

초격차 패키지 Online.

# Boot의 기본기

## PART1 | Spring Boot Properties

이걸 안 쓸거면 boot 쓰지 마세요

## PART2 | @SpringBootApplication

스프링 부트의 시작점

## PART3 | @Component vs. @Configuration

둘의 차이가 뭐야?

## PART4 | Configuration Properties

프로퍼티를 설정하는 우아한 방법

# Boot의 기본기

## 3 @Component vs. @Configuration

## @Component vs. @Configuration

### 3. @Component vs. @Configuration

@Component?? @Configuration???

실무에 나가보면..

```
@EnableWebMvc
@Component
public class Config implements WebMvcConfigurer {

    @Override
    public void addFormatters(FormatterRegistry registry) {
        WebMvcConfigurer.super.addFormatters(registry);
    }
}
```

```
@Configuration
public class SortService {

    private final Sort<String> sort;

    public SortService(@Qualifier("bubbleSort") Sort<String> sort, CustomProperties customProperties) {
        System.out.println("프로퍼티: " + customProperties.getDuration());
        System.out.println("구현체: " + sort.getClass().getName());
        this.sort = sort;
    }

    public List<String> doSort(List<String> list) { return sort.sort(list); }
}
```

## @Component

### 3. @Component vs. @Configuration

"이 클래스는 커스텀 빈이다."

1. @ComponentScan -> base package 에서부터 모든 @Component 검색
2. 인스턴스화: 필요한 의존성을 모두 주입
3. 스프링 컨테이너에 등록: 필요한 곳에 주입

## 참고: 빈을 만드는 방법 2+1 가지

### 3. @Component vs. @Configuration

1. @Component
2. @Bean (in @Configuration)
3. @Bean (in @Component): Lite Mode

<https://docs.spring.io/spring-framework/docs/current/reference/html/core.html#beans-java-basic-concepts>

```
@Component
public class BubbleSort <T extends Comparable> {

    @Override
    public List<T> sort(List<T> list) {
        List<T> output = new ArrayList<>(list);

        for (int i = output.size() - 1; i > 0; i--) {
            for (int j = 0; j < i; j++) {
                if (output.get(j).compareTo(output.get(j+1)) > 0) {
                    T temp = output.get(j);
                    output.set(j, output.get(j+1));
                    output.set(j+1, temp);
                }
            }
        }

        return output;
    }
}
```

```
@Configuration
public class Config {

    @Bean
    public Sort<String> bubbleSort() {
        return new BubbleSort<>();
    }
}
```


```
@Component
public class Config {

    @Bean
    public Sort<String> bubbleSort() {
        return new BubbleSort<>();
    }
}
```

```
@Service
public class SortService {

    private final Sort<String> sort;

    public SortService(@Qualifier("bubbleSort") Sort<String> sort,
        System.out.println("프로퍼티: " + customProperties.getDuration());
        System.out.println("구현체: " + sort.getClass().getName());
        this.sort = sort;
    }
}
```



## 참고: @Component vs. @Bean

### 3. @Component vs. @Configuration

## @Component

- Class-level annotation
- 등록하려는 빈의 클래스 소스가 편집 가능한 경우 사용
- auto-detection 에 걸림

## @Bean

- method-level annotation
- 좀 더 읽기 쉬움
- 인스턴스화 하는 코드가 수동으로 작성됨
- 빈의 인스턴스 코드와 클래스 정의가 분리된 구조
- 따라서 외부 라이브러리, 써드 파티 클래스도 빈으로 등록 가능

## @Component: Stereotype Annotations

### 3. @Component vs. @Configuration

컴포넌트에 해당하는 스테레오 타입 애노테이션들

- @Controller
- @Service
- @Repository

## @Component: 결론

"컴포넌트는 빈이다."

**3.**  
@Component vs.  
@Configuration



## @Configuration

### 3. @Component vs. @Configuration

**"이 클래스는 각종 빈 설정을 담고 있다."**

1. @SpringBootApplication 이 컴포넌트 스캔을 통해 @Configuration 을 찾아냄
2. 안의 빈 설정(메소드)을 읽어서 스프링 컨테이너에 등록
3. 필요한 곳에 주입
4. 또는 각종 스프링 인터페이스의 구현에 함께 활용

## @Configuration: 스프링 설정 인터페이스의 구현

### 3. @Component vs. @Configuration

```
@Configuration
@EnableWebSecurity
public class WebSecurityConfig extends
    WebSecurityConfigurerAdapter {

    @Override
    protected void configure(HttpSecurity http) {
        http
            // ...
            .headers(headers -> headers
```

```
@EnableWebMvc
@Configuration
public class Config implements WebMvcConfigurer {

    @Override
    public void addFormatters(FormatterRegistry registry) {
        WebMvcConfigurer.super.addFormatters(registry);
    }
}
```

## 결론

### 3. @Component vs. @Configuration

## "애노테이션이 의도에 맞게 사용되었는지 잘 봐주자"

- 빈 설정은 @Configuration, 클래스 빈 등록은 @Component
- 정확한 목적을 모르고 쓰면 "잘 모르겠는데 어쨌든 돌아가요" 운영의 시한폭탄
- IDE 가 노란 줄 그어주면
  - 응 돌아는 간다는 거지 (X)
  - 내 사전에 노란 줄이란 있을 수 없다 (O)