스프링시작하기

스프링 2강

스프링DI
1) 객체 의존과 의존주입
2) 객체 조립
3) 스프링 DI설정

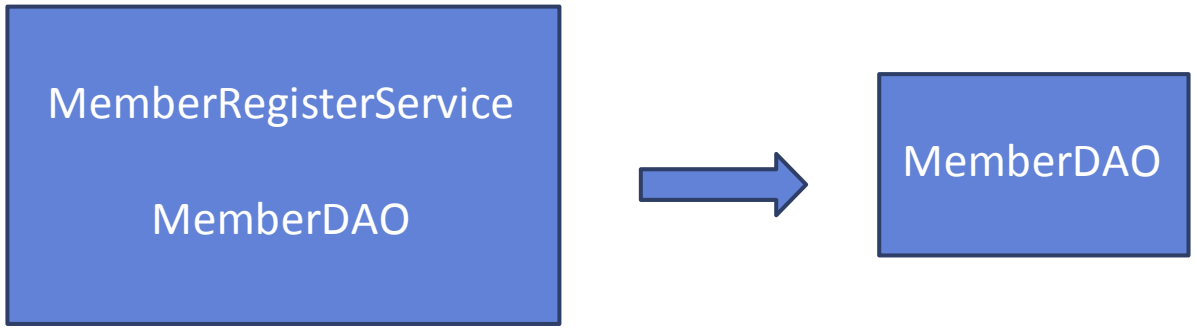
박명회

1.1 의존이란?

1) DI는 Dependency Injection의 약자로 우리말로는 의존주입이라고 번역한다.

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

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



1.2 스프링의 DI설정

AppCtx.java

```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();
    }
}
```

1.3 DI 방식 1 : 생성자 방식

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

```
private MemberDao memberDao;

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

1.4 DI 방식 2 : 세터 메서드 방식

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

```
@Bean
public MemberInfoPrinter infoPrinter() {
    MemberInfoPrinter infoPrinter = new MemberInfoPrinter();
    infoPrinter.setMemberDao(memberDao());
    infoPrinter.setPrinter(memberPrinter());
    return infoPrinter;
}
```

1.5 두개 이상의 설정 파일 사용하기

1. AppConf1.java 2. AppConf2.java

```java
@Configuration
public class AppConf1 {
    @Bean
    public MemberDao memberDao() {
        return new MemberDao();
    }
    @Bean
    public MemberPrinter memberPrinter() {
        return new MemberPrinter();
    }
}
```

```java
@Configuration
public class AppConf2 {
    @Autowired
    private MemberDao memberDao;
    @Autowired
    private MemberPrinter memberPrinter;

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

```java
private static ApplicationContext ctx = null;

Run | Debug
public static void main(String[] args) throws IOException {
    ctx = new AnnotationConfigApplicationContext(AppConf1.class, AppConf2.class);
```

1.6 @Configuration, @Bean, @Autowired

1. AppConf1.java 2. AppConf2.java

```java
@Configuration
public class AppConf1 {
    @Bean
    public MemberDao memberDao() {
        return new MemberDao();
    }
    @Bean
    public MemberPrinter memberPrinter() {
        return new MemberPrinter();
    }
}
```

```java
@Configuration
public class AppConf2 {
    @Autowired
    private MemberDao memberDao;
    @Autowired
    private MemberPrinter memberPrinter;

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

```java
private static ApplicationContext ctx = null;

Run | Debug
public static void main(String[] args) throws IOException {
    ctx = new AnnotationConfigApplicationContext(AppConf1.class, AppConf2.class);
```

1.7 @Import 어노테이션 사용

1. AppConfImport.java

```java
@Configuration
@Import({AppConf2.class})
public class AppConfImport {

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

    @Bean
    public MemberPrinter memberPrinter() {
        return new MemberPrinter();
    }
}
```

고맙습니다.