스프링시작하기

스프링 2강

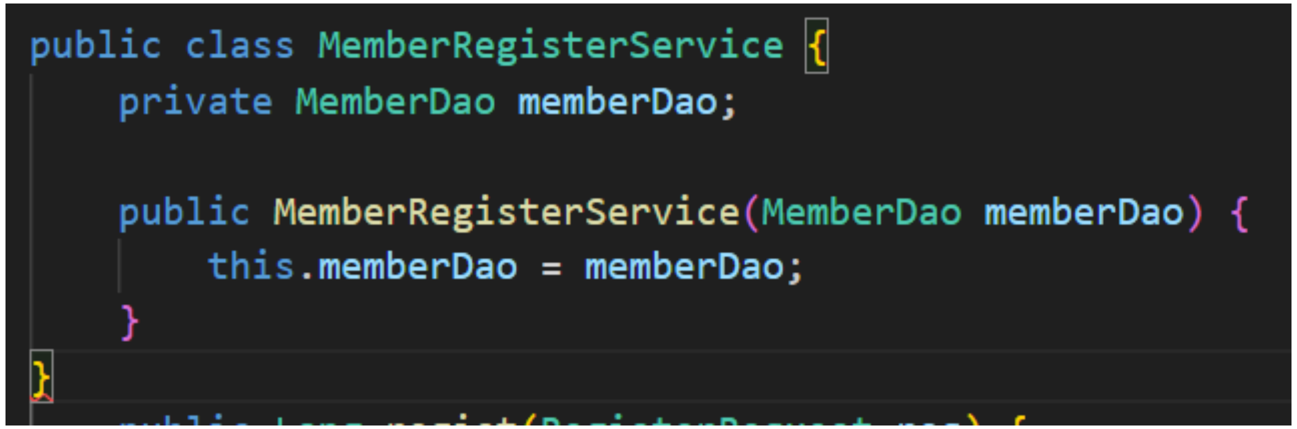
스프링DI

1) 객체 의존과 의존주입
2) 객체 조립
3) 스프링 DI설정

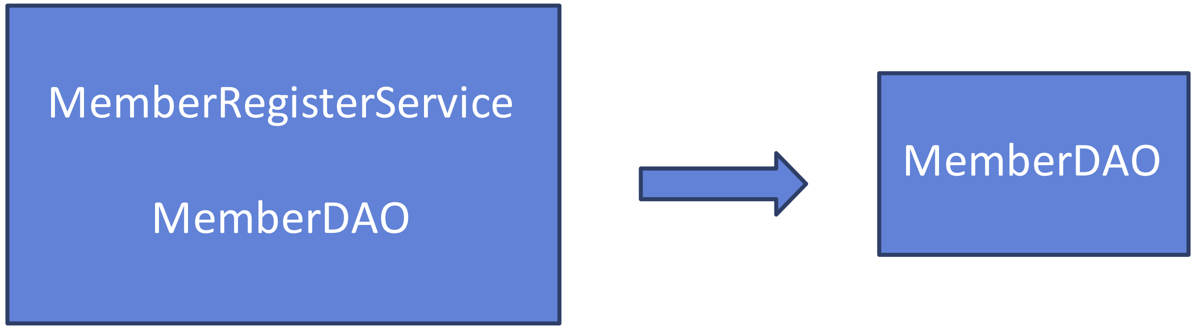
박명회

1.1 의존이란?

1) DI는 Dependency Injection의 약자로 우리말로는 의존주입이라고 번역한다.

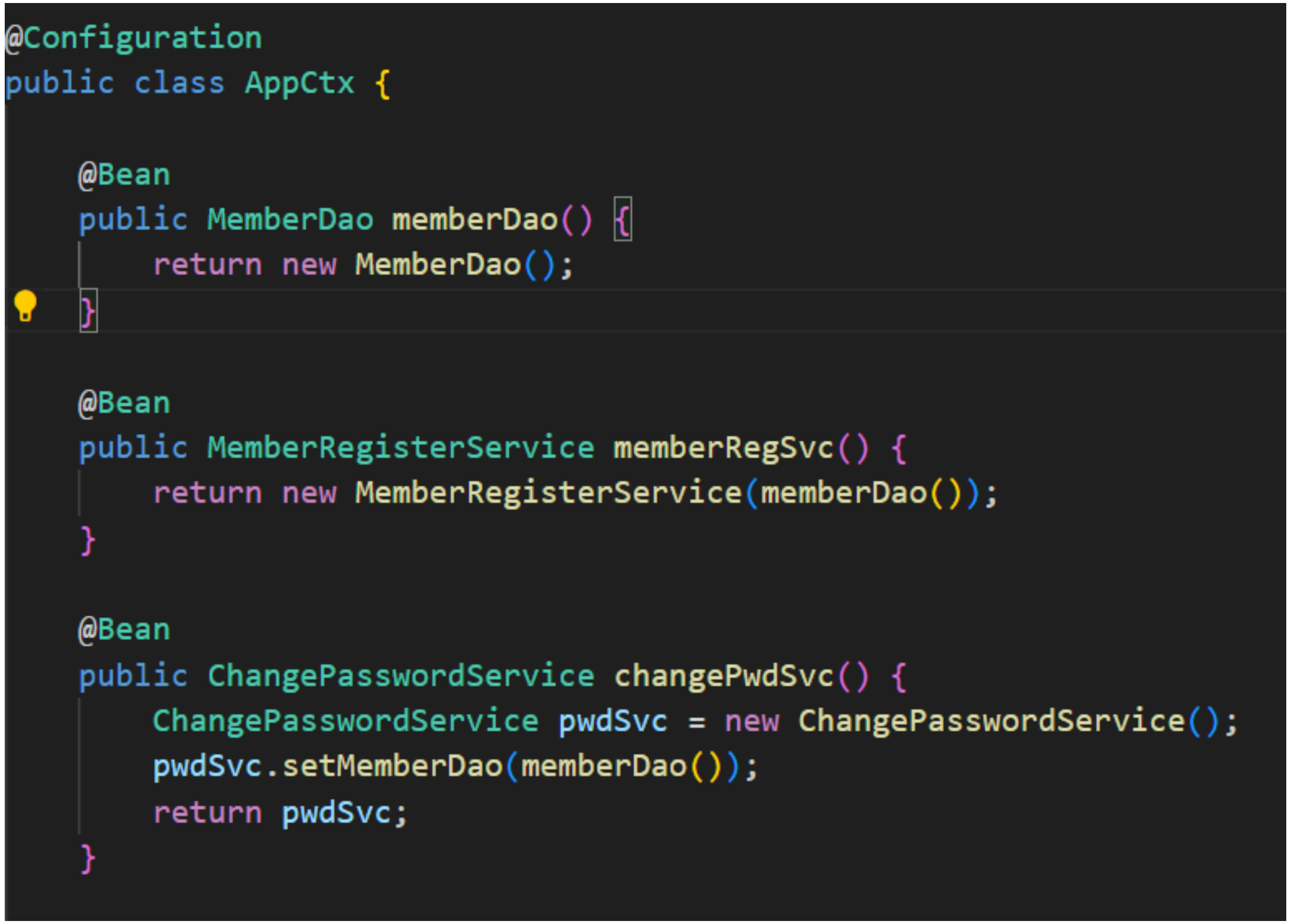


```
public class MemberRegisterService {
    private MemberDao memberDao;

    public MemberRegisterService(MemberDao memberDao) {
        this.memberDao = memberDao;
    }
}
```



1.2 스프링의 DI설정

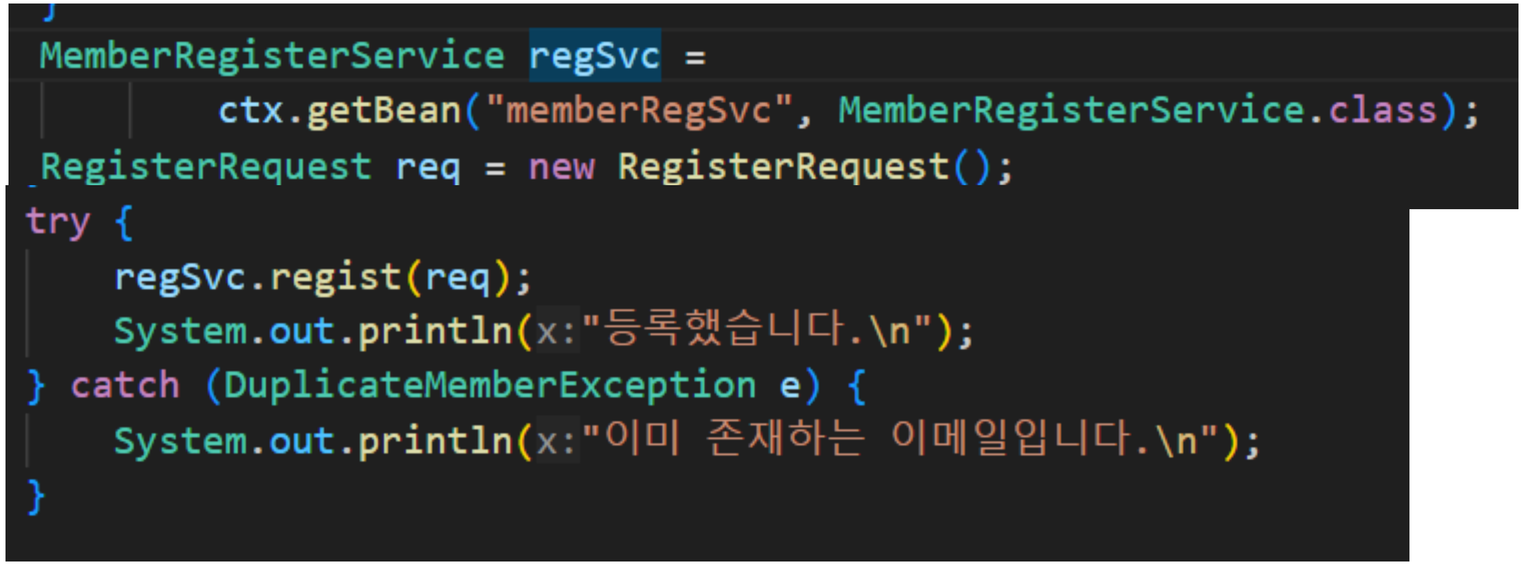
AppCtx.java



```
@Configuration
public class AppCtx {

    @Bean
    public MemberDao memberDao() {
        return new MemberDao();
    }

    @Bean
    public MemberRegisterService memberRegSvc() {
        return new MemberRegisterService(memberDao());
    }

    @Bean
    public ChangePasswordService changePwdSvc() {
        ChangePasswordService pwdSvc = new ChangePasswordService();
        pwdSvc.setMemberDao(memberDao());
        return pwdSvc;
    }
```

1.2 스프링의 DI설정

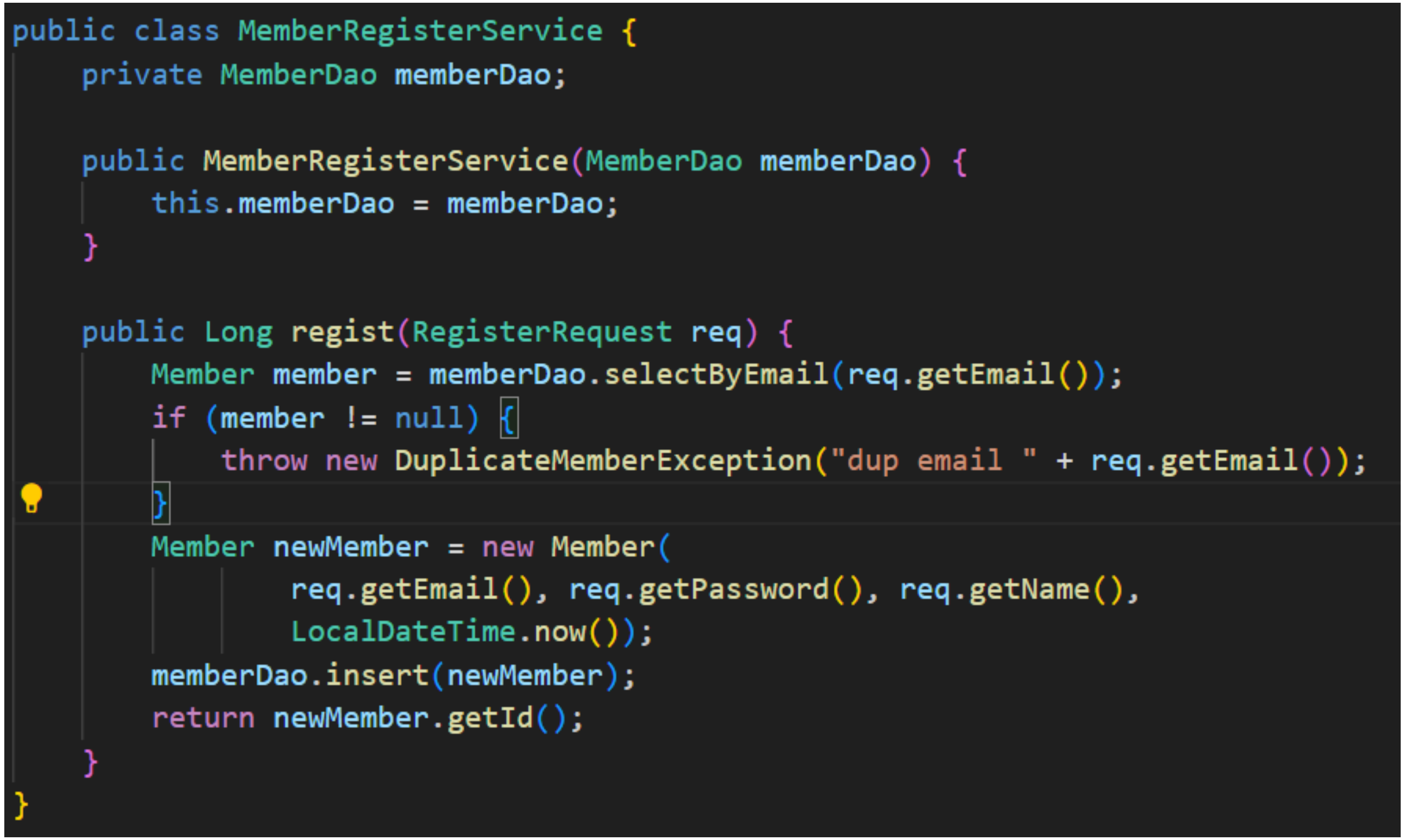
MainForSpring.java



```
MemberRegisterService regSvc =
        ctx.getBean("memberRegSvc", MemberRegisterService.class);
RegisterRequest req = new RegisterRequest();
try {
    regSvc.regist(req);
    System.out.println(x:"등록했습니다.\n");
} catch (DuplicateMemberException e) {
    System.out.println(x:"이미 존재하는 이메일입니다.\n");
}
```

1.2 스프링의 DI설정

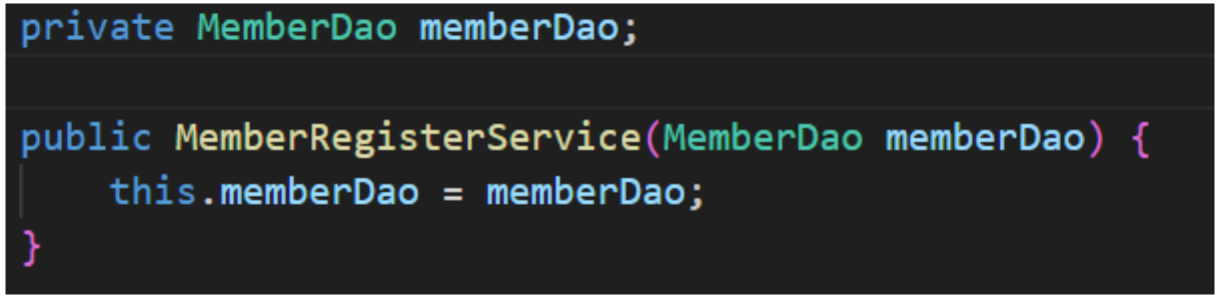
MemberRegisterService.java



```java
public class MemberRegisterService {
    private MemberDao memberDao;

    public MemberRegisterService(MemberDao memberDao) {
        this.memberDao = memberDao;
    }

    public Long regist(RegisterRequest req) {
        Member member = memberDao.selectByEmail(req.getEmail());
        if (member != null) {
            throw new DuplicateMemberException("dup email " + req.getEmail());
        }
        Member newMember = new Member(
                req.getEmail(), req.getPassword(), req.getName(),
                LocalDateTime.now());
        memberDao.insert(newMember);
        return newMember.getId();
    }
}
```

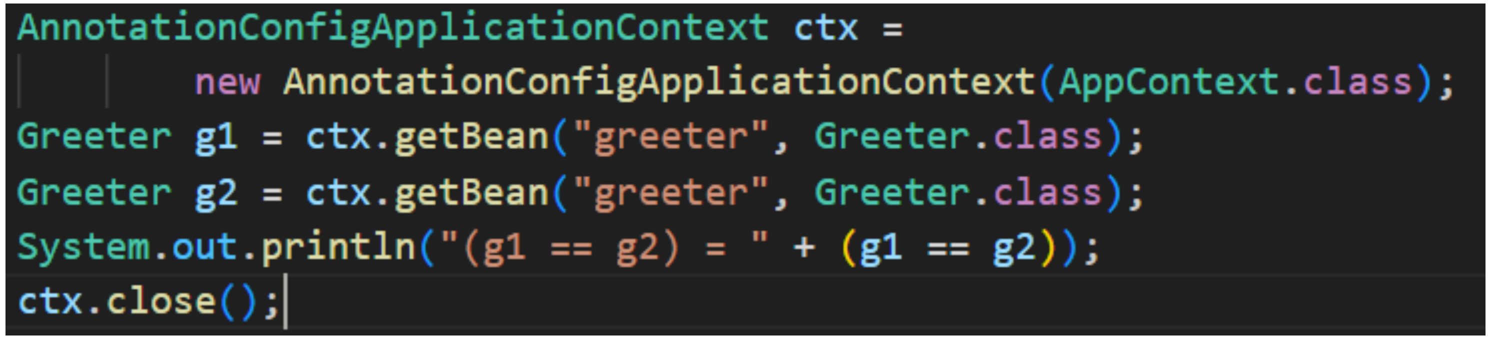
1.4 생성자 방식 주입

앞서 작성한 MemberRegisterService 클래스를 보면 아래 코드처럼 생성자를 통해 의존 주입받아 필드 This.



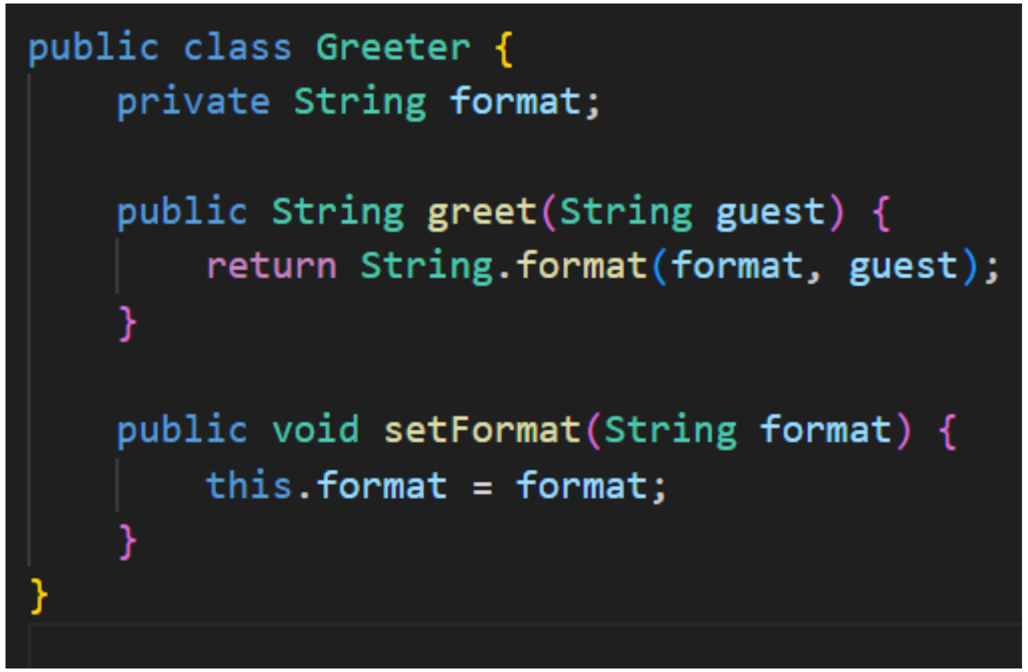
```java
private MemberDao memberDao;

public MemberRegisterService(MemberDao memberDao) {
    this.memberDao = memberDao;
}
```

1.4 실습

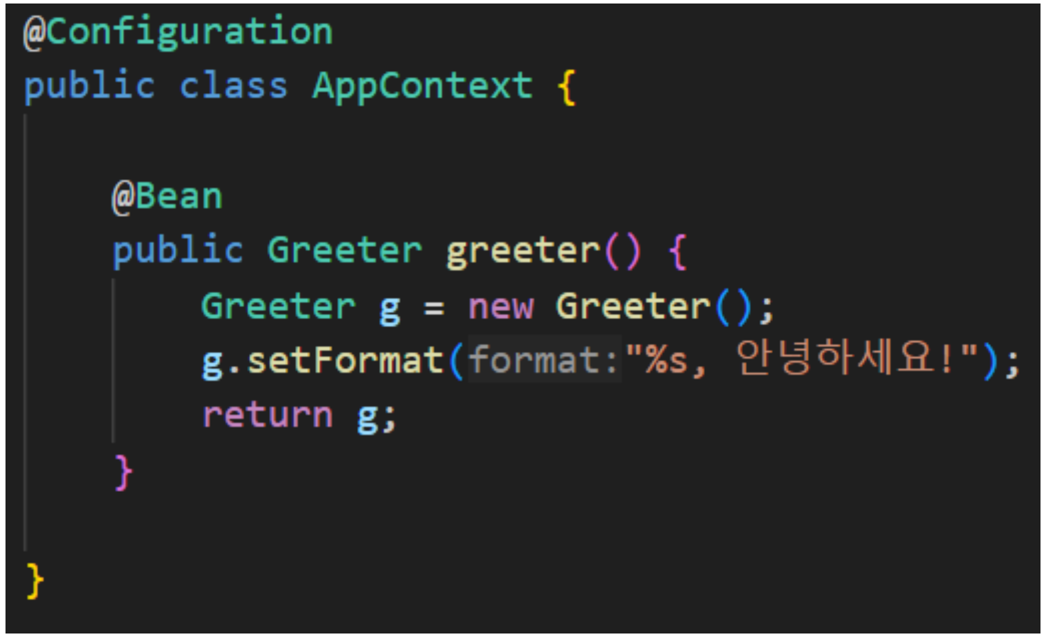


```java
AnnotationConfigApplicationContext ctx =
        new AnnotationConfigApplicationContext(AppContext.class);
Greeter g1 = ctx.getBean("greeter", Greeter.class);
Greeter g2 = ctx.getBean("greeter", Greeter.class);
System.out.println("(g1 == g2) = " + (g1 == g2));
ctx.close();
```

어노테이션

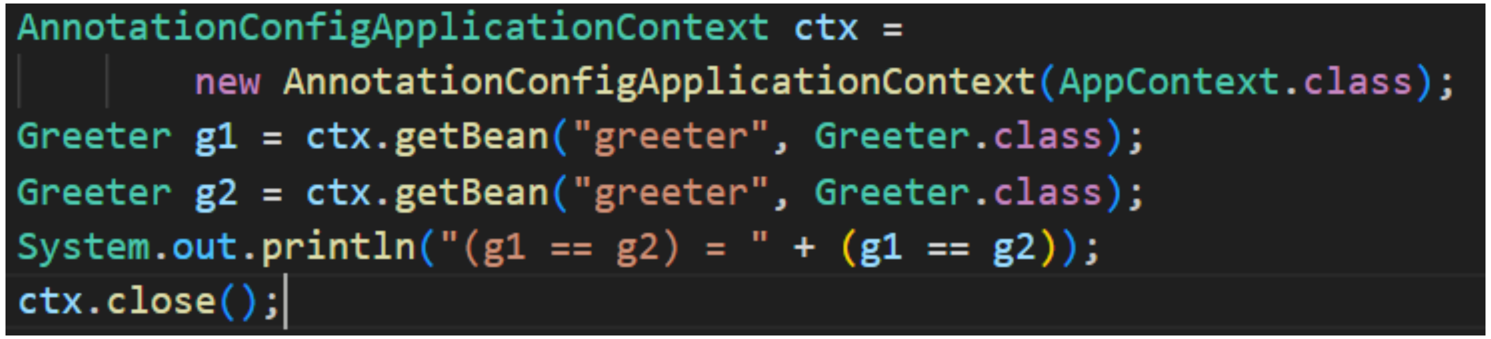


```java
public class Greeter {
    private String format;

    public String greet(String guest) {
        return String.format(format, guest);
    }

    public void setFormat(String format) {
        this.format = format;
    }
}
```



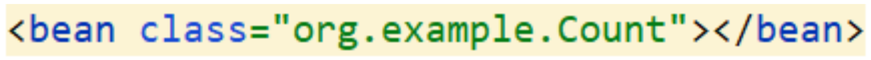
```java
@Configuration
public class AppContext {

    @Bean
    public Greeter greeter() {
        Greeter g = new Greeter();
        g.setFormat(format:"%s, 안녕하세요!");
        return g;
    }

}
```

1.4 실습

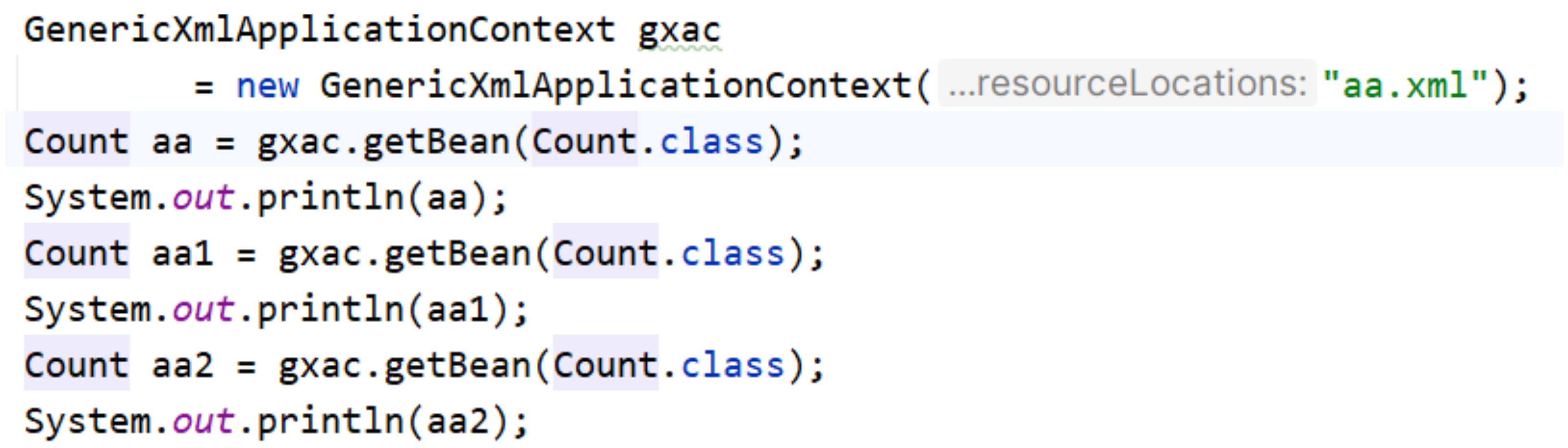


```
AnnotationConfigApplicationContext ctx =
        new AnnotationConfigApplicationContext(AppContext.class);
Greeter g1 = ctx.getBean("greeter", Greeter.class);
Greeter g2 = ctx.getBean("greeter", Greeter.class);
System.out.println("(g1 == g2) = " + (g1 == g2));
ctx.close();
```

XML 방식으로 객체 조립



```
<bean class="org.example.Count"></bean>
```



```
GenericXmlApplicationContext gxac
        = new GenericXmlApplicationContext( ...resourceLocations: "aa.xml");
Count aa = gxac.getBean(Count.class);
System.out.println(aa);
Count aa1 = gxac.getBean(Count.class);
System.out.println(aa1);
Count aa2 = gxac.getBean(Count.class);
System.out.println(aa2);
```

고맙습니다.