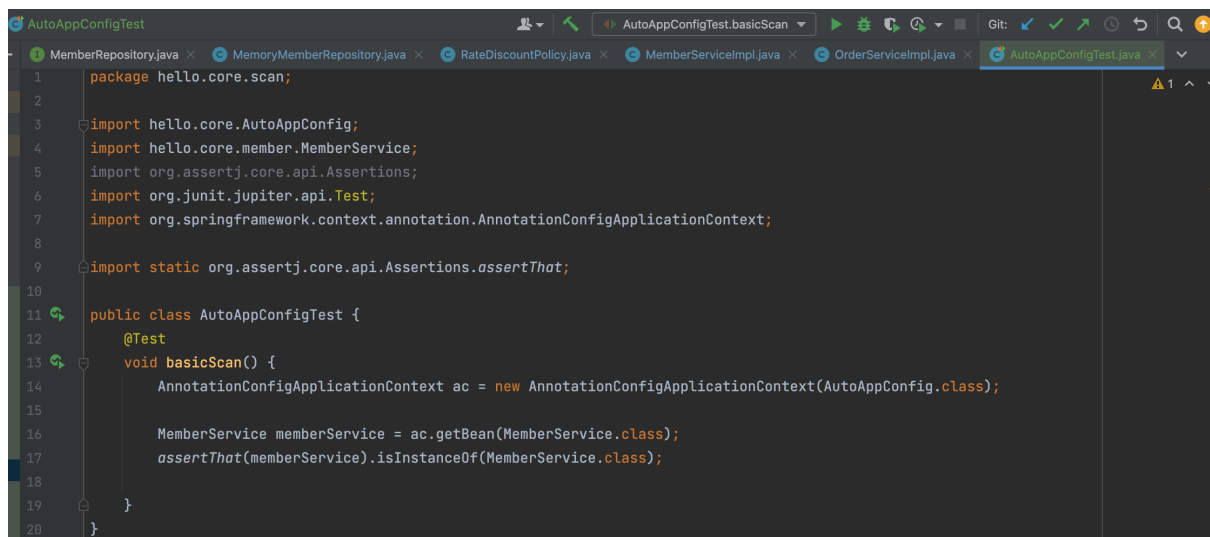


Chapter 7. 컴포넌트 스캔

컴포넌트 스캔과 의존관계 자동 주입 시작하기

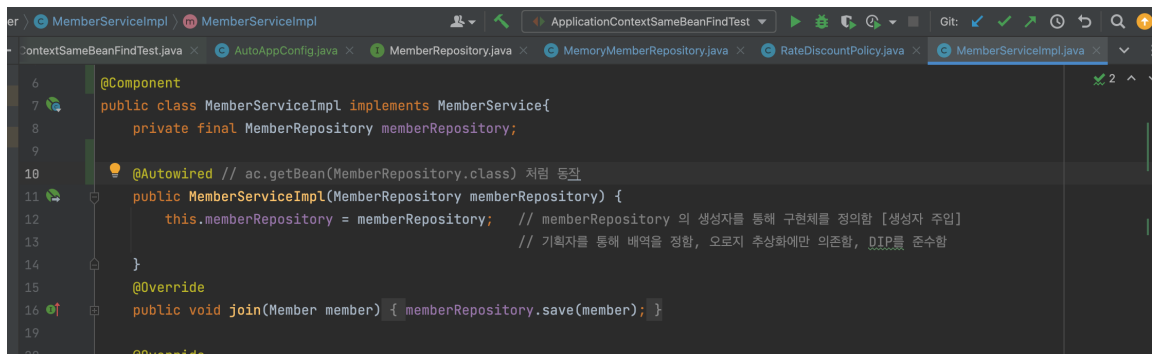
- 지금까지 스프링 빈을 등록할 때에는 자바 코드의 @Bean이나 XML의 <bean>을 통해서 설정 정보에 직접 등록할 스프링 빈을 나열했다.
- 스프링 빈이 수백개가 되면 곤란해짐
- 그래서 스프링을 설정 정보 없이도 자동으로 bean을 등록하는 “컴포넌트 스캔” 이라는 기능을 제공한다.
- 또 의존관계도 자동으로 주입하는 “@Autowire” 기능도 제공

@컴포넌트 스캔을 사용하려면 먼저 '@ComponentScan'을 설정 정보에 붙여주면 된다.



```
1 package hello.core.scan;
2
3 import hello.core.AutoAppConfig;
4 import hello.core.member.MemberService;
5 import org.assertj.core.api.Assertions;
6 import org.junit.jupiter.api.Test;
7 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
8
9 import static org.assertj.core.api.Assertions.assertThat;
10
11 public class AutoAppConfigTest {
12     @Test
13     void basicScan() {
14         AnnotationConfigApplicationContext ac = new AnnotationConfigApplicationContext(AutoAppConfig.class);
15
16         MemberService memberService = ac.getBean(MemberService.class);
17         assertThat(memberService).isInstanceOf(MemberService.class);
18     }
19 }
20 }
```

- Configuration 으로 등록한 애들은 빨거야 !! 인식하지마
- 기존에 AppConfig와는 다르게 @Bean으로 등록한 클래스가 하나도 없다.
- 이제 @Component를 붙이면 component가 붙은 애들만 불러옴
- MemberRepository는 어떻게 가져올 것인가? → @Autowired



```
6 @Component
7 public class MemberServiceImpl implements MemberService {
8     private final MemberRepository memberRepository;
9
10    @Autowired // ac.getBean(MemberRepository.class) 처럼 동작
11    public MemberServiceImpl(MemberRepository memberRepository) {
12        this.memberRepository = memberRepository; // memberRepository의 생성자를 통해 구현체를 정의함 [생성자 주입]
13        // 기획자를 통해 배역을 정함, 오로지 추상화에만 의존함, DIP를 준수함
14    }
15    @Override
16    public void join(Member member) { memberRepository.save(member); }
17
18    @Override
```

탐색 위치와 기본 스캔 대상

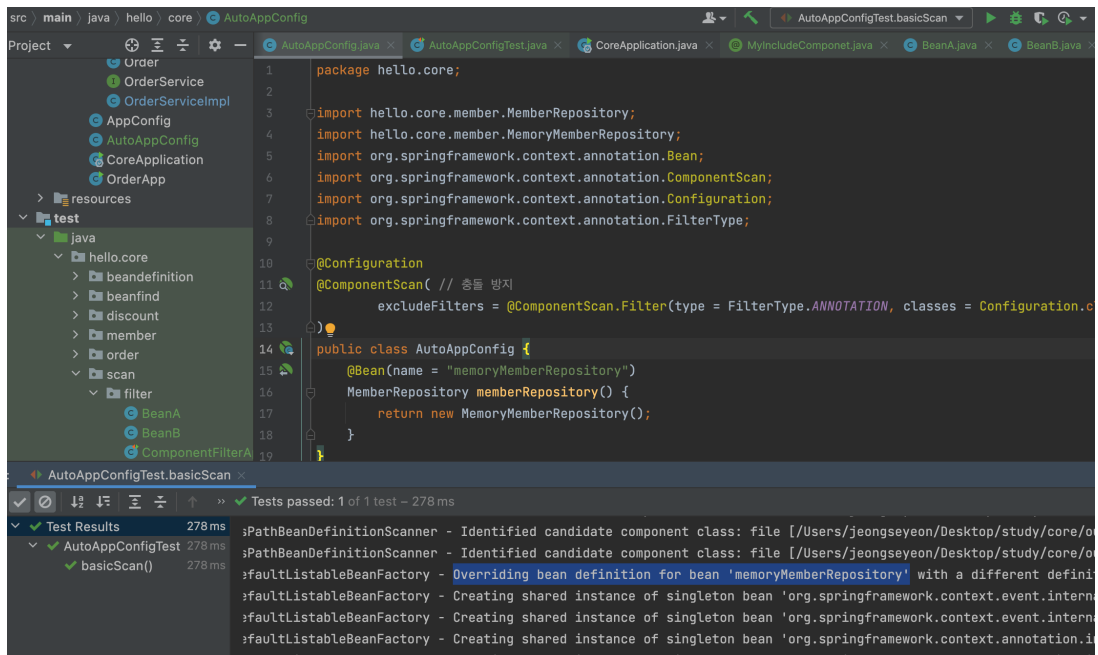
- 꼭 필요한 위치부터 탐색하도록 시작 위치를 지정할 수 있다.
- 권장하는 방법
 - 설정정보를 프로젝트의 최상단에 두는 것
- 기본 스캔 대상
 - “@Component” : 컴포넌트 스캔에서 사용
 - “@Controller” : 스프링 MVC 컨트롤러에서 사용
 - “@Service” : 스프링 비즈니스 로직에서 사용
 - @Repository : 스프링 데이터 접근 계층에서 사용
 - @Configuration : 스프링 설정 정보에서 사용
- 부가 기능
 - @Controller : 스프링 MVC 컨트롤러로 인식
 - @Repository : 스프링 데이터 접근 계층으로 인식하고, 데이터 계층의 예외를 스프링 예외로 변환해준다.
 - @Configuration : 스프링 설정정보로 인식하고, 스프링 빈이 싱글톤을 유지하도록 추가 처리를 한다.
 - @Service : 스프링에서 특별한 처리는 x, 개발자들이 핵심 비즈니스 로직이 여기구나! 하게끔 비즈니스 계층을 인식하는데 도움을 줌

필터

- 'includeFilters'에 MyIncludeComponet 를 추가해 BeanA가 스프링 빈에 등록 됨
- 'excludeFilters'에 MyexcludeComponent를 추가해 BeanB가 스프링 빈에 제외

중복등록과 충돌

- 자동 빈 등록 vs 자동 빈 등록
 - 컴포넌트 스캔에 의해 자동으로 스프링 빈이 등록되는데, 이름이 같은 경우 스프링은 오류를 발생시킴
 - `ConflictingBeanDefinitionException`
- 수동 빈 등록 vs 자동 빈 등록
 - 수동 빈 등록과 자동 빈 등록에서 빈 이름이 충돌되면?



- Overriding bean definition으로 <수동 빈>이 우선권을 가짐
- 스프링부트에서는 에러를 띄워줌 !! 개발자의 실수에 의한 수동 빈과 자동 빈의 충돌일 수도 있기 때문에