스프링 \_01

가로선

# 스프링 부트

* 단독으로 실행할 수 있음 웹 서버가 내장되어 있어 설치하지 않아도 된다.
* 빌드 구성을 쉽게 도와줌
* 스프링과 3rd party(외부) 라이브러리 자동 구성 (궁합 조합을 스스로 지정)
* 메트릭, 상태 확인, 외부 구성 같은 프로덕션 준비 기능 제공
* 설정이 편하고 간결
* 스프링 부트는 스프링 프레임 워크를 더 편하게 사용하게 도와주는 기술이다.

### 

### 스프링의 핵심

* 좋은 객체 지향 프로그래밍을 도와주는 프레임워크

### 다형성

* 실세상 예1 ) 운전자 역할 -> 자동차 역할 전부 운전가능 ( 자동차 종류[구현] 무관)
* 클라이언트에 영향을 주지 않고 새로운 기능을 추가 할 수 있다. (새로운 자동차)
* 실세상 예2) 공연에서 로미오 ,줄리엣 역할 , 배우 ( 구현 )
* 역할과 구현으로 구분하면 단순해지고, 유연해지며, 변경도 편해진다.
* 클라이언트를 변경하지 않고! 구현 기능을 구현하고 추가 할 수 있다.

의존한다 : 알고있다 ?

# SOLID 좋은 객체 지향 설계 5가지 원칙

## SRP( Single responsibility principle ) 단일 책임 원칙

한 클래스는 하나의 책임만 가져야 한다. 변경 했을 때 변경이 파급효과가 적어야한다.

## OCP( Open/closed principle ) 개방-폐쇄 원칙

소프트웨어 요소는 확장에는 열려 있으나 변경에는 닫혀 있어야 한다. 확장을 할때 다형성을 최대한으로 활용한다고 생각

-클라이언트에서 직접 구현시에는 OCP가 깨지게 된다. 스프링이 외부에서 주입을 통해 가능하다.

## LSP ( Liskov substitution principle) 리스코프 치환 원칙

객체는 프로그램의 정확성을 깨뜨리지 않으면서 하위 타입의 인스턴스를 바꿀수 있어야한다. 즉 다형성을 구현할때 인터페이스 규약을 지켜야한다.

## ISP ( Interface segregation principle ) 인터페이스 분리 원칙

특정 클라이언트를 위한 인터페이스 여러 개가 범용 한개보다 낫다.

인터페이스가 명확해짐, 대체 가능성이 높아짐

## DIP ( Dependency inversion principle ) 의존 관계 역전 원칙

프로그래머는 추상화에 의존해야 한다. 구체화에 의존하면 안된다.

역할(Role)에 의존해야 한다 ( 알아야 한다. )

## 

## 팁

<https://kim-oriental.tistory.com/7>

스프링 부트 프로젝트 Intellij Community 버전에서 생성하기

<https://start.spring.io/>