## 다양한 의존성 주입 방법과 생성자 주입을 사용해야 하는 이유

- [ 생성자 주입 요약 및 정리 ]
  - 객체의 불변성을 확보할 수 있다.
  - 테스트 코드의 작성이 용이해진다.
  - final 키워드를 사용할 수 있고, Lombok과의 결합을 통해 코드를 간결하게 작성할 수 있다.
  - 스프링에 침투적이지 않은 코드를 작성할 수 있다.
  - 순환 참조 에러를 애플리케이션 구동(객체의 생성) 시점에 파악하여 방지할 수 있다.

## @Autowired 빈 탐색 전략과 @Qualifier와 @Primary

## 의존성 주입(Dependency Injection, DI)이란? 및 Spring이 의존성 주입을 지원하는 이유

- [ 의존성 주입(Dependency Injection) 정리 ]

> 한 객체가 어떤 객체(구체 클래스)에 의존할 것인지는 별도의 관심사이다. Spring은 의존성 주입을 도와주는 DI 컨테이너로써, 강하게 결합된 클래스들을 분리하고, 애플리케이션 실행 시점에 객체 간의 관계를 결정해 줌으로써 결합도를 낮추고 유연성을 확보해준다. 이러한 방법은 상속보다 훨씬 유연하다. 단, 한 객체가 다른 객체를 주입받으려면 반드시 DI 컨테이너에 의해 관리되어야 한다는 것이다.

- 두 객체 간의 관계라는 관심사의 분리
- 두 객체 간의 결합도를 낮춤
- 객체의 유연성을 높임
- 테스트 작성을 용이하게 함

## 빈을 찾기 위한 다양한 의존성 검색 방법, DL(Dependency LookUp)

## 필터(Filter) vs 인터셉터(Interceptor) 차이 및 용도 - (1)