

애너테이션을 사용하여 빈 주입하기

◎ 미션	[백엔드] 스프링 핵심 원리 - 기본
※ 상태	완료
≡ 설명	Spring 프레임워크에서 애너테이션(@Autowired, @Inject 등)을 사용하여 빈을 주입합니다. 애너테이션을 이용하여 빈 의존성을 설정하고, 해당 방식으로 빈을 주입하는 코드를 작성합니다. 애너테이션으로 주입된 빈이 올바르게 동작하는지 확인하기 위해 실행 결과를 확인합니다. 결과물로 애너테이션으로 주입된 빈의 실행 결과 스크린샷을 결과물로 제출합니다.

@ComponentScan 애너테이션으로 빈을 자동 등록하게 되면, 의존관계를 명시할 공간이 없게 된다. 따라서 @Autowired 애너테이션으로 자동으로 의존관계 주입을 하도록 한다.

이전에는 @Bean 애너테이션을 이용해 수동으로 의존관계를 주입하였다. 이러한 방식은 하나하나 빈을 주입해주어야 하기 때문에 일일이 등록하기도 힘들고 귀찮다.

```
@Configuration  
public class AppConfig {  
  
    @Bean  
    public MemberService memberService() {  
        return new MemberServiceImpl(memoryMemberRepository());  
    }  
  
    @Bean  
    public OrderService orderService() {  
        return new OrderServiceImpl(  
            memoryMemberRepository(),  
            discountPolicy()  
        );  
    }  
  
    @Bean
```

```
public MemoryMemberRepository memoryMemberRepository() {  
    return new MemoryMemberRepository();  
}  
  
@Bean  
public DiscountPolicy discountPolicy() {  
    // return new FixDiscountPolicy();  
    return new RateDiscountPolicy();  
}  
}
```

기존의 `AppConfig.java`에서는 각 생성자마다 빈으로 등록해주어서 사용하였다.

```
@Configuration  
@ComponentScan(  
    excludeFilters = {@ComponentScan.Filter(type= FilterType.ANNOTATIO  
N, classes=Configuration.class)}  
)  
public class Auto AppConfig {  
  
}
```

새로운 `Auto AppConfig.java`를 통해 `@ComponentScan`으로 빈을 자동 주입할 것이다.

컴포넌트 스캔 및 의존관계 자동 주입

```
@Component  
public class MemberServiceImpl implements MemberService {  
  
    private final MemberRepository memberRepository;  
  
    @Autowired  
    public MemberServiceImpl(MemberRepository memberRepository) {  
        this.memberRepository = memberRepository;  
    }
```

```
}
```

```
@Component
public class OrderServiceImpl implements OrderService {

    private final MemberRepository memberRepository;
    private final DiscountPolicy discountPolicy;

    @Autowired
    public OrderServiceImpl(MemberRepository memberRepository, DiscountPolicy discountPolicy) {
        this.memberRepository = memberRepository;
        this.discountPolicy = discountPolicy;
    }
}
```

```
@Component
public class MemoryMemberRepository implements MemberRepository {

    private static Map<Long, Member> store = new HashMap<>();
}
```

```
@Component
public class RateDiscountPolicy implements DiscountPolicy {

    private int discountRate = 10;
}
```

각 클래스가 컴포넌트 스캔의 대상이 되도록 `Component` 애너테이션을 붙여준다.

또한 의존관계도 자동으로 주입될 수 있도록 의존관계가 필요한 경우 `@Autowired` 애너테이션도 함께 명시해준다.

테스트코드

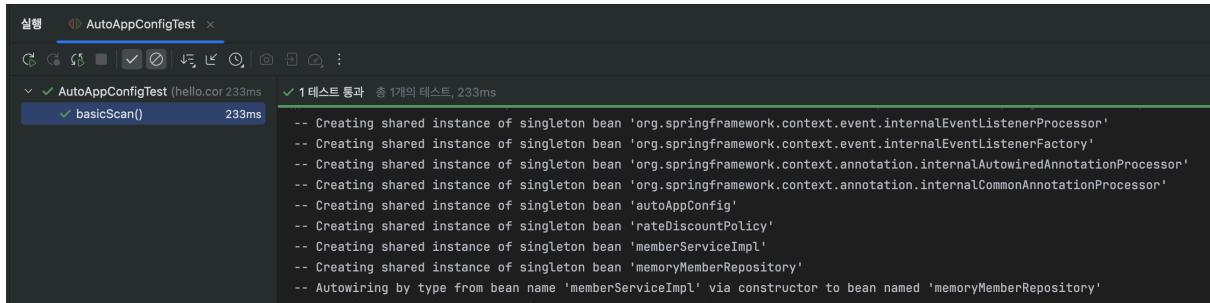
```

public class AutoAppConfigTest {

    @Test
    void basicScan() {
        AnnotationConfigApplicationContext ac = new AnnotationConfigApplicationContext(AutoAppConfig.class);

        MemberService memberService = ac.getBean(MemberService.class);
        Assertions.assertThat(memberService).isNotNull();
    }
}

```



싱글톤 빈이 잘 생성되어 주입된 것을 확인할 수 있다!