Bean

주차	40주차
스터디 일자	@2024/09/04

@Bean

☝ @Bean과 @Component의 차이 ??

> `Configuration + Bean` , `Component` 어노테이션이 사용되면 모두 스프링의 Ioc 컨테이너에 객체로 관리되며, 어플리케이션 전역적으로 의존성 주입에 활용될 수 있다.

`@Bean`과 `@Component` 어노테이션은 여러 차이가 있지만 제일 대표적인 것은 **선언 위치다.**

`Component`는 **클래스 레벨**에 선언되고 `Bean`은 **메소드 레벨**에 선언된다.

@Bean

```
public class Student {

    public Student() {
        System.out.println("hi");
    }
```

```
}
```

```
@Configuration
public class ApplicationConfig {

    @Bean
    public Student student() {
        return new Student();
    }

}
```

```
public class Main {
    public static void main(String[] args) {

        ApplicationContext context = new AnnotationConfigAppl
icationContext(ApplicationConfig.class);
        Student student = context.getBean("student", Student.
class);

    }
}
```

`AnnotationConfigApplicationContext` 객체 (Bean으로 지정된 객체들을 가지고 있는 context == 컨테이너)를 생성하고 매개변수로 `@Configuration` 어노테이션을 부여한 `ApplicationConfig` 클래스를 넘겨준다.

그 후 `getBean()`을 이용하여 등록한 빈(Student)을 사용할 수 있다.

→ 이렇게 `@Bean`은 `@Configuration` 과 함께 사용해야 하고 메서드 레벨에 사용해야 한다!

이번에는 아래 코드처럼 student 메서드의 리턴값을 없애보았다.

```
@Configuration
public class ApplicationConfig {

    @Bean
    public void student() {
        new Student();     // return -> X
    }

}
```

이러면 객체가 메서드 안에만 존재하여 스프링 컨테이너에 빈으로 등록되지 않는다고 한다!

⇒ `@Bean` 어노테이션은 **return 값이 있는** 메서드에 사용해야 빈으로 등록할 수 있다!

만약 아래처럼 String, int 와 같은 **기본 타입**이 리턴값이라면 ,,?

```
@Configuration
public class ApplicationConfig {

    @Bean
    public String student() {
        "hi";
    }

}
```

→ 기본 데이터 타입도 빈으로 등록이 된다!

이 빈을 의존성 주입 해주고 싶을 경우에는

```
@Component
public class Sample {

    private final String str;

    @Autowired// 의존성 주입을 위한 어노테이션
    public Sample(String str) {
        this.str = str; // 여기서 "hi"가 주입됨
    }

    public void print() {
        System.out.println(str); // "hi" 출력
    }

}
```

이렇게 사용하면 된다!

→ 이 때, 의존성 주입할 빈이 `hi`를 출력하는 `student()` 메서드인지 어디에도 명시하지 않았는데 의존성 주입이 된다.

🤔 **왜그런 것일까?**

의존성 주입 시 사용하는 `@Autowired` 는 필요한 의존 객체의 **"타입"**에 해당하는 빈을 찾아 주입하기 때문이다.

`student()` 메서드의 리턴 객체가 String 타입이기 때문에 위 생성자 String str 에 자동으로 주입되는 것이다.

🖐 잠깐!

만약에 빈으로 등록된 String 타입이 여러개면 어떡해? 원하는 빈이 뭔지 어떻게 알고 찾지?

→ 이럴 때는 **3가지 방법**이 있다.

여러개의 빈을 찾을 때 해결할 수 있는 3가지 방법

1. `@Autowired` **필드 명 매칭**
2. `@Primary`
3. `@Qualifier`

@Autowired 필드 명 매칭

`@Autowired` 는 우선적으로 타입(Type)으로 빈을 찾지만, 찾지 못하면 **필드 이름**으로 찾는 특징이 있다.

```
@Configuration
public class ApplicationConfig {

    @Bean
    public String student1() {
        return "hi1"; // 빈 이름: student1
    }

    @Bean
    public String student2() {
        return "hi2"; // 빈 이름: student2
    }

}
```

☝️ 이렇게 빈으로 등록을 해두었을 경우,

```
@Component
public class Sample {

    private final String str;

    @Autowired// 의존성 주입을 위한 어노테이션
    public Sample(String student1) {
        this.str = student1; // 여기서 "hi"가 주입됨
    }

    public void print() {
        System.out.println(str); // "hi" 출력
    }

}
```

String 타입 빈이 여러개여도, `student1()` 메서드를 통해 등록한 빈이 주입된다!

근데 이건 추천하는 방법이 아니라고 한다!

@Primary

`@Primary` 어노테이션을 붙여서 우선순위를 지정하는 방식

```
@Configuration
public class ApplicationConfig {

    @Bean
    public String student1() {
        return "hi1"; // 빈 이름: student1
    }
```

```
    @Primary
    @Bean
    public String student2() {
        return "hi2"; // 빈 이름: student2
    }

}
```

이런식으로 @Primary 어노테이션을 붙여주면 같은 타입이지만 student2가 반환하는 객체 빈이 더 우선순위가 높아진다!

```
@Component
public class Sample {

    private final String str;

    @Autowired
    public Sample(String str) {
        this.str = str; // // student2 가 주입됨
    }

    public void print() {
        System.out.println(str); // "hi" 출력
    }

}
```

▼ 궁금증 💭

근데, 여기서 궁금한 점

```
@Component
public class Sample {

    private final String str;

    @Autowired
    public Sample(String student1) {
        this.str = student1; // 이렇게 하면 어떤 빈이 주입될까
    }

    public void print() {
        System.out.println(str);
    }

}
```

`@Autowired` 가 필드네임으로 찾아서 student1이 선택될까,

`@Primary` 가 적용되어 student2()가 선택될까?

찾아봐도 나오질 않아서 아직도 궁금하다 ^.^ 아는 사람 있다면 알려주시길

@Qualifier

빈에 별명을 부여하는 방법

```
@Configuration
public class ApplicationConfig {

        @Qualifier("first")
    @Bean
    public String student1() {
        return "hi1"; // 빈 이름: student1
```

```
    }

        @Qualifier("second")
    @Bean
    public String student2() {
        return "hi2"; // 빈 이름: student2
    }

}
```

이렇게 별명을 부여해주면,

```
@Component
public class Sample {

    private final String str;

    @Autowired
    public Sample(@Qualifier("second") String str) {
        this.str = student;
    }

    public void print() {
        System.out.println(str);
    }

}
```

⇒ `student2()` 객체 빈이 선택됨.

이렇게 별명으로 빈을 찾아올 수 있다.

출처

https://galid1.tistory.com/494

https://velog.io/@neity16/Spring-핵심-원리-기본편-8-Primary-Qualifier

https://medium.com/sjk5766/bean과-component-차이-96a8d0533bfd

https://ducktopia.tistory.com/80

https://tech.kakaopay.com/post/martin-dev-honey-tip-2/