오브젝트에서 가능