**Spring Framework의 주요 전략**

- POJO(평범한 자바 객체)를 이용한 가볍고 비 침투적인 개발(특정 기술을 적용했을 때 그 기술을 사용하기 위한 코드나 규약 등이 사용되어야 하는 것.)

- DI(종속객체 주입)와 인터페이스 지향을 통한 느슨한 결합도

- 애스펙트와 공통 규약을 통한 선언적 프로그래밍

- 애스펙트와 템플릿을 통한 상투적인 코드 축소

* **DI(종속객체 주입)**

객체는 시스템에서 각 객체를 조율하는 제 3자에 의해 생성 시점에 종속객체가 부여된다. 즉, 객체는 직접 종속객체를 생성하거나 얻는게 아닌 종속객체가 필요한 객체에 주입된다.

이때 종속객체를 주입하는 방법을 와이어링이라 하고 XML 설정파일에 사용할 객체(빈)와 종속객체(빈)를 선언하고 관계를 설정해준다.

장점 : 단위 테스트가 가능해지며 종속객체를 사용하는 객체에서는 종속객체가 어떤 것인지 전혀 신경쓰지않고 개발할 수 있다.(-> 종속객체를 언제든 다른 구현체로 바꾸어 사용할 수 있다.)

* **AOP**

애스펙트 지향 프로그래밍은 애플리케이션 전체에 걸쳐 사용되는 기능(로깅, 트랜잭션 관리, 보안 등)을 재사용할 수 있는 컴포넌트에 담아 모듈화하여 다른 컴포넌트들에 선언적으로 적용할 수 있게 해준다.

장점 : 각 컴포넌트들은 공통 컴포넌트들에 대해서는 신경쓰지않고 본연의 관심사에 집중할 수 있다.

* **빈의 생명주기**

1. 인스턴스화
2. 프로퍼티 할당
3. BeanNameAware의 setBeanName()
4. BeanFactoryAware의 setBeanFactory()
5. ApplicationContextAware의 setApplicationContext()
6. BeanPostProcessor의 사전 초기화
7. InitializingBean의 afterPropertiesSet()
8. 커스텀 초기화 메소드 호출
9. BeanPostProcessor의 후속 초기화
10. 빈 사용 후 컨테이너 종료
11. DisposableBean의 destroy()
12. 커스텀 소멸 메소드 호출