스프링부트로 **RestFulAPI** 구현하기

3장 SPRING

컴포넌트스캔

- **@ComponentScan**
- **@Autowired**
- **@RequiredArgsConstructor**

박명회

3장 @ComponentScan, @Autowired, @RequiredArgsConstructor

여기 **Spring Boot**에서 **@ComponentScan**, **@Autowired**, 및 **@RequiredArgsConstructor**를 사용한 의존성 주입을 설명하는 그림이 있습니다.

- **AppConfig** 클래스는 **@ComponentScan** 어노테이션을 사용하여 패키지를 스캔합니다.
- **Engine** 클래스는 **@Component**로 등록되어 빈으로 관리됩니다.
- **Car** 클래스는 **@RequiredArgsConstructor** 어노테이션을 통해 생성자 주입으로 **Engine** 객체를 주입받습니다.
- 화살표가 **Engine** 객체가 **Car** 객체로 주입되는 과정을 나타냅니다.

2장 @Configuration과 @Bean

Engine.class

```
@Component
public class Engine {
    private final String type = "V8";

    public String getType() {
        return type;
    }
}
```

Car.class

```
@Component
@RequiredArgsConstructor
public class Car {
    private final Engine engine;

    public void printEngineType() {
        System.out.println("Engine type: "
+ engine.getType());
    }
}
```

2장 @Configuration과 @Bean

@Configuration을 사용하여 **Car**와 **Engine** 객체를 조립하는 설정 클래스를 작성합니다.

```
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;

@Configuration
@ComponentScan(basePackages = "com.pmh")
public class AppConfig {
}
```

2장 @Configuration과 @Bean

이제 **AppConfig**를 사용하여 **Spring** 컨테이너를 설정하고 **Car** 빈을 가져오는 예제입니다.

```
import org.springframework.context.ApplicationContext;
import
org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Main {
    public static void main(String[] args) {
        // Spring 컨테이너 생성
        ApplicationContext context =
                new AnnotationConfigApplicationContext(AppConfig.class);

        // Car 빈 가져오기
        Car car = context.getBean(Car.class);
        System.out.println("Engine Type: " + car.getEngine().getType());
    }
}
```

감사합니다