PRESENTATION

40주차 주제 Bean

~Spring Bean~

by mun

Spring Bean

Spring IoC 컨테이너에 등록되어 관리되고 있는 객체

IoC

Inversion of Control: 제어의 역전
객체에 대한 제어권이 역전됨
컴포넌트 의존관계 설정, 및 생명주기를 해결하기 위한 디자인 패턴

Spring 컨테이너

스프링 프레임워크에서 객체를 생성하고 관리하고
책임지고 의존성을 관리해주는 컨테이너
인스턴스 생명주기 관리를 개발자가 아닌 컨테이너가 대신 한다

등록된 객체를 Bean이라 부른다

Java Bean vs Spring Bean

Java Bean

자바 코드로 작성된 특정 클래스
전달 인자가 없는 생성자를 가지는 형태의 클래스

- getter / setter
- public의 no-argument 생성자
- 모든 필드는 private로 getter와 setter를 통해서만 접근 가능

Spring Bean

Spring IoC 컨테이너에 등록되어 관리되는 객체

개발자가 관리하는 객체가 아닌
스프링에게 제어권을 넘긴 객체

Spring 컨테이너에 등록하는 방법

1.수동으로 등록하기 @Bean

설정 클래스를 만들고 @Configuration 적용 후
내부에 스프링 빈으로 등록할 메서드 위에 @Bean 어노테이션을 적용

2.자동으로 등록하기 @Component

스프링에서 컴포넌트 스캔을 이용해
@Component가 붙은 모든 오브젝트를 빈으로 자동 등록

@Bean vs @Component

@Bean

메서드 위에 선언 가능

```java
@Target({ElementType.METHOD, ElementType.ANNOTATION_TYPE})
@Retention(RetentionPolicy.RUNTIME)
@Documented
public @interface Bean {
```

외부 라이브러리를 Bean으로 등록할 때 사용

외부 라이브러리는 ReadOnly File
= @Component 사용 불가능 (개발자가 컨트롤 불가능)
인스턴스를 생성하는 메서드를 만든 후
해당 메서드 위에 @Bean을 선언해 빈으로 등록한다

```java
@Bean
public PasswordEncoder BCryptPasswordEncoder() {
    return new BCryptPasswordEncoder(STRENGTH);
}
```

@Component

클래스, 인터페이스, Enum 등에 선언 가능

```java
@Target(ElementType.TYPE)
@Retention(RetentionPolicy.RUNTIME)
@Documented
@Indexed
public @interface Component {
```

개발자가 직접 컨트롤이 가능한 객체에 사용

개발자가 생성한 클래스 같이 직접 컨트롤할 수 있는
클래스에 선언해 등록한다

두 차이를 이해하고
적절하게 선언해 사용해 보아요!

감사합니다

참고

- https://dev-coco.tistory.com/80
- https://m42-orion.tistory.com/98
- https://jjingho.tistory.com/10
- https://jjingho.tistory.com/11
- https://youwjune.tistory.com/43#%EB%B-%--%EC%-D%--%--%EC%--%--%EB%-F%--%EC%-C%BC%EB%A-%-C%--%EB%--%B-%EB%A-%-D%ED%--%--%EB%-A%--%--%EB%B-%A-%EB%B-%--%---%--%--Bean%--%EC%-D%B-%EC%-A%A-%ED%--%--%EA%B-%B-