JavaStudy ***SPRING***

https://github.com/tomfromkr/javastudy

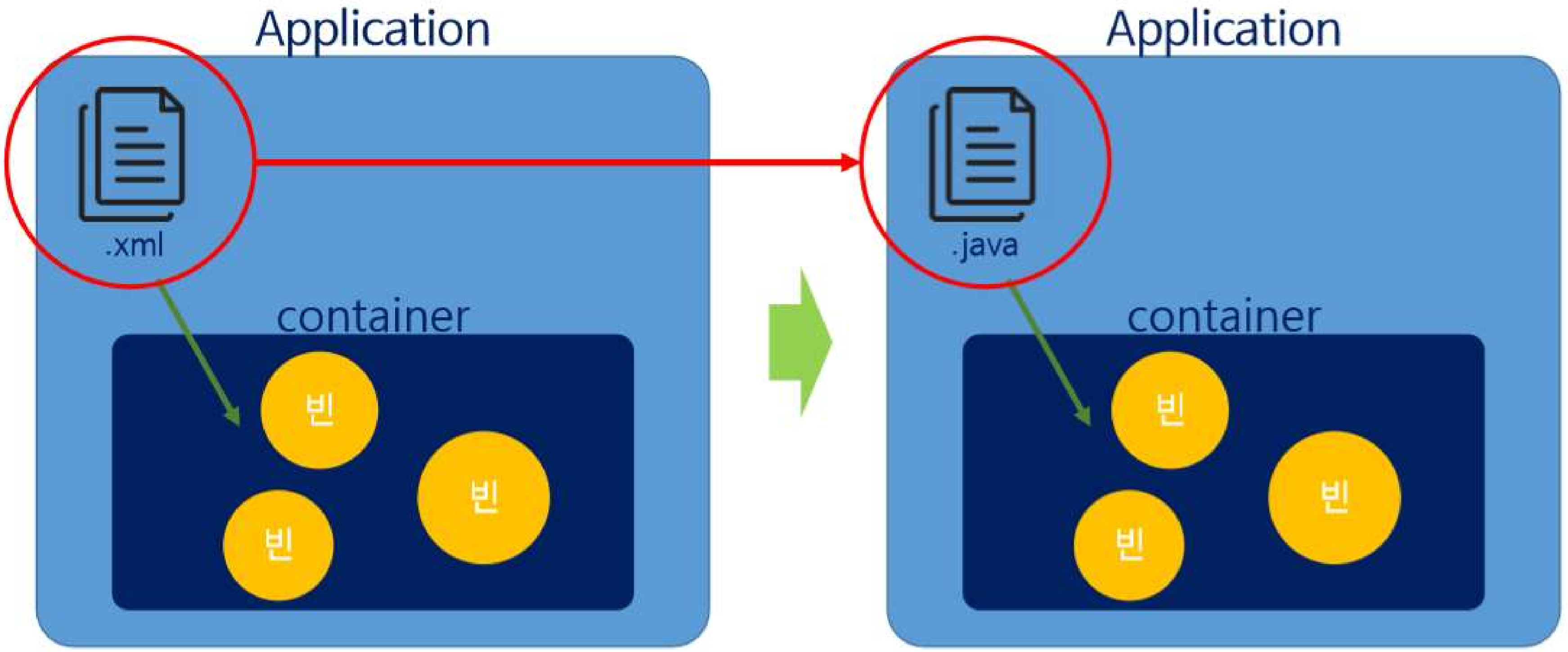
스프링 어노테이션

스프링 컨테이너에서 사용하는 빈 객체들끼리
의존성이 주입이 되어서 프로젝트를 만들 수 있다.

이러한 스프링 프로젝트를 설정해주는 설정 파일을 기존에는 xml을 이용하였다.

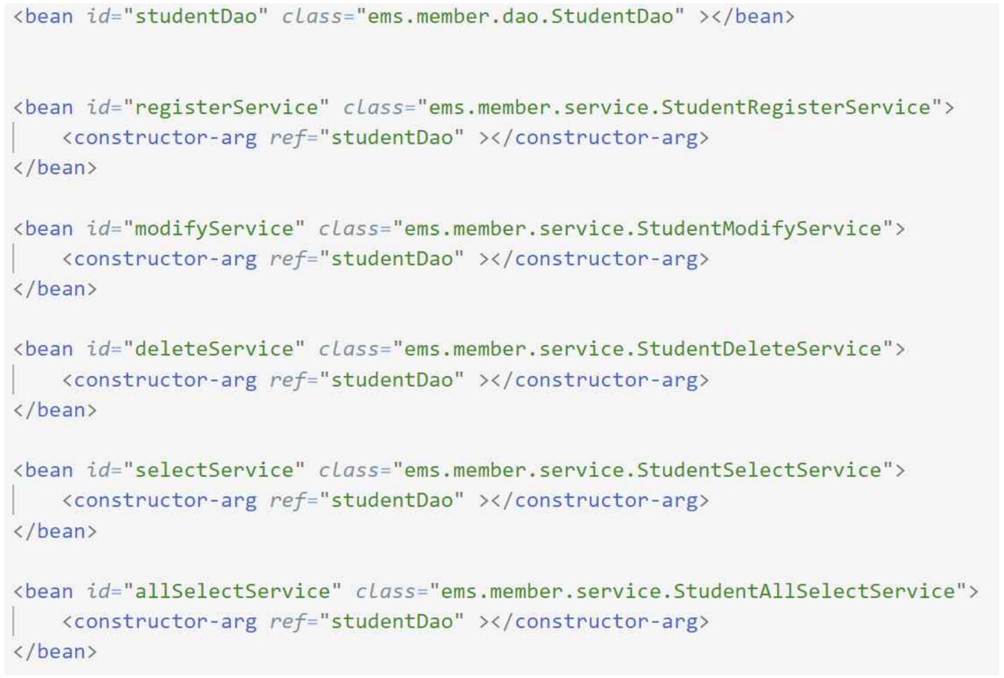
이 때 xml을 사용하지 않고 자바파일을 이용하자

JavaStudy ***Spring***

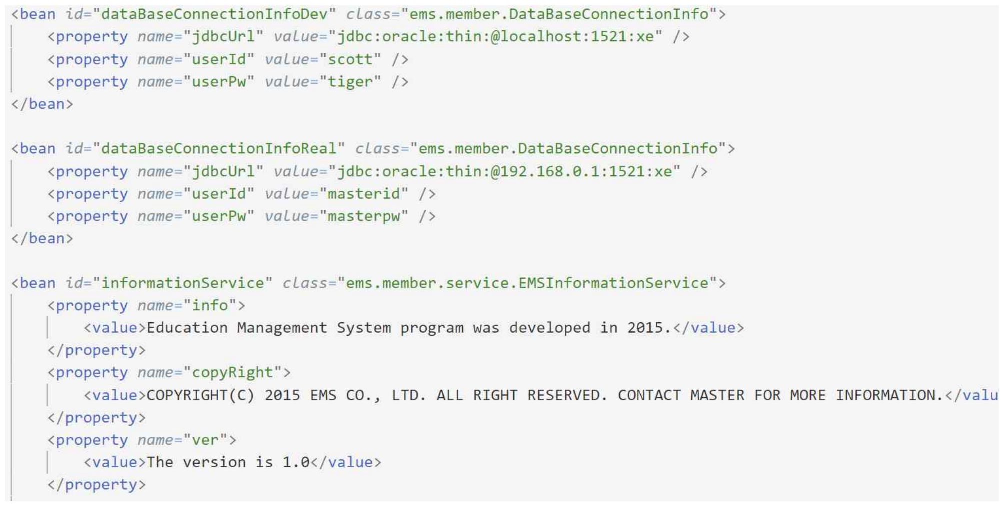


< >

JavaStudy **Spring**

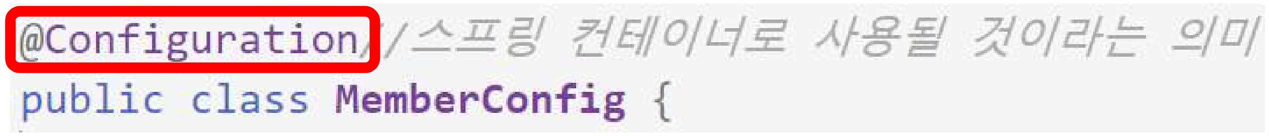


```xml
<bean id="studentDao" class="ems.member.dao.StudentDao" ></bean>


<bean id="registerService" class="ems.member.service.StudentRegisterService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>

<bean id="modifyService" class="ems.member.service.StudentModifyService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>

<bean id="deleteService" class="ems.member.service.StudentDeleteService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>

<bean id="selectService" class="ems.member.service.StudentSelectService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>

<bean id="allSelectService" class="ems.member.service.StudentAllSelectService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>
```



```xml
<bean id="dataBaseConnectionInfoDev" class="ems.member.DataBaseConnectionInfo">
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="userId" value="scott" />
    <property name="userPw" value="tiger" />
</bean>

<bean id="dataBaseConnectionInfoReal" class="ems.member.DataBaseConnectionInfo">
    <property name="jdbcUrl" value="jdbc:oracle:thin:@192.168.0.1:1521:xe" />
    <property name="userId" value="masterid" />
    <property name="userPw" value="masterpw" />
</bean>

<bean id="informationService" class="ems.member.service.EMSInformationService">
    <property name="info">
        <value>Education Management System program was developed in 2015.</value>
    </property>
    <property name="copyRight">
        <value>COPYRIGHT(C) 2015 EMS CO., LTD. ALL RIGHT RESERVED. CONTACT MASTER FOR MORE INFORMATION.</valu
    </property>
    <property name="ver">
        <value>The version is 1.0</value>
    </property>
```

스프링 설정파일 생성



```
@Configuration//스프링 컨테이너로 사용될 것이라는 의미
public class MemberConfig {
```

빈 객체 생성

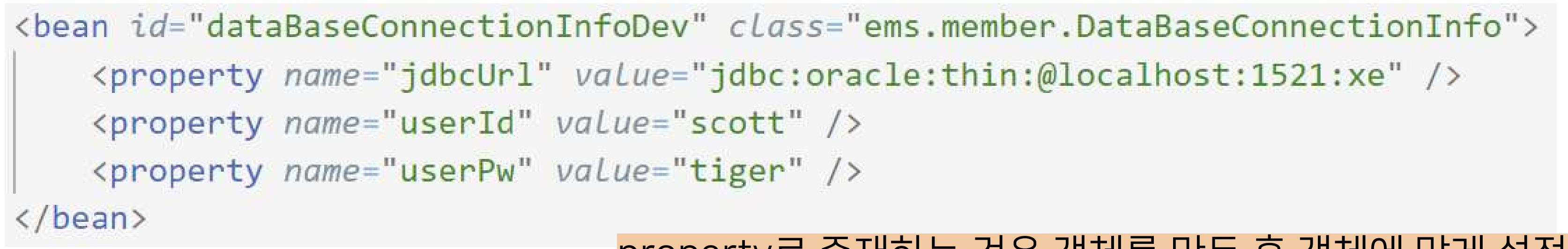
```xml
<bean id="studentDao" class="ems.member.dao.StudentDao" ></bean>
```

```java
@Bean
public StudentDao studentDao() {
    return new StudentDao();
}
```

```xml
<bean id="registerService" class="ems.member.service.StudentRegisterService">
    <constructor-arg ref="studentDao" ></constructor-arg>
</bean>
```

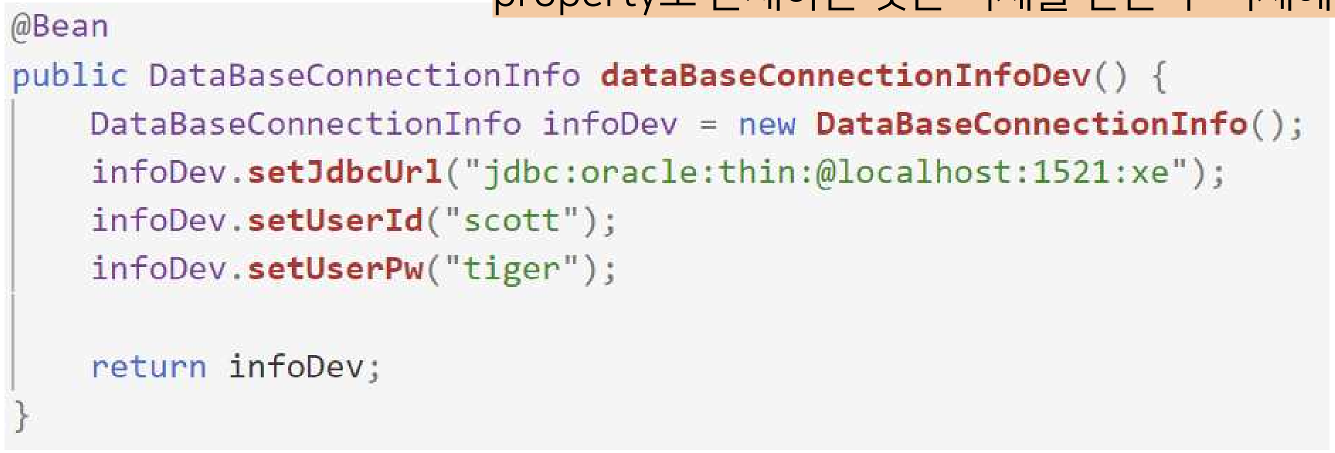
```java
@Bean
public StudentRegisterService registerService() {
    return new StudentRegisterService(studentDao());
}
```

빈 객체 생성



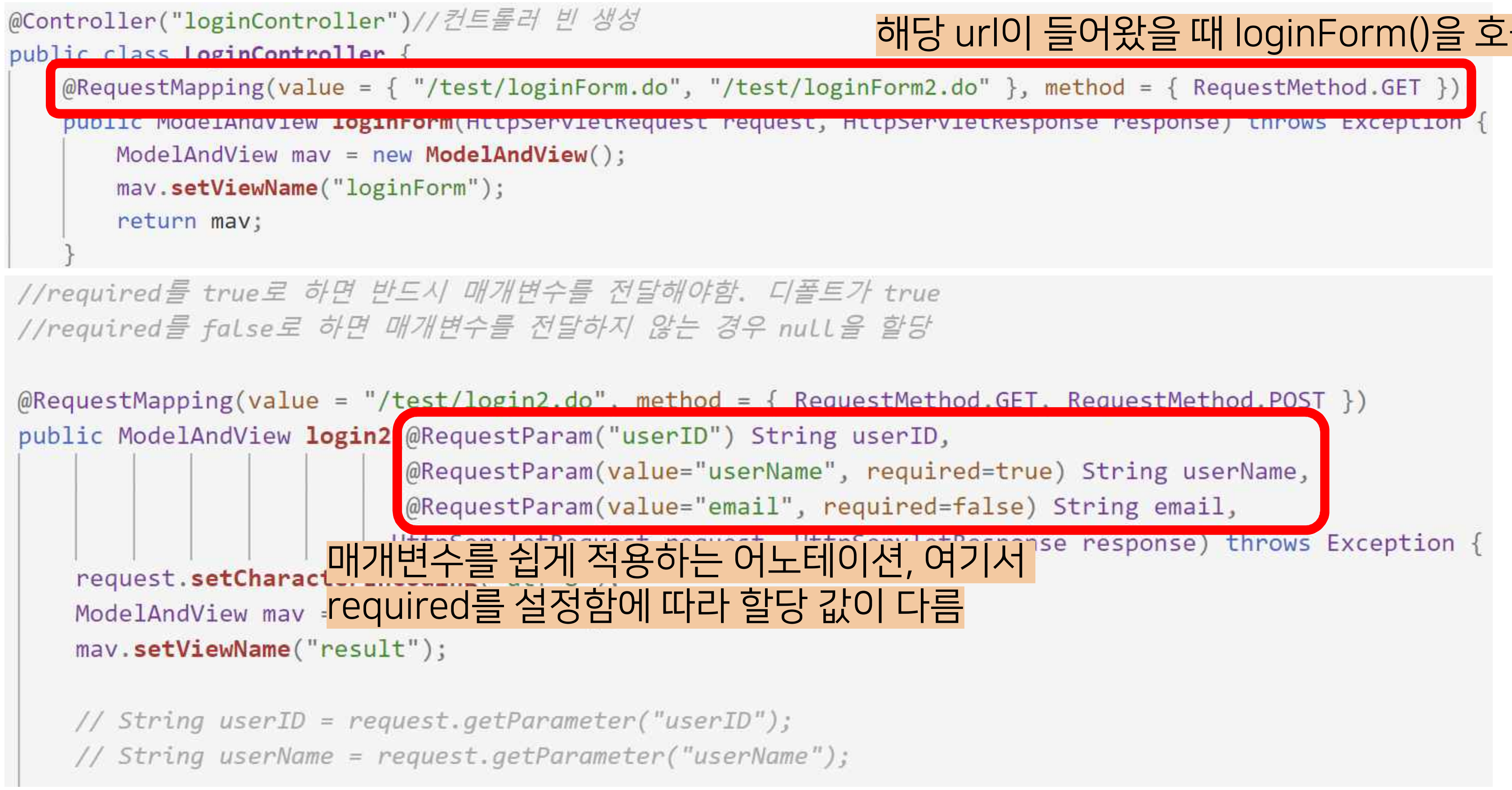
```
<bean id="dataBaseConnectionInfoDev" class="ems.member.DataBaseConnectionInfo">
    <property name="jdbcUrl" value="jdbc:oracle:thin:@localhost:1521:xe" />
    <property name="userId" value="scott" />
    <property name="userPw" value="tiger" />
</bean>
```

property로 존재하는 것은 객체를 만든 후 객체에 맞게 설정해준다.



```
@Bean
public DataBaseConnectionInfo dataBaseConnectionInfoDev() {
    DataBaseConnectionInfo infoDev = new DataBaseConnectionInfo();
    infoDev.setJdbcUrl("jdbc:oracle:thin:@localhost:1521:xe");
    infoDev.setUserId("scott");
    infoDev.setUserPw("tiger");

    return infoDev;
}
```

@RequestMapping, @RequestParam



해당 url이 들어왔을 때 loginForm()을 호출하라

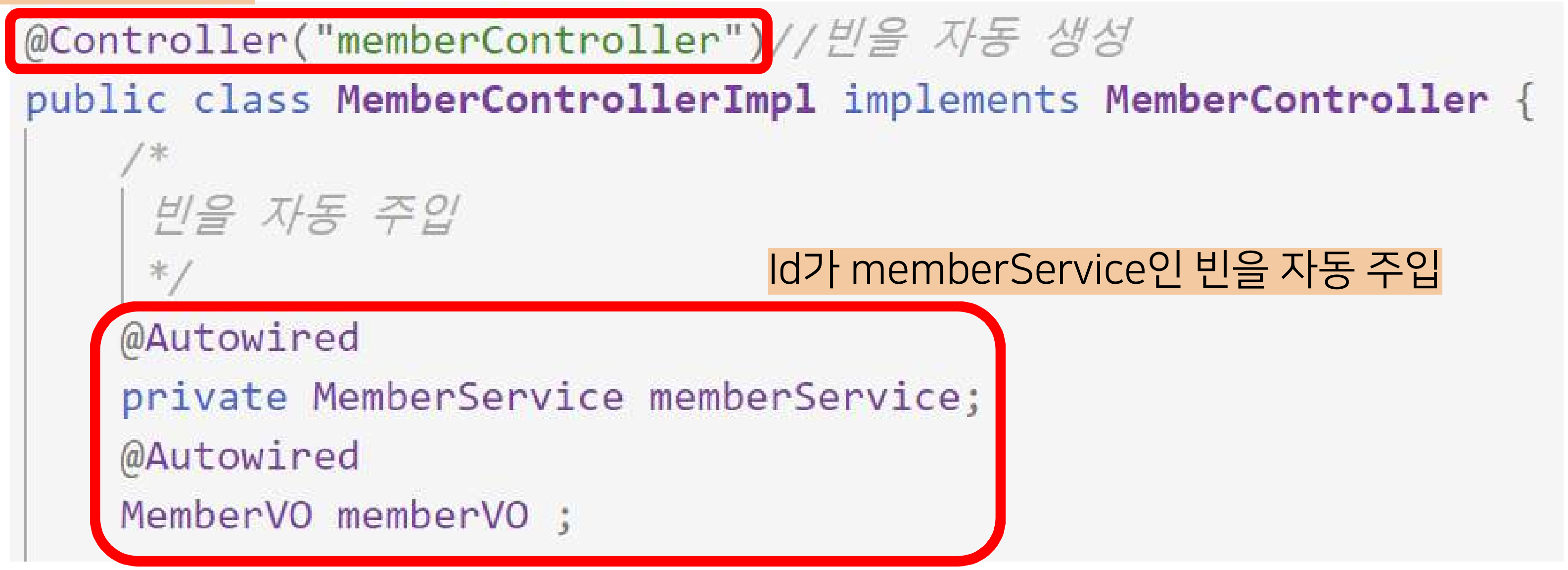
```
@Controller("loginController")//컨트롤러 빈 생성
public class LoginController {
    @RequestMapping(value = { "/test/loginForm.do", "/test/loginForm2.do" }, method = { RequestMethod.GET })
    public ModelAndView loginForm(HttpServletRequest request, HttpServletResponse response) throws Exception {
        ModelAndView mav = new ModelAndView();
        mav.setViewName("loginForm");
        return mav;
    }
```

```
//required를 true로 하면 반드시 매개변수를 전달해야함. 디폴트가 true
//required를 false로 하면 매개변수를 전달하지 않는 경우 null을 할당

@RequestMapping(value = "/test/login2.do", method = { RequestMethod.GET, RequestMethod.POST })
public ModelAndView login2(@RequestParam("userID") String userID,
                           @RequestParam(value="userName", required=true) String userName,
                           @RequestParam(value="email", required=false) String email,
                           HttpServletRequest request, HttpServletResponse response) throws Exception {
    request.setCharacterEncoding("utf-8");
    ModelAndView mav =
    mav.setViewName("result");

    // String userID = request.getParameter("userID");
    // String userName = request.getParameter("userName");
```

매개변수를 쉽게 적용하는 어노테이션, 여기서
required를 설정함에 따라 할당 값이 다름

@Controller, @Autowired

컨트롤러 빈 생성



```
@Controller("memberController")//빈을 자동 생성
public class MemberControllerImpl implements MemberController {
    /*
     빈을 자동 주입
     */
    @Autowired
    private MemberService memberService;
    @Autowired
    MemberVO memberVO ;
```

Id가 memberService인 빈을 자동 주입