스프링핵심원리-컴포넌트 스캔

2021년 8월 10일 화요일 오후 6:17

스프링 빈을 직접 등록하는 것이 많아지면서, 등록하는 과정에서의 데이터 누수, 노동력의 낭비를 해결하기 위해 스프링이 자동으로 스프링 빈을 등록하는 컴포넌트 스캔이라는 기능 제공
+ 의존관계도 자동으로 주입하는 @Autowired 라는 기능도 제공

스프링 빈의 기본이름은 클래스명을 사용, 맨 앞글자만 소문자 사용
직접 지정할수도 있음 ex) @Component('빈 이름")

@Autowired 를 지정하면 스프링 컨테이너가 자동으로 해당 스프링 빈을 찾아서 주입한다
이때 기본 조회 전략은 타입이 같은 빈을 찾아서 주입한다. (먼저 타입으로 찾아봄, 근데같은 타입이 여러 개 있다면 충돌남=> 어떻게 처리??)
getBean(MemberRepository.class) 와 동일하다고 보면 됨

컴포넌트 스캔의 탐색 위치와 범위를 지정할 수 있음
ex) basePackages = "hello.core.member",
member 패키지부터 찾음
ex2) basePackageClasses : 지정된 클래스의 패키지를 탐색 시작 위로 지정한다
지정안하면 @ComponentScan 붙은 설정 정보 클래스의 패키지가 시작 위치가 된다

추천 : 패키지 위치 지정하지말고 설정정보를 담은 클래스를 최상단에 배치

컴포넌트 스캔 기본 대상
@Component : 컴포넌트 스캔에서 사용
//아래 것들은 저 어노테이션들이 @Conponent 가지고 있음
@Controller : 스프링 MVC 컨트롤러에서 사용
@Service : 스프링 비즈니스 로직에서 사용
@Repository : 스프링 데이터 접근 계층에서 사용
@Configuration : 스프링 설정 정보에서 사용

필터
includeFilters : 컴포넌트 스캔 대상을 추가로 지정
excludeFilters : 컴포넌트 스캔에서 제외할 대상을 지정

옵션:

```
@ComponentScan(
        includeFilters = @Filter(type = FilterType.ANNOTATION, classes = MyIncludeComponent.class),
        excludeFilters = @Filter(type = FilterType.ANNOTATION, classes = MyExcludeComponent.class)
)
```

ex) FilterType.ANNOTATION

ANNOTATION : 기본값, 어노테이션을 인식해서 동작함 (기본값이라 안써도됨)
ASSIGNABLE_TYPE : 지정한 타입과 자식 타입을 인식해서 동작함
ASPECTJ : AspectJ 패턴 사용
REGEX : 정규 표현식
CUSTOM : TypeFilter 라는 인터페이스 구현해서 처리

중복등록과 충돌

1. 자동 빈 등록 vs 자동 빈 등록

컴포넌트 스캔에 의해 자동으로 스프링 빈이 등록되는데, 이름이 같은 경우 스프링은 오류 발생시킴
= ConflictingBeanDefinitionException 예외 발생

2. 수동 빈 등록 vs 자동 빈 등록

이러한 경우 수동 빈 등록이 우선권을 가지고 수동 빈이 자동 빈을 오버라이딩 해버림
=> 의도치 않은 경우, 잡기 힘든 버그 발생
=> 스프링 부트의 경우는 충돌나면 에러발생하도록 기본 값을 변경함