1. Bean 의 라이프 사이클

Bean 객체 생성 → DI → Init() 초기화

1. AOP 기초
2. AOP 개념

관점 지향 프로그래밍 이란 ?

개별 프로그래밍 언어는 프로그램 개발을 위해 고유한 관심사 분리(Separation of Concerns) 패러다임을 갖는다. 예를 들면 절차적 프로그래밍은 상태값을 갖지 않는 연속된 함수들의 실행을 프로그램으로 이해하고 모듈을 주요 분리 단위로 정의한다.

객체지향 프로그래밍은 일련의 함수 실행이 아닌 상호작용하는 객체들의 집합으로 보며 클래스를 주요 단위로 한다.   
객체지향 프로그래밍은 많은 장점에도 불구하고, 다수의 객체들에 분산되어 중복적으로 존재하는 공통 관심사가 존재한다. 이들은 프로그램을 복잡하게 만들고, 코드의 변경을 어렵게 한다.

관점 지향 프로그래밍(AOP, Apsect-oriented programming)은 이러한 객체지향 프로그래밍의 문제점을 보완하는 방법으로 핵심 관심사를 분리하여 프로그램 모듈화를 향상시키는 프로그래밍 스타일이다. AOP는 객체를 핵심 관심사와 횡단 관심사로 분리하고, 횡단 관심사를 관점(Aspect)이라는 모듈로 정의하고 핵심 관심사와 엮어서 처리할 수 있는 방법을 제공한다.

* + 관점(Aspect)은 프로그램의 핵심 관심사에 걸쳐 적용되는 공통 프로그램 영역을 의미한다. 예를 들면 로깅, 인증, 권한확인, 트랜잭션은 비지니스 기능 구현시에 공통적으로 적용되는 요소이며 하나의 관점으로 정의될 수 있다.
  + 핵심 관심사(Core concern)는 프로그램을 작성하려는 핵심 가치와 목적이 드러난 관심 영역으로 보통 핵심 비지니스 기능에 해당한다.
  + 횡단 관심사(Cross-cutting concern)는 핵심 관심에 영향을 주는 프로그램의 영역으로, 로깅과 트랜잭션, 인증처리와 같은 시스템 공통 처리 영역이 해당된다.

1. 자체 개념 정리

객체 지향 프로그램은 클래스를 기준으로 클래스의 Object 를 상호작용을 중심으로

프로그래밍을 하여 기존 코드의 재사용성을 추구하는 프로그래밍 방법이다.

주로 객체지향 언어에는 객체 지향 프로그래밍을 구현하는 방법 중에 상속, 추상, 인터페이스 등등의 개념들을 이용하여 Object들간의 관계를 이용하여 구현된다.

하지만 OOP 의 많은 장점에도 불구하고 다른 클래스의 로직들과 중복되는 로직 부분 들을 처리하기 위해서 관점 지향 프로그래밍에서는 기존의 클래스들의 중복이 되는 횡단 관심사(Cross-cutting concern )와 핵심관심사(Core concern) 로 분리하여 가장 핵 심적인 기능은 핵심 관심사 로직으로 분리하고 , 다른 클래스에서 적용될수 있는 로깅,

트랜잭션 , 보안 등 같은 여러 영역에서 적용될수 있는 중복적인 비지니스 로직을 횡단 관심사로 분류한다.

SPING Framework에서는 이러한 핵심로직들에 횡단 관심사가 되는 로직을 적용할수 있는 AOP 적인 기능들을 지원한다.

1. AOP 용어 정리

|  |  |
| --- | --- |
| JoinPoint | 횡단 관심사를 적용할수 있는 모든 지점을 말한다.  메소드 호출, 필드 값 변경 |
| PointCut | JoinPoint의 부분 집합으로 실제로 적용되는 JoinPoint의 그룹으로 지정(범위 기준으로)하여 적용하는 것을 말한다. |
| Advice | 언제 공통 관심 기능(횡단 관심사)을 핵심 로직에 적용할지를 말하는데 크렉 기준으로 언제 그 작업을 실행할것인지를 정하는 것이다. |
| Weaving | Advice를 핵심 로직 코드에 적용하는 것을 weaving이라고 한다. |
| Aspect(adpect) | 여러 객체에 공통으로 적용되는 기능을 Aspect 라고 한다.  예 )트랜잭션, 보안 등 |

1. Advice

|  |  |
| --- | --- |
| Before Advice | 대상 객체의 메서드의 호출전에 공통 기능을 실행한다. |
| After Advice | 대상 객체의 익센셥 여부와 상관없이 메서드 실행후에 공통기능을 실행한다. |
| After Returning Advice | 대상 객체의 익셉션이 없이 실행되면 공통 기능 실행 |
| After Throwing Advice | 대상 객체의 익셉션이 발생한 이후에 공통 기능 실행 |
| Around Advice | 대상 객체의 메서드 실행 전, 후 또는 익센셥 발생 시점에 공통 기능을 실행하는데 사용된다.(반드시 JoinPoint 정보를 생략하면 안된다! - 정보가 있어야 실행가능하다 ! ^^ ) |

7. ProxyFactoryBean