DI란 무엇인가?

Dependency injection = 의존관계 주입

그럼 의존관계는 뭔데?

A가 B를 의존한다 = 의존대상 B가 변하면, 그것이 A에 영향을 미친다.

```
class BurgerChef {
    private HamBurgerRecipe hamBurgerRecipe;

    public BurgerChef() {
        hamBurgerRecipe = new HamBurgerRecipe();
    }
}
```

```
class BurgerChef {
    private BurgerRecipe burgerRecipe;

    public BurgerChef() {
        burgerRecipe = new HamBurgerRecipe();
        //burgerRecipe = new CheeseBurgerRecipe();
        //burgerRecipe = new ChickenBurgerRecipe();
    }
}

interface BugerRecipe {
    newBurger();
    // 이외의 다양한 메소드
}

class HamBurgerRecipe implements BurgerRecipe {
    public Burger newBurger() {
        return new HamBerger();
    }
    // ...
}
```

의존관계 주입을 충족하는 3가지 조건

- 클래스 모델이나 코드에는 런타임 시점의 의존관계가 드러나지 않는다. 그러기 위해서는 인터페이스만 의존하고 있어야 한다.

- 런타임 시점의 의존관계는 컨테이너나 팩토리 같은 제3의 존재가 결정한다.

- 의존관계는 사용할 오브젝트에 대한 레퍼런스를 외부에서 제공해줌으로써 만들어진다.

DI 방식

1. 필드 주입(field injection)
2. 메소드 주입(setter based injection)
3. 생성자 주입(constructor based injection)

생성자 주입을 사용해야함
왜? NullPointerException을 방지할 수 있고, 주입받을 필드를 final로 선언 가능하다.
+ 순환 참조를 방지

DI 장점

1. 의존성이 줄어든다
2. 재사용성이 높은 코드가 된다
3. 테스트하기 좋은 코드가 된다
4. 가독성이 높아진다

그렇다면 자주 함께 나오는 제어의 역전(inversion of control)은 무엇인가?

근본적으로는 디자인 패턴과 같은 설계 원칙이다.

프로그래머가 직접 객체의 생성과 소멸 객체간 관계같은 객체의 제어를 수행하는 것이 아니라,
여러 프레임워크, 컨테이너에서 제어를 수행하는 것

DI와 무슨 관계일까?

제어의 역전은 일반적인 디자인 패턴 중 하나이고 의존성 주입은 이런 제어의 반전 패턴을 달성하는 방법 중 하나임

스프링 웹 MVC 프레임워크