

## 04강 자바 코드를 이용한 설정

## 예제 준비

-spring.exception

- AlreadyExistingMemberException.java
- IdPasswordNotMatchingException.java

-spring.vo

- Member.java
- RegisterRequest.java

-spring.dao

- MemberDao.java

-spring.service

- MemberRegisterService.java

-spring.printer

- MemberPrinter.java
- MemberInfoPrinter.java

-spring.config

-spring.main

## 자바 코드 설정 기초

- 자바를 이용해서 설정하는 것은 XML과 크게 다르지 않다.
- 자바코드가 스프링 설정이라는 것을 알려주기 위한 애노테이션(@Configuration)을 클래스에서 설정해두고 각각의 코드가 빈 객체임을 알려주기 위한 애노테이션(@Bean)을 설정해 둔다.
- 차이점
  - 자바 코드를 이용해서 빈 객체를 생성
  - AnnotationConfigApplicationContext 클래스를 이용해서 컨테이너를 생성함

### 예제1

## 자바 코드 설정 기초

- Spring.config 패키지에 JavaConfig클래스를 생성한다.
  - 해당 클래스가 스프링 설정파일임을 나타내주는 어노테이션 (@Configuration)을 붙여 준다.

```
@Configuration  
public class JavaConfig {
```

- 어노테이션(@Bean)을 통해서 빈객체를 생성하는 메서드를 만들어 본다.

```
// <bean id="memberDao" class="spring.dao.MemberDao"></bean>  
@Bean  
public MemberDao memberDao() {  
    return new MemberDao();  
}
```

---

## 자바 코드 설정 기초

- Spring.config 패키지에 JavaConfig클래스를 생성한다.

```
@Configuration
public class JavaConfig {

    // <bean id="memberDao" class="spring.dao.MemberDao"></bean>
    @Bean
    public MemberDao memberDao() {
        return new MemberDao();
    }

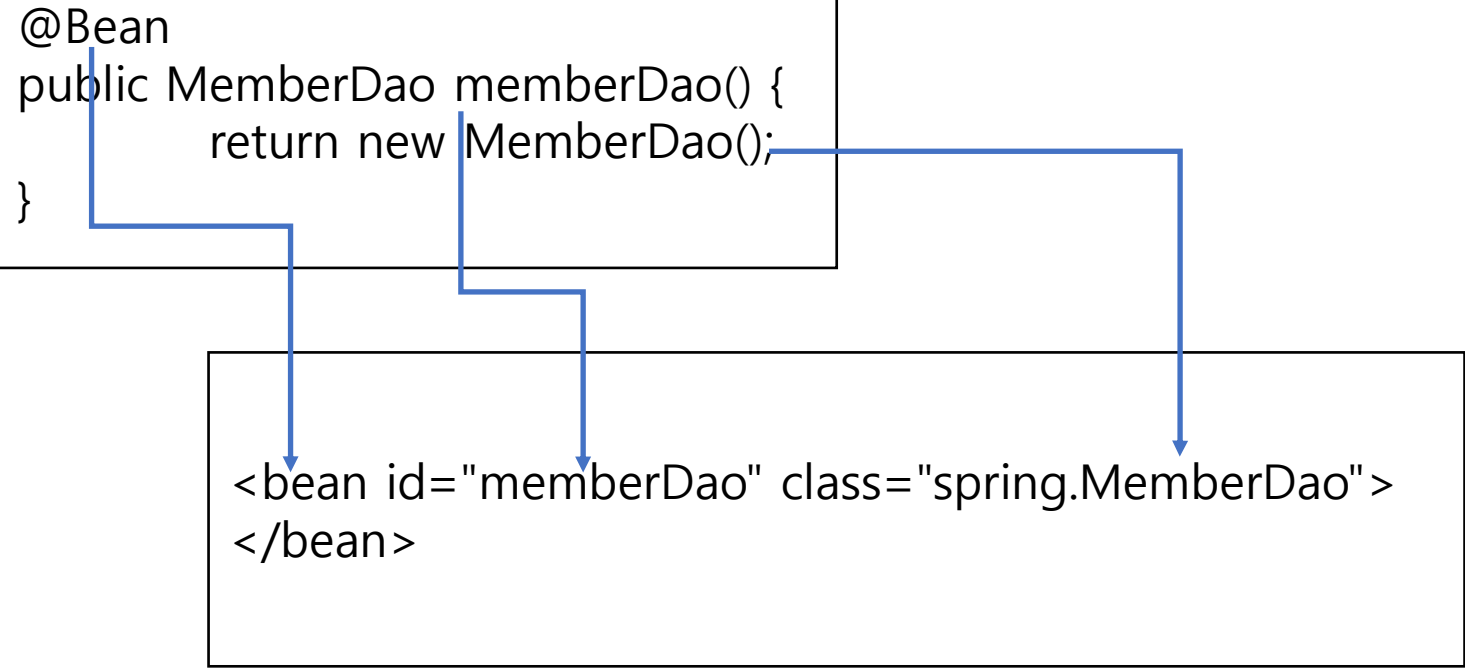
    @Bean
    public MemberRegisterService memberRegSvc() {
        return new MemberRegisterService(memberDao());
    }

    @Bean
    public MemberPrinter printer() {
        return new MemberPrinter();
    }

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

## 자바 코드 설정 기초

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



```
<bean id="memberDao" class="spring.MemberDao">  
</bean>
```

## 자바 코드 설정 기초

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

주입

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

## 자바 코드 설정 기초

- 자바 코드를 이용한 설정파일을 활용해서 스프링 컨테이너를 생성하는 코드를 작성해보자

```
public class Main01 {  
    public static void main(String[] args) {  
        ApplicationContext ctx =  
            new AnnotationConfigApplicationContext(JavaConfig.class);  
  
        MemberRegisterService regSvc =  
            ctx.getBean("memberRegSvc", MemberRegisterService.class);  
  
        MemberInfoPrinter infoPrinter =  
            ctx.getBean("infoPrinter", MemberInfoPrinter.class);  
  
        RegisterRequest regReq = new RegisterRequest();  
        regReq.setName("홍길동");  
        regReq.setEmail("hong@naver.com");  
        regReq.setPassword("1234");  
        regReq.setConfirmPassword("1234");  
  
        regSvc.regist(regReq); // 회원 가입 => Dao의 Map을 채운것  
  
        infoPrinter.printMemberInfo("hong@naver.com"); // 이메일을 이용 => 회원 정보 출력  
    }  
}
```

---



## 자바 코드 설정과 자동 주입

- 자바코드로 설정 하는 경우 별도의 작업이 없어도 자동 주입 기능을 사용할 수 있다.
- 자동 주입 대상이 되는 set메소드에는 @Autowired를 붙여야 한다.
- 생성자는 자동 주입 대상이 될수 없다.

### 예제3

```
public class MemberInfoPrinter {  
  
    private MemberDao memDao;  
    private MemberPrinter printer;  
  
    public void setMemberDao(MemberDao memberDao) {  
        this.memDao = memberDao;  
    }  
  
    @Autowired  
    public void setPrinter(MemberPrinter printer) {  
        this.printer = printer;  
    }  
}
```

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

## 두 개 이상의 클래스를 사용한 설정

- XML과 마찬가지로 자바 클래스도 2개 이상의 파일로 나눠서 활용이 가능하다.
- 다만 다른 클래스에 존재하는 객체를 메소드로 바로 불러오는 것은 불가능하므로 주입에 사용할 빈 객체를 주입 받은 다음 내 객체에 주입을 할 수 있다.
- 스프링은 @Configuration 애노테이션이 적용된 클래스는 자동 주입 적용 대상이 된다. 그러므로 위 예제를 다음과 같이 바꿔도 마찬가지로 적용이 된다.

```
@Configuration
public class ConfigPart1 {
    @Bean
    public MemberDao dao() { // 빈 객체 등록
        return new MemberDao();
    }

    @Bean
    public MemberRegisterService memberRegSvc() {
        return new MemberRegisterService(dao()); // 생성자를 통한 빈 객체 주입
    }
}
```

```
@Configuration
public class ConfigPart2 {

    @Autowired
    private ConfigPart1 configPart1;

    @Bean
    public MemberPrinter printer() {
        return new MemberPrinter();
    }

    @Bean
    public MemberInfoPrinter infoPrinter() {
        MemberInfoPrinter infoPrinter = new MemberInfoPrinter();
        infoPrinter.setMemberDao(configPart1.dao()); // set 메서드를 통한 빈객체 주입
        infoPrinter.setPrinter(printer()); // set메서드에는 자동 주입 기능이 활성화되어 있음

        return infoPrinter;
    }
}
```

## 두 개 이상의 클래스를 사용한 설정

- XML과 마찬가지로 자바 클래스도 2개 이상의 파일로 나눠서 활용이 가능하다.
- 다만 다른 클래스에 존재하는 객체를 메소드로 바로 불러오는 것은 불가능하므로 주입에 사용할 빈 객체를 주입 받은 다음 내 객체에 주입을 할 수 있다.
- 또한 설정 객체를 가져올 때도 두 설정 클래스 모두 가져와야 한다.

```
// java 설정 파일이 두개 이상 있는 경우 스프링 설정파일 불러오기
ApplicationContext ctx =
    new AnnotationConfigApplicationContext(ConfigPart1.class, ConfigPart2.class);
```

## 두 개 이상의 클래스를 사용한 설정

- XML과 마찬가지로 Import방식으로 다른 설정 클래스를 가져오는 것도 가능하다.
- 단 설정파일이 여러 개인 경우 배열의 형태로 가져온다..

```
@Import({ConfigPart1.class, ConfigPart2.class, ConfigPart3.class, ....})
```

```
@Configuration
@Import(ConfigPartSub.class) //다른 자바 설정파일을 가져와서 하나의 설정파일로 합치는 방법
//@Import({ConfigPartSub.class,ConfigPartThird.class})// 두개 이상 설정파일을 합칠때는 배열 형태로 합쳐야 한다.
public class ConfigPartMain {
```

```
    @Bean
    public MemberDao dao() { //빈 객체 등록
        return new MemberDao();
    }
```

```
    @Bean
    public MemberRegisterService memberRegSvc() {
        return new MemberRegisterService(dao()); // 생성자를 통한 빈 객체 주입
    }
}
```

```
@Configuration
public class ConfigPartSub {
    @Autowired
    private MemberDao dao;

    @Bean
    public MemberPrinter printer() {
        return new MemberPrinter();
    }

    @Bean
    public MemberInfoPrinter infoPrinter() {
        MemberInfoPrinter infoPrinter = new MemberInfoPrinter();
        infoPrinter.setMemberDao(dao); //set 메서드를 통한 빈객체 주입
        infoPrinter.setPrinter(printer()); // set메서드에는 자동 주입 기능이 활성화되어 있음
        return infoPrinter;
    }
}
```

## 두 개 이상의 클래스를 사용한 설정

- XML과 마찬가지로 Import방식으로 다른 설정 클래스를 가져오는 것도 가능하다.
- 단 설정파일이 여러 개인 경우 배열의 형태로 가져온다..

```
@Import({ConfigPart1.class, ConfigPart2.class, ConfigPart3.class, ....})
```

```
//두개 이상 java설정파일을 하나로 합쳐서 설정불러오기
```

```
ApplicationContext ctx =  
    new AnnotationConfigApplicationContext(ConfigPartMain.class);
```

## 설정 클래스와 설정 XML의 혼합

- 스프링은 XML설정 과 자바 설정을 같이 지원한다.
  - 어떤 때에는 XML이 어떤 때에는 자바 설정이 편할 수 있으므로 이런 경우 각각의 편한 방식으로 설정한 후 하나로 합쳐서 처리 할 수 있다.
1. 자바설정에서 XML설정을 임포트 하기 – 예제 7
  2. XML설정에서 자바설정을 임포트 하기 – 예제 8

# 설정 클래스와 설정 XML의 혼합

## 1. 자바설정에서 XML설정을 импорт 하기

- JavaMainConfig.java (+sub-config.xml)

```
@Configuration
@ImportResource("classpath:sub-config.xml") //xml 설정파일을 가져와서 자바 설정 파일에 합치기
public class JavaMainConfig {

    @Bean
    public MemberDao dao() { //빈 객체 등록
        return new MemberDao();
    }

    @Bean
    public MemberRegisterService memberRegSvc() {
        return new MemberRegisterService(dao()); // 생성자를 통한 빈 객체 주입
    }
}
```

```
<bean id="printer" class="spring.printer.MemberPrinter" />
```

```
<bean id="infoPrinter" class="spring.printer.MemberInfoPrinter" >
    <property name="memberDao" ref="dao"/>
    <property name="printer" ref="printer"/>
</bean>
```

# 설정 클래스와 설정 XML의 혼합

1. 자바설정에서 XML설정을 импорт 하기
  - JavaMainConfig.java (+sub-config.xml)

//java와 xml 서로 다른 설정파일을 하나로 합쳐서 설정 불러오기 (JAVA <= XML)

```
ApplicationContext ctx =  
    new AnnotationConfigApplicationContext(JavaMainConfig.class);
```



# 설정 클래스와 설정 XML의 혼합

## 2. XML설정에서 자바설정을 импорт 하기

- main-config.xml (+JavaSubConfig.java)

```
<context:annotation-config/>
```

```
<bean class="spring.config.JavaSubConfig" />
```

```
<bean id="dao" class="spring.dao.MemberDao" />
```

```
<bean id="memberRegSvc" class="spring.service.MemberRegisterService">  
    <constructor-arg ref="dao"/>  
</bean>
```

```
@Configuration  
public class JavaSubConfig {  
  
    @Autowired  
    private MemberDao dao;  
  
    @Bean  
    public MemberPrinter printer() {  
        return new MemberPrinter();  
    }  
  
    @Bean  
    public MemberInfoPrinter infoPrinter() {  
        MemberInfoPrinter infoPrinter = new MemberInfoPrinter();  
        infoPrinter.setMemberDao(dao);           //set 메서드를 통한 빈객체 주입  
        infoPrinter.setPrinter(printer());       // set메서드에는 자동 주입 기능이 활성화되어 있음  
  
        return infoPrinter;  
    }  
}
```

# 설정 클래스와 설정 XML의 혼합

## 2. XML설정에서 자바설정을 импорт 하기

- main-config.xml (+JavaSubConfig.java)

//java와 xml 서로 다른 설정파일을 하나로 합쳐서 설정 불러오기 (XML <= JAVA)

```
ApplicationContext ctx =  
    new GenericXmlApplicationContext("classpath:main-config.xml");
```