6️⃣

Section6. 컴포넌트 스캔

컴포넌트 스캔과 의존관계 자동 주입 시작하기

스프링은 설정 정보가 없어도 자동으로 스프링 빈을 등록하는 컴포넌트 스캔이라는 기능을 제공

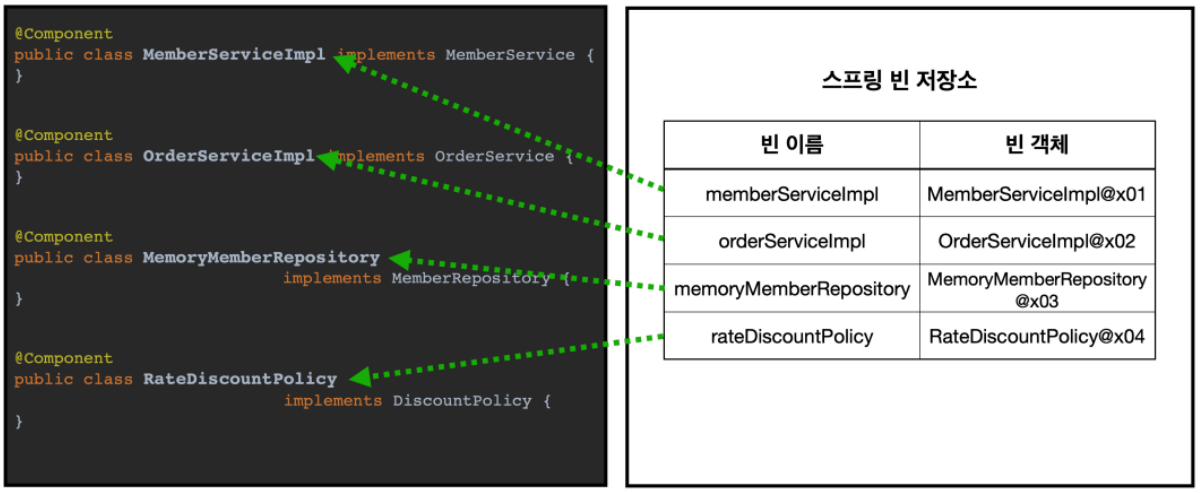
- `@ComponentScan` 어노테이션 사용
  - `@Component` 어노테이션이 붙은 클래스들을 자동으로 스프링 빈으로 등록 시켜준다.

```
@ComponentScan(
        // 컴포넌트 스캔 대상 제외
        excludeFilters = @ComponentScan.Filter(type = FilterT
)
```

- 이전에 `AppConfig`에서는 `@Bean` 으로 직접 설정 정보를 작성했고, 의존관계도 직접 명시했다.
- 설정 정보 자체가 없기 때문에, 의존관계 주입도 해당 클래스 안에서 해결해야 한다.
  `@Autowired` 는 의존관계를 자동으로 주입해준다.

```
@Autowired // ac.getBean(MemberRepository.class)
public MemberServiceImpl(MemberRepository memberRepository
    this.memberRepository = memberRepository;
}
```

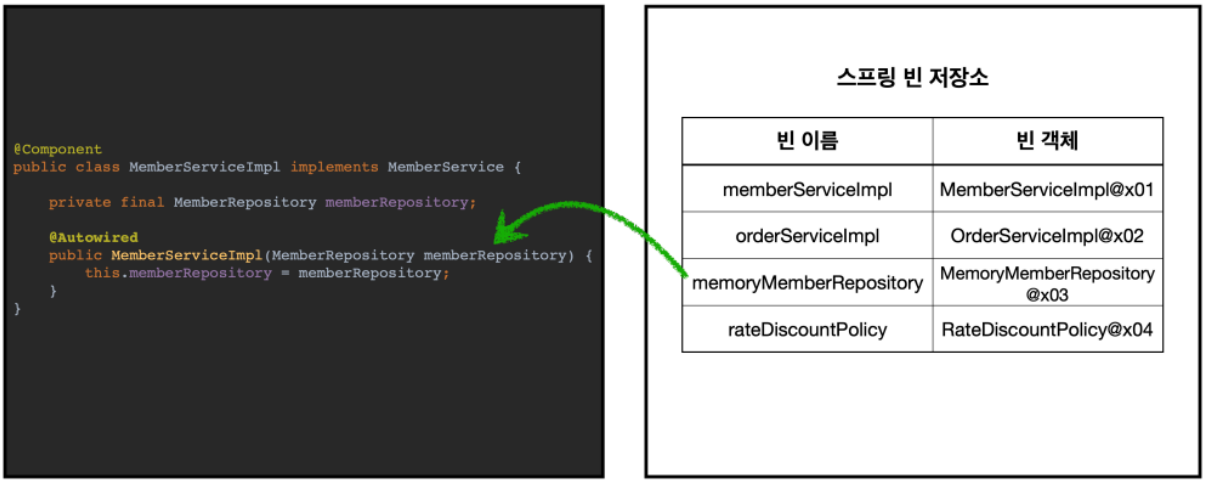
@Component 컴포넌트 스캔



- `@ComponentScan` 은 `@Component` 가 붙은 모든 클래스를 스프링 빈으로 등록한다.
- 이때 스프링 빈의 기본 이름은 클래스명을 사용하되 맨 앞글자만 소문자를 사용한다.
- **빈 이름 기본 전략**: `MemberServiceImpl` 클래스 `memberServiceImpl`
- **빈 이름 직접 지정**: 만약 스프링 빈의 이름을 직접 지정하고 싶으면

  `@Component("memberService2")` 이런식으로 이름을 부여하면 된다.

@Autowired 의존관계 자동 주입



- 생성자에 `@Autowired` 를 지정하면, 스프링 컨테이너가 자동으로 해당 스프링 빈을 찾아서 주입한다.
- 이때 기본 조회 전략은 타입이 같은 빈을 찾아서 주입한다.
- `getBean(MemberRepository.class)` 와 동일하다고 이해하면 된다.

탐색 위치와 기본 스캔 대상

탐색 위치

- 아래와 같이 탐색할 위치를 지정해줄 수 있다.

```
@Component(basePackages = "hello.core.member")
@ComponentScan(
        // 컴포넌트 스캔 대상 제외
        excludeFilters = @ComponentScan.Filter(type = FilterT
                // 지정 패키지 탐색
        basePackages = "hello.core.member",
        // 해당 클래스의 패키지 탐색
        basePackageClasses = AutoAppConfig.class
)
```

- 패키지 탐색 위치의 default 값은 `Config` class가 포함된 패키지 전체이다.
- 설정 파일은 프로젝트의 최상단 위치에 두는 것이 좋다.
  - `main` 클래스를 보면 `@SpringBootApplication` 어노테이션이 붙어있는데 이 또한 `@ComponentScan` 이 포함되어있다.

기본 스캔 대상

- `@Component` : 컴포넌트 스캔에서 사용
- `@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 지정한다.
- `'excludeFilters'` : 컴포넌트 스캔에서 제외할 대상을 지정한다.

> 💡 실습 코드 작성

필터 타입

- default 값은 `FilterType.ANNOATION` 이다.
- `ANNOTATION`: 기본값, 애노테이션을 인식해서 동작한다.
  ex)
  `org.example.SomeAnnotation`
- `ASSIGNABLE_TYPE`: 지정한 타입과 자식 타입을 인식해서 동작한다.
  ex)
  `org.example.SomeClass`
- `ASPECTJ`: AspectJ 패턴 사용
  ex)
  `org.example..Service+`
- `REGEX`: 정규 표현식
  ex)
  `org\.example\.Default.`
- `CUSTOM`: `TypeFilter` 라는 인터페이스를 구현해서 처리
  ex)
  `org.example.MyTypeFilter`

중복 등록과 충돌

컴포넌트 스캔에서 같은 빈 이름이 있는 상황

자동 빈 등록 vs 자동 빈 등록

- 컴포넌트 스캔에 의해 자동으로 스프링 빈이 등록되는데, 그 이름이 같은 경우 스프링은 오류를 발생시킨다.
  - ConflictingBeanDefinitionException 예외 발생

수동 빈 등록 vs 자동 빈 등록

이 경우에는 스프링이 오류를 안 발생시키는데 로그를 출력해보면

자동으로 등록된 빈 → 수동으로 등록된 빈이 **오버라이딩** 하여 충돌이 일어나지 않는다.

- **수동 등록 빈의 우선순위가 더 높다.**

> 💡 최근 스프링부트에서는 자동 vs 수동 도 오류를 발생 시키게 기본 값을 바꾸었다.