2024/10/02 3일차

만들어보기

HRService - EmployeeService ,DeptService,JobService 의존

EmployeeService - Emp,Dept,JobDao 의존

DeptService - Emp,Dept 의존

JobService - JobDao 의존

EmpDao

DeptDao

JobDao

@Autowired 객체주입

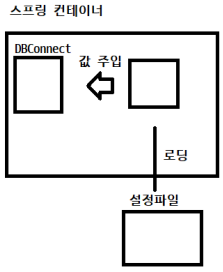
값 주입하기

```java
@Component
public class DBConnect {
    @Value("${db.url}")      // $ 표현식 값이 치환되게 사용
    private String url;

    @Value("${db.username}")
    private String username;

    @Value("${db.password}")
    private String password;

    public void config() {
        System.out.println("url:"+ url);
        System.out.println("username:"+ username);
        System.out.println("password:"+ password);
    }
}
```




```xml
<!--
    <context:component-scan/> 태그
        + 스프링 컨테이너에 등록될 준비가 된 클래스를 스캔해서 스프링의 빈으로 등록시킨다.
        + 등록된 빈의 의존관계를 조사해서 의존성 주입한다.
            * 실제로는 위의 작업을 수행하는 여러 개의 xxxAnnotationProcessor 객체를
              스프링 컨테이너의 빈으로 등록시킨다.
-->
<context:component-scan base-package="di_7.prop"/>

<!--
    <context:property-placeholder/> 태그
        + PropertiesPlaceholderConfigure객체를
          스프링의 빈으로 등록시킨다.
        * 지정된 위치의 properties 파일에 설정된 설정정보를 읽어온다.
        * 스프링 컨테이너에 등록된 모든 빈을 조사해서
          @Value어노테이션을 분석해서 적절한 설정값을 주입한다.
-->
<context:property-placeholder
    location="classpath:/context/di-7.properties"/>
```

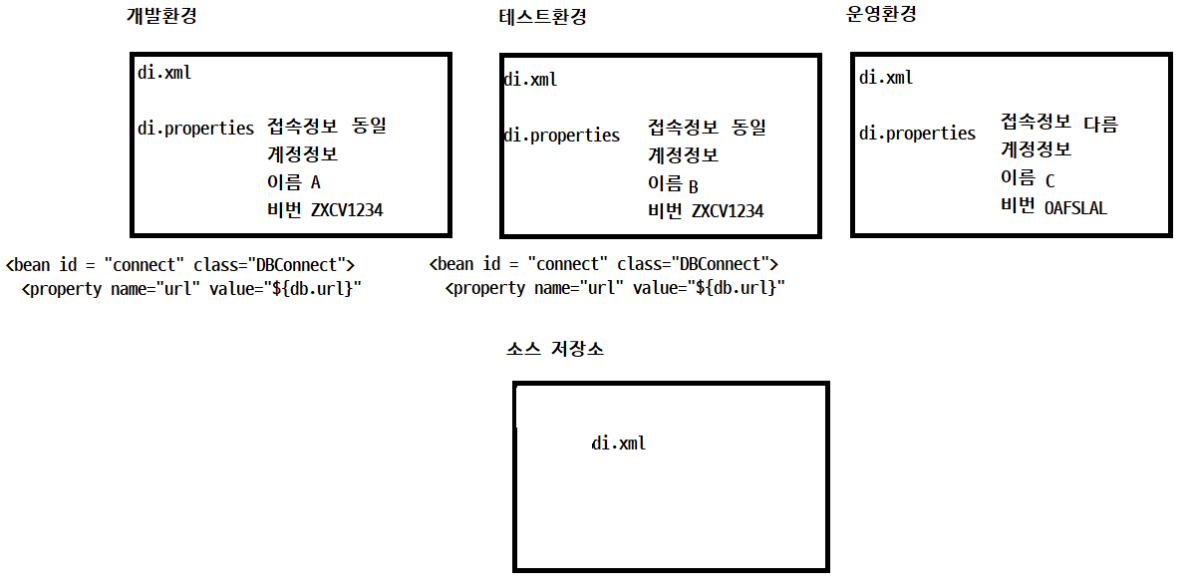
```
db.url =oracle:jdbc:thin:@localhost:1521:xe
db.username =hta
db.password =zxcv1234

# smtp 이메일서버 연결정보
smtp.host=stmp.gmail.com
smtp.port=587
smtp.username=hong
smtp.password=1234567890
```

```
@Component 컨테이너가 스캔할 수 있도록
public class A {
```

```
 B b;
 @Autowired 의존성 주입
 set ...
}

빈 와이어링 : 객체끼리 묶어준다.
    수동or자동으로 가능
```



- **빈(Bean)**
  - **스프링에서 스프링 컨테이너가 생성한 객체를말한다**
  - **스프링 컨테이너가 라이프 사이클을 관리한다.(생성하고 유지보관한다.)**
  - **의존성 주입에 이용한다.**
- **스프링 컨테이너의 스캔 대상 클래스**
  - 아래의 어노테이션이 부착된 클래스는 스캔대상 클래스다.
    - **@Component: 스프링 빈 정의**
    - **@Repository: 데이터 접근 클래스**
    - **@Service: 비즈니스 로직 처리**
    - **@Controller: 웹 요청 처리**
    - **@RestController: RESTful 웹 서비스**

- **@ControllerAdvice: 공통 기능 정의**
- **@RestControllerAdvice: REST 컨트롤러 공통 기능**
- **@Configuration: Java 설정 클래스**

JavaConfig를 이용한 객체 생성 및 조립

- **Spring의 JavaConfig는 XML 기반의 설정을 대체/보완하기 위한 자바 기반 설정방법이다.**
- Java Config의 장점
  - XML 설정보다 더 직관적이다.
  - 타입 안정성(type-safety)를 제공한다.
  - IDE의 자동완성 기능을 활용할 수 없다.
  - IDE의 자동완성 기능을 활용할 수 없다.

주요 어노테이션

@ Configuratohn

- **이 어노테이션이 붙은 클래스 하나 이상의 스프링 빈을 정의하고 있다.**
- 이 어노테이션이 붙붙은 클래스에서 정의한 스프링 빈은 스프링 컨테이너가 객체를 생성하고 반으로 등록한다
- 즉 다른 변에게 의존성 주입으로 시킬 수 있다.
- Bean 어노테이션이 필수는 아니다.

@ Beann

- 이 어노테이션이 붙은 메소드가 반환하는 객체는 스프링의 반으로 등록한다.
- 이 어노테이션이 붙은 메소드내에서는 객체를 생성하고, 객체를 반환하는 코드가 필요하다.
- 이 어노테이션이 메소드에 매개변수를 정의하면,
  객체 생성에 필요한 객체 혹은 값을 매개변수에 스프링이 의존성 주입한다.)
- @Bean 어노테이션이 필요한경우
  - 클래스를 스캔해서 등록할 수 없는 경우(주로 개발자가 작성하지 않은 외부 라이브러리의 클래스(

- 객체를 생성하고 초고화하는 과정이 매우 복잡한 경우

Config

- JavaConfig이용한 객체 생성및 조립

  스프링빈으로 등록될 클래스 준비하기

```
kr.co.jhta
BoardDao.java
BoardService.java
------------------------
--------
@Repository 스캔
class BoardDao{
...
}
------------------------
--------
@Service
class BoardService{
@autowireed
private BoarDao boarDa
o;

...
}
------------------------
--------
```

  Java 기반 설정을 정의한다.

```
@Configuration
 이 클래스는 XML 설정파일을
대신하는 자바설정클래스다.
@ComponentScan(basePack
age="kr.co.jhta");
 스프링 빈으로 등록될 준비가
끝난 클래스를 스캔해서 스프링
테이너의 빈으로 등록시킨다.
public class Config {

    @Bean
    @Bean 어노테이션이 부착
된 메소드는 그 메소드가 반환
하는 객체가 스프링 컨테이너의
빈으로 등록된다.
    * 이 경우 스프링 컨테이
너는 객체를 생성하지않고, 이
메소드 반환하는 객체를 스프링
의 빈으로 등록시킨다.
    public PasswordEnco
der passwordEncoder(){
          return new BCr
yptPasswordEncoder();
    }

}

컨테이너가 객체를 만들게 하는
이유
조립에 활용할려고 (의존성 주
입)
```

Bean Scope

- Bean Scop은 객체가 유효한범위로 아래 5가지의 scope이 있다.
- singleton타입
  - 하나의 Bean 정의에 대해서 Spring IoC Container 내에 단 하나의 객체만 존재한다. 최초로 만든거 계속줌
- prototype타입
  - 하나의 Bean 정의에 대해서 다수의 객체가 존재할 수 있다. getBean할때마다 새로만듬

    상태가 있는 객체는 싱글턴객체로 생성할 필요가 없음 즉 스프링 컨테이너에서 안만듬

싱글턴 객체

잘못된코드

```
@Service
public class UserService {
@Autowried
private UserDao userDao

    public void 회원가입 (User user){
        User savedUser = userDao.getUserByid(user.getid());
        if(savedUser !=) {
            throw new RuntimeException("아이디중복");
        }

    }

}
```

💡 싱글턴 객체 주의점 !

싱글턴 객체에는 멤버변수가 없어야함. 절대로 없어야함 . 유지가 안되기 때문에

싱글턴 객체에 멤버변수가 있는경우 : 읽기전용으로만 쓸 때

정상적인 코드(읽기전용)

```
@Service
public class FileUploaderSerivce{

    @AutoWired <- 이거 못씀

    private final fileDao fileDao;  // 파이널을 붙여서 수정 못함
읽기전용
    private final String saveDirectory;

    권장
    @Autowire
    public FileUploaderService(FileDao fileDao, @Value("${f
ile.save-directory}") String saveDirectory
// 생성자 메서드는 가능함 (한번만 생성) ,Setter 메서드(여러번 생성 가
능)는 허용 안함
        this.fileDao = fileDao;
        this.saveDirectory = saveDirectory;
    }
```

AOP 구현체

스프링에서는 메서드 실행시점에 공통기능을 실행한다.

Target : 핵심기능이 구현된 객체

Joinpoint : 공통기능이 적용된지점 스프링은 메소드실행시점 조인포인트만 지원

공통기능 두가지 패키지로 구성

- Adivce 내가 실행하고싶은 공통기능
  - when+ what
- Pointcute 적용규칙
  - where

공통기능과 적용규칙 작성하기

메서드이름 명확하게

get find set