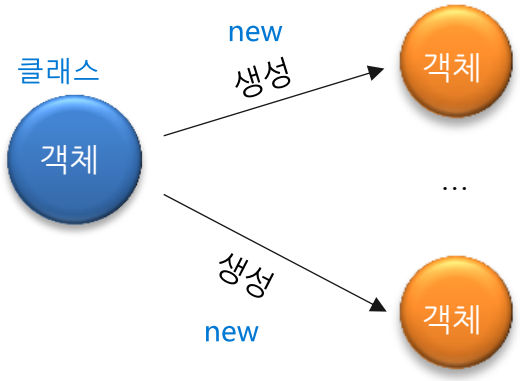
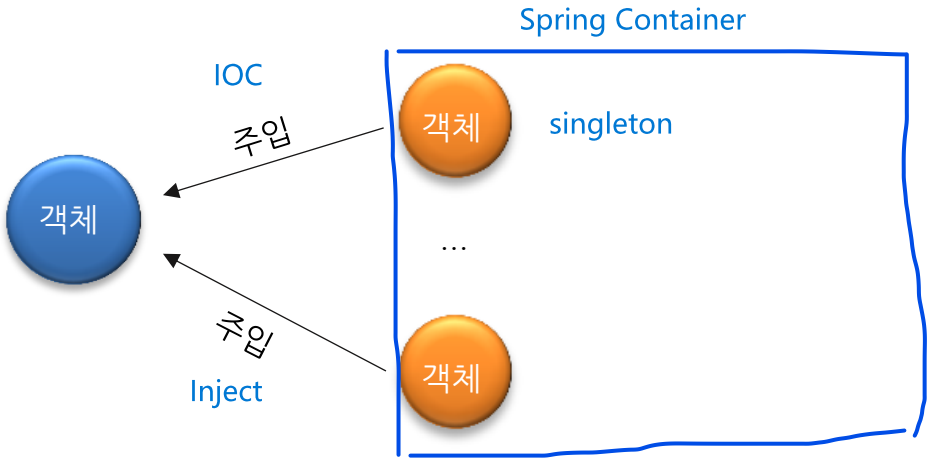
2장 IoC/DI와 AOP

목차

1. IoC/DI

- IoC Inversion of Control 는 객체의 생성과 생명주기를 컨테이너에게 위임하는 형태로 객체의 제어가 바뀜을 의미하는 제어의 역행
- DI Dependency Injection 는 의존성 주입으로 컨테이너로 부터 객체를 주입 받는 기법
- IoC/DI를 이용하면 객체 생성과 의존 관계 처리를 컨테이너가 담당하기 때문에 낮은 결합도의 컴포넌트 구현





2. 스프링 컨테이너

- 스프링 컨테이너 Spring Container는 스프링 애플리케이션을 구성하는 빈 Bean(객체)을 생성, 관리 및 제공하는 역할
- 스프링 빈 Spring Bean은 스프링 컨테이너로 관리하는 자바 객체로 빈 등록은 xml, Annotaion, 설정 클래스 사용
- ApplicationContext는 Bean을 싱글톤으로 관리하는 스프링 컨테이너

주요 어노테이션	설명
@Configuration	설정 클래스 정보를 바탕으로 스프링 컨테이너 생성
@ComponentScan	특정 패키지나 클래스를 기준으로 컴포넌트들을 스캔하여 빈으로 등록
@Bean new -> 등록	설정 클래스에서 사용하는 외부 라이브러리를 Bean으로 등록
@Component new -> 등록	• 사용자 정의 클래스를 Bean으로 등록 • @Service, @Controller, @Repository 등
@Autowired	• 적합한 데이터 타입을 이용해서 의존 객체를 자동으로 주입 • @Resource, @Inject 등

3. 의존성 주입

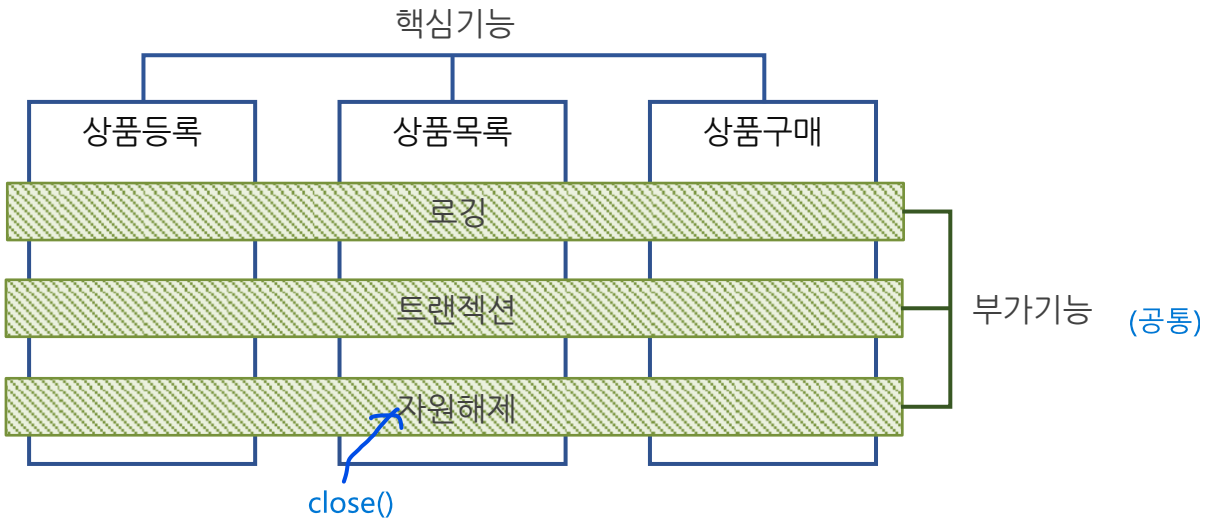
- 의존성 주입DI, Dependency Injection 객체가 직접 의존하는 객체를 생성하거나 관리하지 않고 외부로부터 주입 받는 디자인 패턴
- 의존성 주입은 객체 지향 프로그래밍에서 코드의 결합도를 낮추고 유연성을 높이는 데 사용
- DI는 객체지향 설계의 중요한 원칙인 SOLID 원칙을 준수하도록 돕는 중요한 메커니즘

의존성 주입 방식	설명
생성자 주입 final 속성	• 객체를 생성할 때 생성자를 통해 의존성을 주입하는 방식 • 주입할 의존성을 생성자의 매개변수로 전달하여 객체를 생성 생성자 : 객체 초기화
세터 주입	• 객체를 생성한 후에 setter 메서드를 통해 의존성을 주입하는 방식 • 주입할 의존성을 객체의 setter 메서드를 호출하여 설정
필드 주입	• 객체의 필드에 직접 의존성을 주입하는 방식 • 필드에 주입할 의존성을 선언

스프링 컨테이너 안의 객체가 밖의 객체로
@autowired를 통해 주입

4. 핵심 기능과 부가 기능

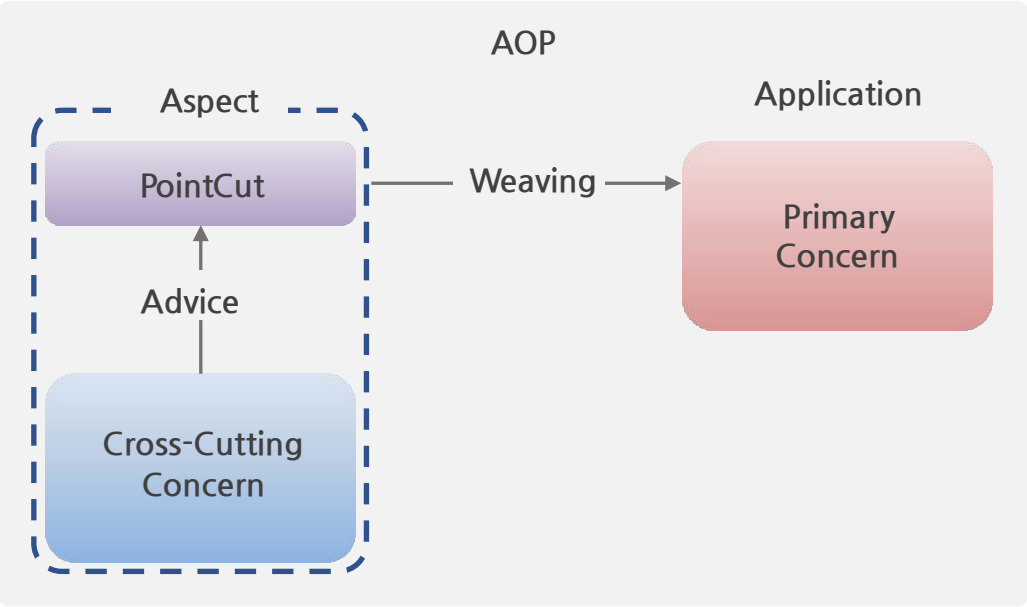
- 업무 Business로직은 핵심 기능 Core Concern과 부가 기능 Cross-cutting Concern으로 구성
- 핵심 기능은 애플리케이션의 주요 비즈니스 로직 또는 주요 기능을 수행하는 모듈 또는 클래스
- 부가 기능은 핵심 기능을 보완하거나 지원하는 기능으로, 여러 핵심 기능에서 공통적으로 사용되는 기능



5. AOP 개요

↔OOP

- AOP Aspect Oriented Programming는 여러 객체에서 공통으로 사용하는 기능을 분리해서 재사용성을 높이는 프로그래밍 기법
- AOP는 핵심 기능에서 부가 기능을 분리해서 관심사 Aspect라는 모듈형태로 만들어서 설계하고 개발하는 방법
- AOP는 핵심 기능과 부가 기능 간의 결합도를 낮추고 코드의 재사용성과 유연성을 향상



6. AOP 어노테이션

- AOP 어노테이션을 사용해 핵심 기능 실행 전 후, 예외 발생 시점 등과 같은 특정 지점에서 부가 기능 적용

어노테이션	설명
@EnableAspectJAutoProxy	Spring에서 AspectJ를 사용하여 AOP를 활성화
@Aspect	• 관심사를 정의하는 클래스 설정 • Advice와 Pointcut 포함
@Pointcut	부가기능 Advice를 적용할 조인 포인트 지정
@Before	Advice를 메서드 실행 전에 실행하도록 지정
@After	Advice를 메서드 실행 후에 실행하도록 지정
@AfterReturning	Advice를 메서드가 성공적으로 반환된 후에 실행하도록 지정
@AfterThrowing	Advice를 메서드에서 예외가 발생한 후에 실행하도록 지정
@Around	Advice를 메서드 실행 전후에 적용하고, 메서드 호출을 직접 제어