박 경 태
comsi.java@gmail.com

고급 자바 프로그래밍

: STS를 이용한 Spring 프로그래밍

강의 내용

순서	내 용
1	• Spring IoC를 이용한 비즈니스 컴포넌트 만들기
2	• Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기 • Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리
3	• Spring MVC를 이용한 MVC 아키텍쳐 적용하기
4	• Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등)
5	• Spring과 MyBatis 연동하기 • Spring과 JPA 연동하기

5. 어노테이션 기반 설정

XML 파일의 과도한 설정에 대한 부담 감소를 위해 어노테이션을 이용한 설정을 지원

어노테이션??

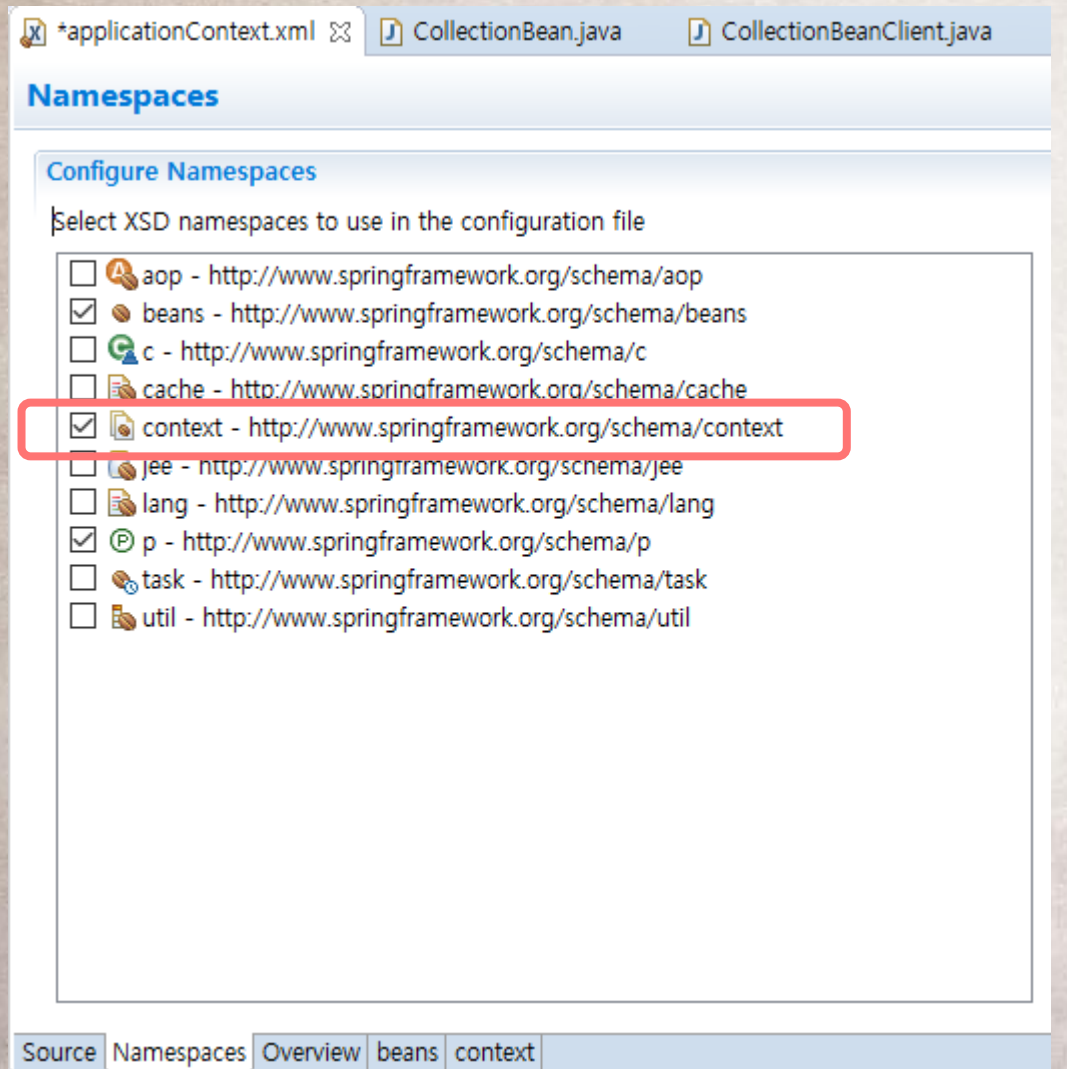
- 일반적으로 Java Annotation은 @를 이용한 주석을 뜻함.
- 자바 어노테이션 (Java Annotation)은 자바5(1.5)부터 제공되기 시작한 기능으로 어노테이션은 한국어로 “주석” 이라 번역.
- Annotation을 “주석”이라 칭할경우, 흔히 쓰던 주석처리 (//, /**/)와 같은 의미로 오해하기 쉽기 때문에, Annotation이라는 명칭을 그대로 사용.
- 주석과 Annotation으로 구분하지만 둘다 프로그래밍의 구동에 영향을 주지 않는 주석임.
- Annotation을 사용하면 코드의 가독성을 높이는 데 도움
  - @Deprecated - 가급적 사용을 자제해 달라는 의미로 개선된 메소드가 있음을 나타내고자 할 때 사용
  - @SuppressWarning(“ ”) - 이 부분에 경고문을 출력하지 말라는 의미
    - @SuppressWarning(“deprecation”) - deprecated 된 메소드를 이용했을 때 발생하는 deprecation 경고문을 띄우지 말라는 의미

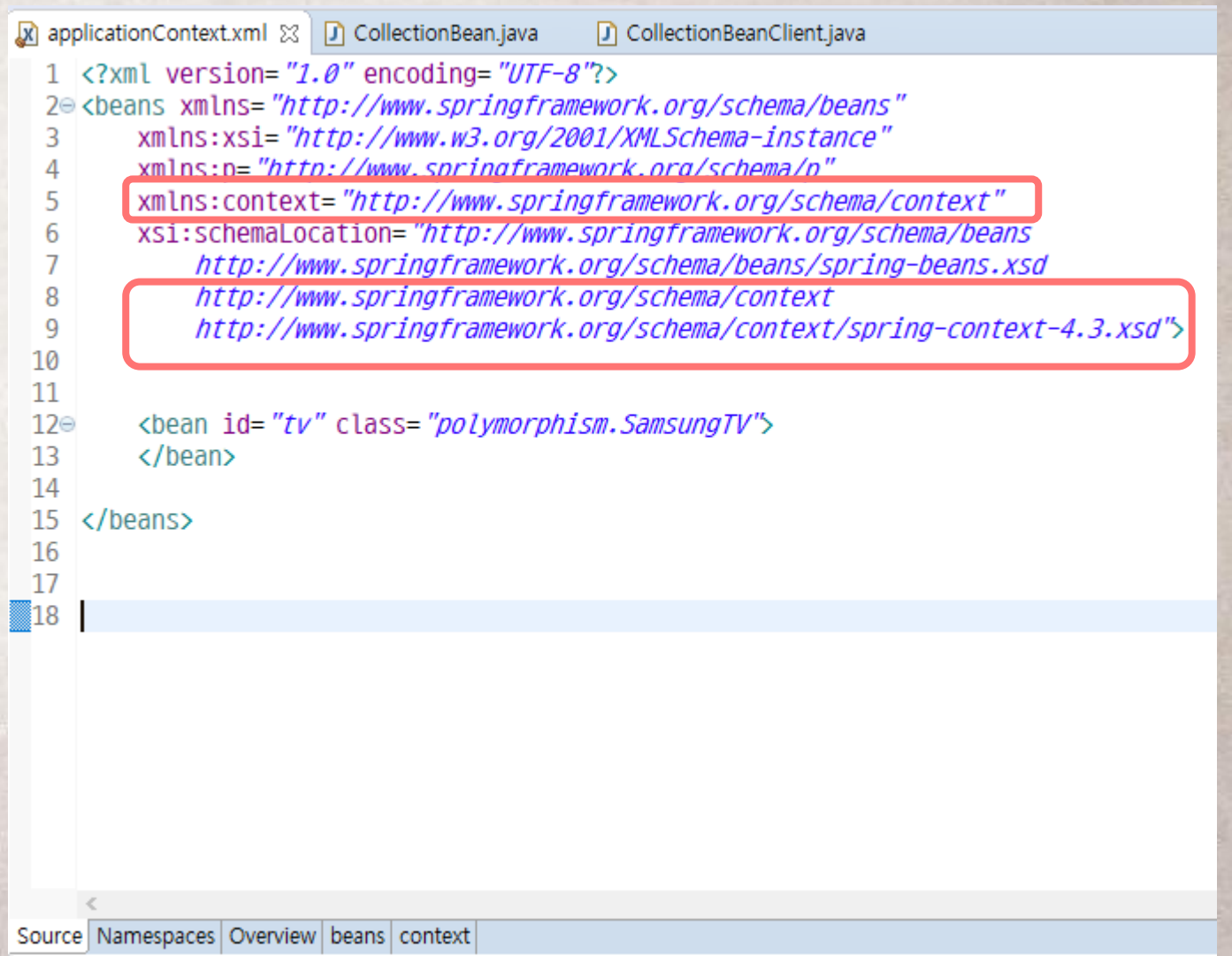
출처: http://cho2.tistory.com/entry/%EC%96%B4%EB%85%B8%ED%85%8C%EC%9D%B4%EC%85%98%EC%9D%B4%EB%9E%80-annotation-Override-Deprecated-SuppressWarnings

5.1 어노테이션 설정 기초

1. Context 네임스페이스 추가

- 어노테이션 설정 추가- context 관련 네임스페이스와 스키마 문서 위치 등록





2. 컴포넌트 스캔(component-scan) 설정

- <bean> 등록하지 않고 자동으로 생성하려면<context:component-scan/> 엘리먼트를 정의
- 클래스 패스에 있는 클래스들을 스캔하여 @Component가 설정된 클래스들을 자동으로 객체 생성

applicationContext.xml

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <beans xmlns="http://www.springframework.org/schema/beans"
3      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
4      xmlns:p="http://www.springframework.org/schema/p"
5      xmlns:context="http://www.springframework.org/schema/context"
6      xsi:schemaLocation="http://www.springframework.org/schema/beans
7          http://www.springframework.org/schema/beans/spring-beans.xsd
8          http://www.springframework.org/schema/context
9          http://www.springframework.org/schema/context/spring-context-4.3.xsd">
10
11     <context:component-scan base-package="polymorphism"></context:component-scan>
12     <!--
13     <bean id="tv" class="polymorphism.SamsungTV">
14     </bean>
15      -->
16 </beans>
17
```

3. @Component

- LgTV 클래스에 대한 <bean> 등록

```
<bean class="polymorphism.LgTV"></bean>
```

<bean>객체 생성

applicationContext.xml | *LgTV.java

```
1  package polymorphism;
2
3  import org.springframework.stereotype.Component;
4
5  @Component
6  public class LgTV implements TV {
7
8      public LgTV(){
9          System.out.println("===> LgTV 객체 생성");
10     }
11
```

@Component annotation 사용

- 컨테이너가 생성한 객체 요청

```
// Spring 컨테이너를 구동
AbstractApplicationContext factory = new GenericXmlApplicationContext("applicationContext.xml");

//BeanFactory factory = new BeanFactory();
TV tv = (TV) factory.getBean("tv");
```

컨테이너가 생성한 객체 요청시에는 요청할 아이디나 이름 필요

- LgTV 클래스에 대한 <bean> 등록

```
<bean id="tv" class="polymorphism.LgTV"></bean>
```

applicationContext.xml | LgTV.java | TVUser.java

```
1  package polymorphism;
2
3  import org.springframework.stereotype.Component;
4
5  @Component("tv")
6  public class LgTV implements TV {
7
8      public LgTV(){
9          System.out.println("===> LgTV 객체 생성");
10     }
11
```

컨테이너가 생성한 객체 요청시에는 요청할 아이디나 이름 필요

- id나 name 속성 미지정 시 이름규칙
  - 스프링 컨테이너가 클래스 객체를 생성할 때, id나 name 속성을 지정하지 않았다면, 컨테이너가 자동으로 이름을 설정
  - 이름 규칙은 클래스 이름의 첫글자를 소문자로 변경하기만 하면된다.
  - LgTV 클래스 객체를 요청하면 lgTV라는 이름을 사용하면됨

applicationContext.xml | LgTV.java | TVUser.java

```java
package polymorphism;

import org.springframework.context.support.AbstractApplicationContext;

public class TVUser {

    public static void main(String[] args){

        // Spring 컨테이너를 구동
        AbstractApplicationContext factory = new GenericXmlApplicationContext("applicationContext.xml");

        //BeanFactory factory = new BeanFactory();
        TV tv = (TV) factory.getBean("tv");

        tv.powerOn();
        tv.volumeUp();
        tv.volumeDown();
        tv.powerOff();

        // Spring 컨테이너를 종료
        factory.close();
    }
}
```

실행결과

Console | Progress | Problems

```
<terminated> TVUser [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 22. 오후 5:46:42)
5월 22, 2017 5:46:42 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 22, 2017 5:46:42 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 17:46:42 KST 201
===> LgTV 객체 생성
LgTV -- 전원을 켠다.
LgTV -- 소리를 올린다.
LgTV -- 소리를 내린다.
LgTV -- 전원을 끈다.
5월 22, 2017 5:46:42 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 17:46:42 KST 2017];
```

5.2 의존성 주입 설정

1.의존성 주입 어노테이션

- 스프링에서 의존성 주입을 지원하는 어노테이션은 @Autowired, @Inject, @Qualifier, @Resource가 있다.

어노테이션	설명
@Autowired	주로 변수 위에 설정하여 해당 타입의 객체를 찾아서 자동으로 할당 org.springframework.beans.factory.annotation.Autowired
@Qualifier	특정 객체의 이름을 이용하여 의존성 주입할 때 사용 org.springframework.beans.factory.annotation.Qualifier
@Inject	@Autowired와 동일한 제공 javax.annotation.Resource
@Resource	@Autowired와 @Qualifier의 기능을 결합한 어노테이션 javax.inject.Inject

- @Autowired와 @Qualifier는 스프링에서 제공하지만 나머지는 스프링에서 제공하지 않는다.

2. @Autowired

- 생성자나 메소드, 멤버변수 위에 모두 사용
- 어디서 사용하는 결과가 같아 상관없지만 대부분 멤버변수 위에 선언하여 사용
- 스프링 컨테이너는 멤버변수 위에 붙은 @Autowired를 확인하는 순간 해당 변수의 타입 체크 후 메모리에서 확인 후 그 객체를 변수에 주입
- 만약 @Autowired가 붙은 객체가 메모리에 없다면 NoSuchBeanDefinitionException을 발생 ➔ 대상 객체가 메모리에 존재하지 않는다는 의미

@Autowired 객체가 없을 경우 발생 오류

applicationContext.xml | LgTV.java | TVUser.java | SonySpeaker.java | Speaker.java

```java
package polymorphism;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component("tv")
public class LgTV implements TV {

    @Autowired
    private Speaker speaker;

    public LgTV(){
        System.out.println("===> LgTV 객체 생성");
    }

```

외부 Speaker를 사용
(SonySpeaker 또는 AppleSpeaker객체)

Console | Progress | Problems

<terminated> TVUser [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (

```
월 22, 2017 6:13:39 오후 org.springframework.beans.factory.xml.X
정보: Loading XML bean definitions from class path resource [appl
월 22, 2017 6:13:39 오후 org.springframework.context.support.Gen
정보: Refreshing org.springframework.context.support.GenericXmlAp
==> LgTV 객체 생성
월 22, 2017 6:13:40 오후 org.springframework.context.support.Gen
경고: Exception encountered during context initialization - cance
xception in thread "main" org.springframework.beans.factory.Unsa
        at org.springframework.beans.factory.annotation.Autowired
        at org.springframework.beans.factory.annotation.Injection
        at org.springframework.beans.factory.annotation.Autowired
        at org.springframework.beans.factory.support.AbstractAuto
        at org.springframework.beans.factory.support.AbstractAuto
        at org.springframework.beans.factory.support.AbstractAuto
        at org.springframework.beans.factory.support.AbstractBean
```

• • •

ogram Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 22. 오후 6:13:39)

```
actory NoSuchBeanDefinitionException: No qualifying bean
ype 'polymorphism.Speaker' available: expected at least
```

발생 오류 제거 - Speaker객체를 메모리에 생성

applicationContext.xml | LgTV.java | TVUser.java | SonySpeaker.java | Speaker.java

```java
package polymorphism;

import org.springframework.stereotype.Component;

@Component("sony")
public class SonySpeaker implements Speaker{

    public SonySpeaker(){
        System.out.println("===> SonySpeaker 객체 생성");
    }

    public void volumeUp(){
        System.out.println("SonySpeaker -- 소리를 올린다.");
    }

    public void volumeDown(){
        System.out.println("SonySpeaker -- 소리를 내린다.");
    }
}
```

@Component 를 사용해 Speaker 객체를 컨테이너에 생성

Console | Progress | Problems

<terminated> TVUser [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 22. 오후 7:00:08)

```
5월 22, 2017 7:00:08 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 22, 2017 7:00:08 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 19:00:
===> LgTV 객체 생성
===> SonySpeaker 객체 생성
LgTV -- 전원을 켠다.
SonySpeaker -- 소리를 올린다.
SonySpeaker -- 소리를 내린다.
LgTV -- 전원을 끈다.
5월 22, 2017 7:00:09 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 19:00:08
```

SonySpeaker가 생성되고 문제없이 실행

3. @Qualifier

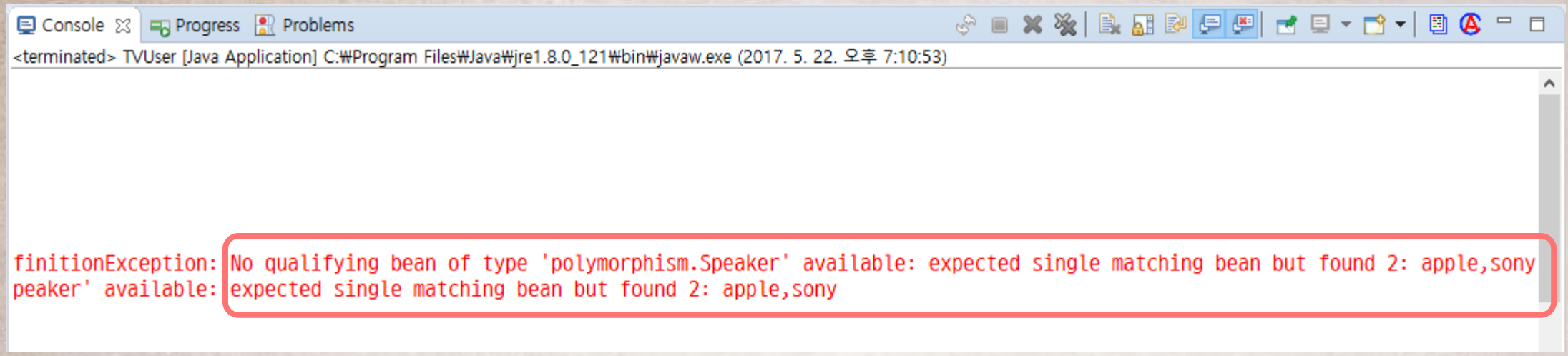
- 의존성 주입 대상이 되는 Speaker 타입 객체가 두개 이상일때는?
  - AppleSpeaker와 SonySpeaker객체가 모두 메모리에 있다면?
  - 컨테이너가 어떤 객체를 할당할지 판단할 수 없기 때문에 에러 발생
- AppleSpeaker에도 @Component 선언해보자.

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java

```java
package polymorphism;

import org.springframework.stereotype.Component;

@Component("apple")
public class AppleSpeaker implements Speaker {

    public AppleSpeaker(){
        System.out.println("==> AppleSpeaker 객체 생성");
    }

    @Override
    public void volumeUp() {
        System.out.println("AppleSpeaker -- 소리를 올린다.");
    }

    @Override
    public void volumeDown() {
        System.out.println("AppleSpeaker -- 소리를 내린다.");

    }
}
```

AppleSpeaker클래스에 대한 annotation추가

실행 결과



- @Autowired 대상이 되는 Speaker 타입의 객체가 AppleSpeaker,SonySpeaker로 두 개이고 둘다 메모리에 있으므로 의존성 주입에 문제 발생

해결 방법은 @Qualifier 어노테이션을 사용

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java

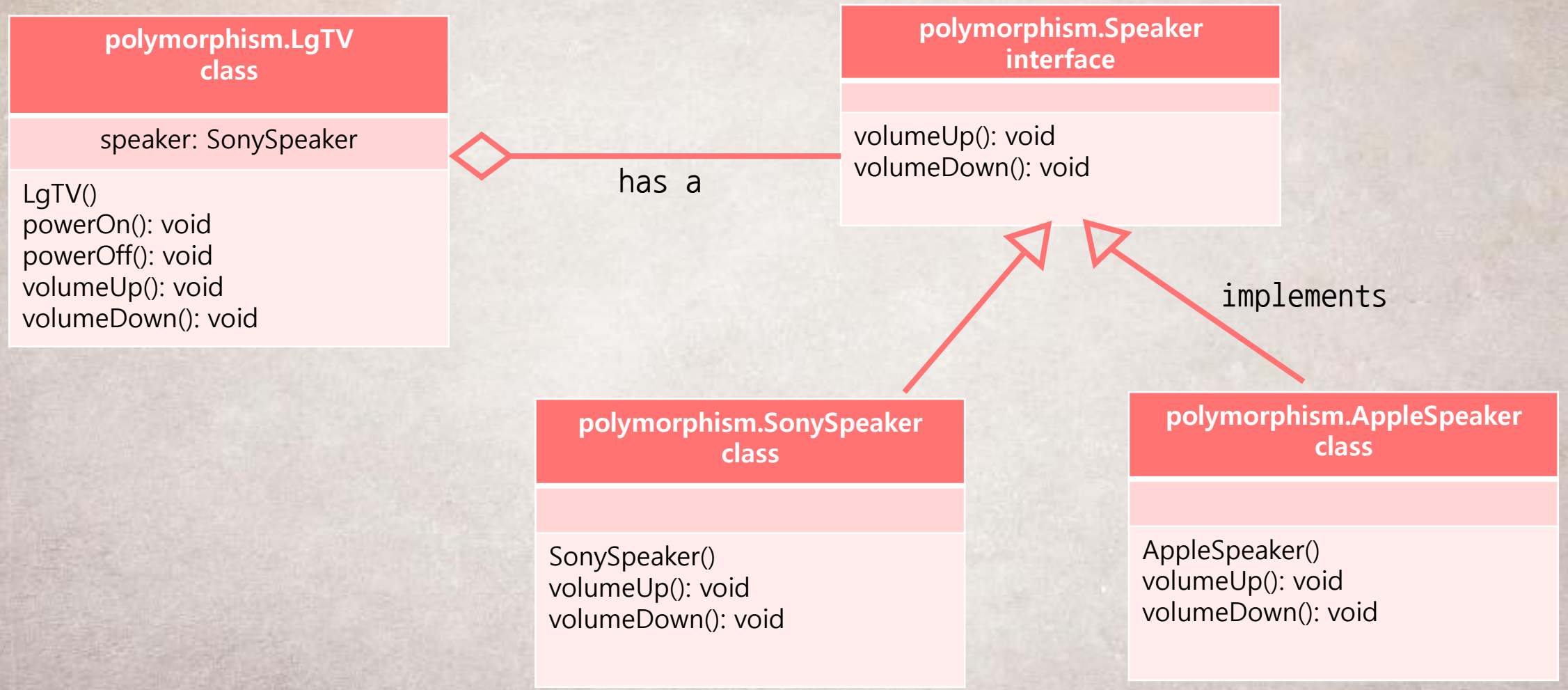
```java
package polymorphism;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Component;

@Component("tv")
public class LgTV implements TV {

    @Autowired
    @Qualifier("apple")
    private Speaker speaker;

    public LgTV(){
        System.out.println("===> LgTV 객체 생성");
    }
```

Console | Progress | Problems

<terminated> TVUser [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 22. 오후 7:45:39)

```
5월 22, 2017 7:45:40 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 22, 2017 7:45:40 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 19:45:
===> AppleSpeaker 객체 생성
===> LgTV 객체 생성
===> SonySpeaker 객체 생성
LgTV -- 전원을 켠다.
AppleSpeaker -- 소리를 올린다.
AppleSpeaker -- 소리를 내린다.
LgTV -- 전원을 끈다.
5월 22, 2017 7:45:40 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 19:45:40
```

TV 컴포넌트의 클래스 다이어그램



4. @Resource

- @Autowired는 변수의 타입을 기준으로 객체의 의존성 주입하지만
- @Resource는 객체의 이름을 이용하여 의존성 주입을 처리
  - name 속성을 사용하여 스프링 컨테이너가 해당 이름의 객체를 검색하여 의존성 주입 처리

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java

```
1  package polymorphism;
2
3  import javax.annotation.Resource;
4
5  import org.springframework.stereotype.Component;
6
7  @Component("tv")
8  public class LgTV implements TV {
9
10     @Resource(name="sony")
11     private Speaker speaker;
12
13     public LgTV(){
14         System.out.println("===> LgTV 객체 생성");
15     }
```

실행 결과

- @Resource와 같은 기능의 어노테이션으로 @Inject가 있고 둘 다 이름 기반의 의존성 주입을 처리한다.

5. 어노테이션과 XML 설정 병행하여 사용하기

- XML 설정 방식
  - XML 방식은 자바 소스를 수정하지 않고 XML 파일의 설정만 변경하면 실행되는 Speaker를 교체할 수 있어서 유지보수가 편함
  - XML 설정에 대한 부담이 역시 존재
  - 자바 소스에 의존관계와 관련된 메타데이터가 없으므로 XML 설정을 해석해야만 의존성 주입을 확인할 수 있다.

- 오노테이션 방식
  - XML 설정에 대한 부담이 없고,의존관계에 대한 정보가 자바 소스에 있어서 사용하기 편하다.
  - 의존성 주입할 객체의 이름이 자바소스에 명시되어야 하므로 자바소스를 수정하지 않고 Speaker를 교체할 수 없다는 문제가 발생

병행하여 사용하기

- 자동 객체 생성을 위한 @Component를 제거하고 사용할 객체를 스프링 설정 파일에 <bean> 등록하여 처리

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java

```java
package polymorphism;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Component;

@Component("tv")
public class LgTV implements TV {

    @Autowired
    private Speaker speaker;

    public LgTV(){
        System.out.println("===> LgTV 객체 생성");
    }
```

@Component를 제거

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java | applicationContext.xml

```java
package polymorphism;

public class SonySpeaker implements Speaker{

    public SonySpeaker(){
        System.out.println("===> SonySpeaker 객체 생성");
    }

    @Override
    public void volumeUp(){
        System.out.println("SonySpeaker -- 소리를 올린다.");
    }

    @Override
    public void volumeDown(){
        System.out.println("SonySpeaker -- 소리를 내린다.");
    }
}
```

LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java | *applicationContext.xml | AbstractApplicationContext.class

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="polymorphism"></context:component-scan>
    <bean class="polymorphism.SonySpeaker"></bean>
</beans>
```

AppleSpeaker 사용할 경우

```xml
<context:component-scan base-package="polymorphism"></context:component-scan>
<bean class="polymorphism.AppleSpeaker"></bean>
```

- 실행 결과

Console | Progress | Problems

```
<terminated> TVUser [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 22. 오후 8:35:55)
5월 22, 2017 8:35:55 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 22, 2017 8:35:55 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 20:35
===> LgTV 객체 생성
===> SonySpeaker 객체 생성
LgTV -- 전원을 켠다.
SonySpeaker -- 소리를 올린다.
SonySpeaker -- 소리를 내린다.
LgTV -- 전원을 끈다.
5월 22, 2017 8:35:56 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@378fd1ac: startup date [Mon May 22 20:35:55
```

병행사용하기

- XML 설정만 사용하기에는 <bean> 등록을 많이 해야 하고 의존관계 설정도 부담
- 어노테이션을 사용하자니 자바 소스를 수행해야 의존성 관계를 변경

➔ 변경되지 않는 객체는 어노테이션으로 설정하여 사용하고 변경될 가능성이 있는 객체는 XML 설정을 사용

➔ 우리가 직접 개발한 클래스는 XML, 어노테이션 둘 모두 사용 가능하나 라이브러리 형태로 제공하는 클래스는 XML 설정을 통해서만 사용 가능

  - Apache에서 제공하는 DB연동 처리를 할 경우에는 'commons-dbcp-1.5.jar' 파일을 사용해야 할 경우, XML에서 <bean> 등록을 해야 한다.

DataSource 설정 예

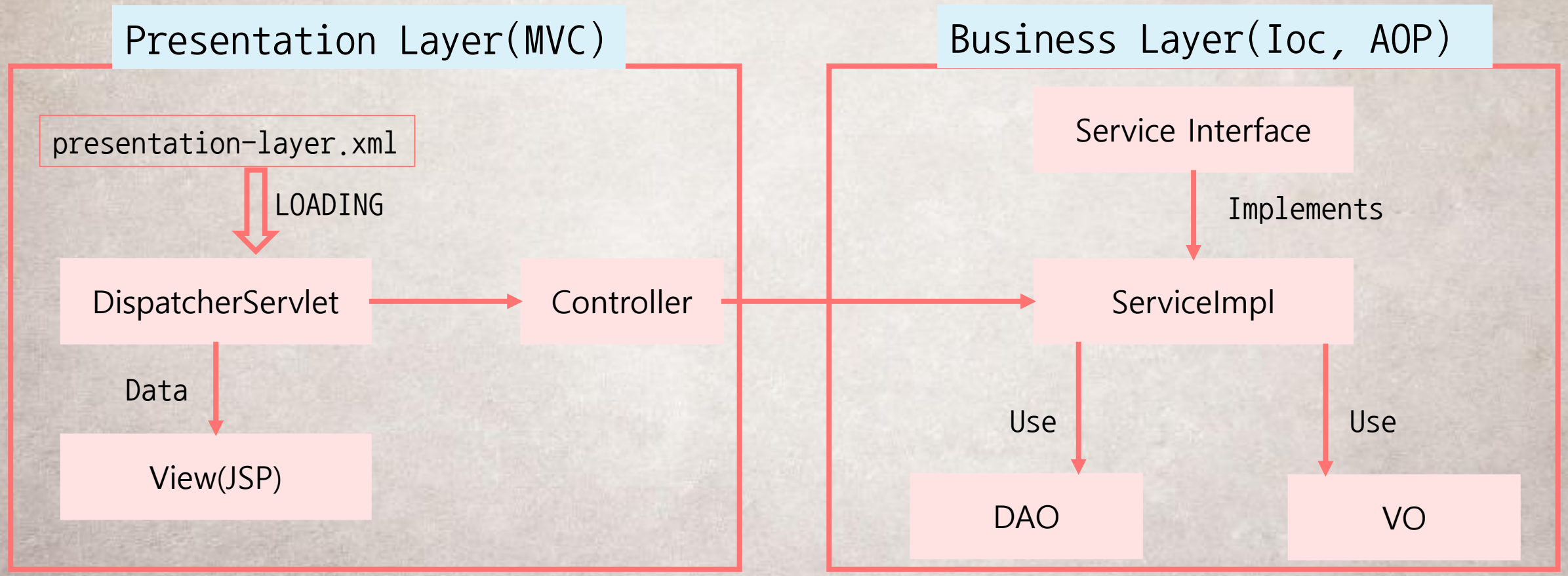
LgTV.java | TVUser.java | SonySpeaker.java | AppleSpeaker.java | applicationContext.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="polymorphism"></context:component-scan>
    <bean class="polymorphism.AppleSpeaker"></bean>

    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" destroy-method="close">
            <property name="driverClassname" value="org.h2.Driver"/>
            <property name="url" value="jdbc:h2:tcp://localhost/~/test"/>
            <property name="username" value="sa"/>
            <property name="password" value=""/>
    </bean>
</beans>

```

5.3 추가 어노테이션

시스템의 전체 구조를 두 개의 레이아웃으로 표현

- 프리젠테이션 레이어 - 사용자와의 커뮤니케이션 담당
- 비즈니스 레이어 - 사용자의 요청에 대한 비즈니스 로직 처리 담당



프리젠테이션과 비즈니스 레이어로 구성된 시스템 구조

시스템의 전체 구조를 두 개의 레이아웃으로 표현

- 가장 핵심 요소는 Controller, ServiceImpl, DAO 클래스
  - Controller - 사용자의 요청을 제어
  - ServiceImpl - 여러 개의 DAO요소를 사용한 실제적인 비즈니스 로직을 처리
  - DAO(Data Access Object) - 데이터 베이스 연동을 담당
- 시스템 구성요소들을 스프링 컨테이너에서 해당 클래스 객체를 생성할 경우 클래스의 역할을 파악하기 어렵다.

어노테이션	위치	의미
@Service	XXXServiceImpl	비즈니스 로직을 처리하는 Service 클래스
@Repository	XXXDAO	데이터베이스 연동을 처리하는 DAO 클래스
@Controller	XXXController	사용자 요청을 제어하는 Controller 클래스

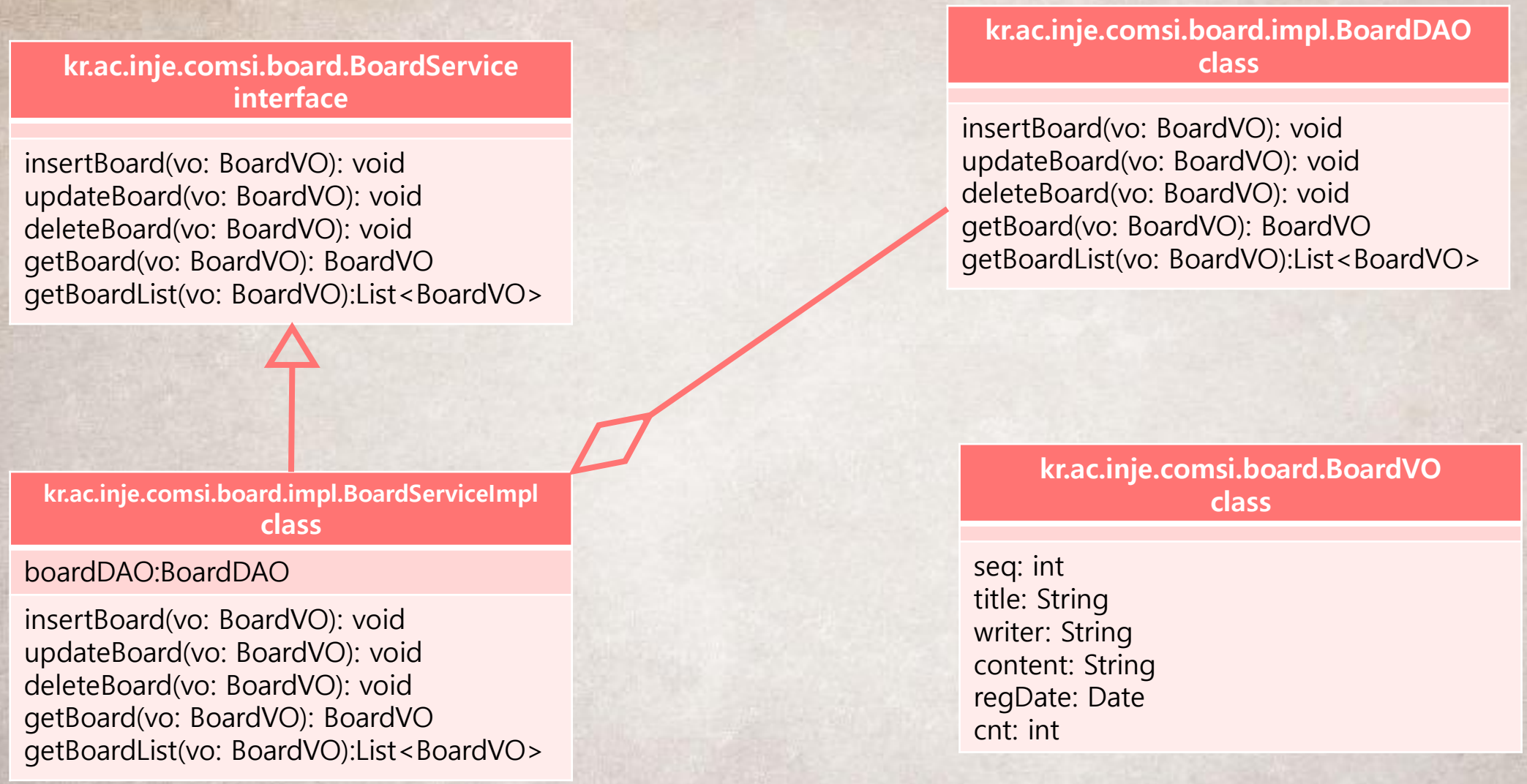
➔ 각 클래스를 해당 객체의 기능으로 인식하도록 함

6. 비즈니스 컴포넌트 실습1

6.1 BoardService 컴포넌트 구조

- 비즈니스 컴포넌트는 4개의 자바 파일로 구성
- 각 자바 파일을 작성하는 순서와 이름 규칙도 어느 정도 정해져 있음
- 게시판 관련 컴포넌트를 구현하면서 비즈니스 컴포넌트의 작성 순서와 이름 규칙을 살펴볼 것임

6.1 BoardService 컴포넌트 구조



- BoardService 컴포넌트 클래스 다이어그램

실습 프로젝트의 구조와 파일 위치

6.2 Value Object 클래스 작성

- VO(Value Object) 클래스는 레이어와 레이어 사이에서 관련된 데이터를 한꺼번에 주고 받을 목적으로 사용하는 클래스
- DTO(Data Transfer Object)라고도 함

게시판의 테이블 구조(board.sql)

```
board.sql
=========================================

CREATE TABLE BOARD(
        SEQ NUMBER(5) PRIMARY KEY,
        TITLE VARCHAR2(200),
        WRITER VARCHAR2(20),
        CONTENT VARCHAR2(2000),
        REGDATE DATE DEFAULT SYSDATE,
        CNT NUMBER(5) DEFAULT 0
);
```

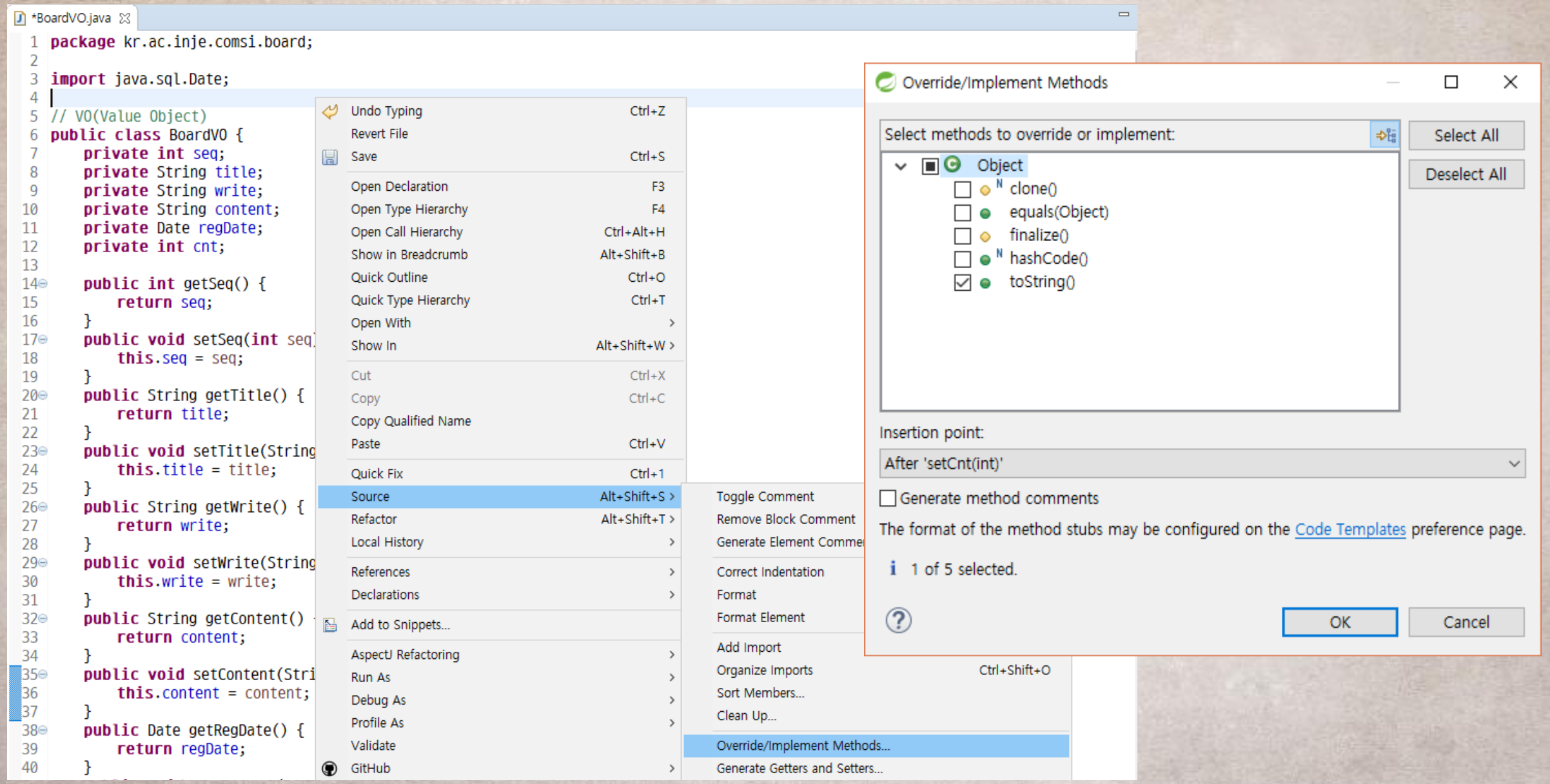
VO - BoardVO.java

BoardVO.java

```java
package kr.ac.inje.comsi.board;

import java.sql.Date;

// VO(Value Object)
public class BoardVO {

    private int seq;
    private String title;
    private String writer;
    private String content;
    private Date regDate;
    private int cnt;

    public int getSeq() {
        return seq;
    }
    public void setSeq(int seq) {
        this.seq = seq;
    }
    public String getTitle() {
        return title;
    }
    public void setTitle(String title) {
        this.title = title;
    }
    public String getWriter() {
        return writer;
    }
    public void setWriter(String writer) {
        this.writer = writer;
    }
    public String getContent() {
        return content;
    }
    public void setContent(String content) {
        this.content = content;
    }
    public Date getRegDate() {
        return regDate;
    }
    public void setRegDate(Date regDate) {
        this.regDate = regDate;
    }
    public int getCnt() {
        return cnt;
    }
    public void setCnt(int cnt) {
        this.cnt = cnt;
    }

    @Override
    public String toString() {
        return "BoardVO [seq=" + seq + ", title=" + title + ", writer="
                + writer + ", content=" + content + ", regDate=" + regDate
                + ", cnt=" + cnt + "]";
    }

}
```

다음 페이지

@Override toString() method



6.3 DAO 클래스 작성

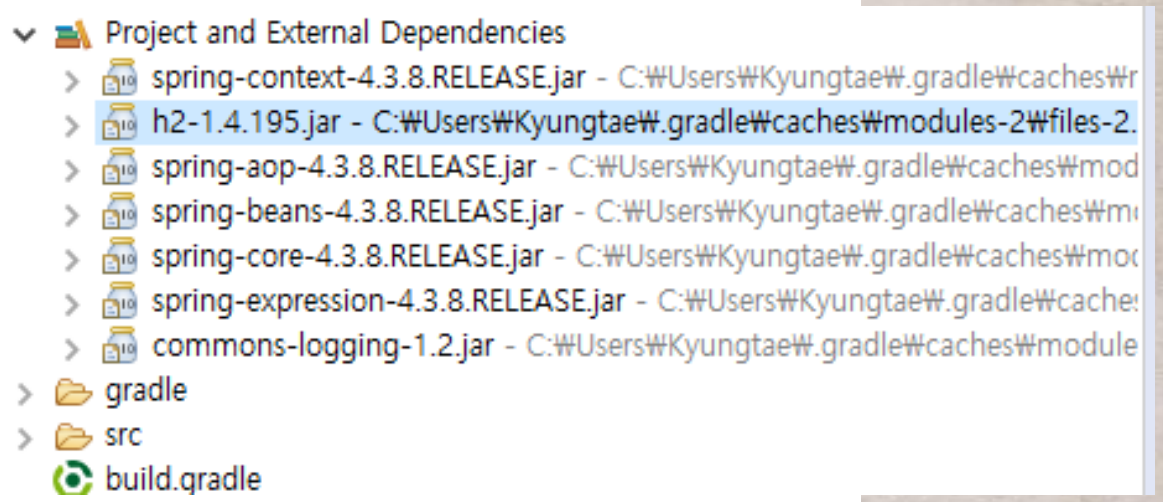
- DAO(Data Access Object)클래스는 데이터 베이스 연동을 담당하는 클래스
- DAO 클래스에서는 CRUD(Create, Read, Update, Delete) 기능의 메소드가 구현되어야 함

1. 드라이버 내려받기

- build.gradle 파일 수정
  - dependencies{ }에 내부에 Driver 정보 추가 후 Refresh Gradle Project

BoardVO.java | *build.gradle

```
1 /*
2  * This build file was generated by the Gradle 'init' task.
3  *
4  * This generated file contains a sample Java project to get
5  * For more details take a look at the Java Quickstart chapt
6  * user guide available at https://docs.gradle.org/3.3/userg
7  */
8
9 // Apply the java plugin to add support for Java
10 apply plugin: 'application'
11
12 mainClassName = System.getProperty("mainClass");
13
14 compileJava.options.encoding = 'UTF-8'
15
16 // In this section you declare where to find the dependencies of your project
17 repositories {
18     mavenCentral()
19 }
20
21 dependencies {
22     compile 'org.springframework:spring-context:4.3.8.RELEASE'
23     compile 'com.h2database:h2:1.4.195'
24 }
25
```



2. JDBC Utility 클래스 - Connection 획득과 해체 작업

BoardVO.java | JDBCUtil.java | BoardDAO.java

```java
package kr.ac.inje.comsi.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class JDBCUtil {
    // get connection
    public static Connection getConnection(){
        try{
            Class.forName("org.h2.Driver");
            return DriverManager.getConnection("jdbc:h2:tcp://localhost/~/test", "sa", "");
        }catch(Exception e){
            e.printStackTrace();
        }
        return null;
    }
    // close connection
    public static void close(PreparedStatement stmt, Connection conn){
        if(stmt != null){
            try{
                if(!stmt.isClosed()) stmt.close();
            }catch(Exception e){
                e.printStackTrace();
            }finally{
                stmt = null;
            }
        }

        if(conn != null){
            try{
                if(!conn.isClosed()) conn.close();
            }catch(Exception e){
                e.printStackTrace();
            }finally{
                conn = null;
            }
        }

    }
```

```
42      // close connection
43      public static void close(ResultSet rs, PreparedStatement stmt, Connection conn){
44          if(rs != null){
45              try{
46                  if(!rs.isClosed()) rs.close();
47              }catch(Exception e){
48                  e.printStackTrace();
49              }finally{
50                  rs = null;
51              }
52          }
53
54          if(stmt != null){
55              try{
56                  if(!stmt.isClosed()) stmt.close();
57              }catch(Exception e){
58                  e.printStackTrace();
59              }finally{
60                  stmt = null;
61              }
62          }
63
64          if(conn != null){
65              try{
66                  if(!conn.isClosed()) conn.close();
67              }catch(Exception e){
68                  e.printStackTrace();
69              }finally{
70                  conn = null;
71              }
72          }
73      }
74
75  }
```

3. DAO 클래스 작성-BOARD테이블과 CRUD 기능 처리

JDBCUtil.java | BoardDAO.java

```java
package kr.ac.inje.comsi.board.impl;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

import org.springframework.stereotype.Repository;

import kr.ac.inje.comsi.board.BoardVO;
import kr.ac.inje.comsi.common.JDBCUtil;

// DAO(Data Access Object)
@Repository("boardDAO")
public class BoardDAO {
    // JDBC 관련 변수
    private Connection conn = null;
    private PreparedStatement stmt = null;
    private ResultSet rset = null;

    // SQL 명령어
    private final String BOARD_INSERT = "insert into board(seq, title, writer, "
            + "content) values((select nvl(max(seq),0)+1 from board),?,?,?)";
    private final String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
    private final String BOARD_DELETE = "delete board where seq=?";
    private final String BOARD_GET = "select * from board where seq=?";
    private final String BOARD_LIST = "select * from board order by seq desc";

```

```
30      // CRUD 기능의 메소드 구현
31      // 글 등록
32      public void insertBoard(BoardVO vo){
33          System.out.println("===> JDBC로 insertBoard() 기능 처리");
34          try{
35              conn = JDBCUtil.getConnection();
36              stmt = conn.prepareStatement(BOARD_INSERT);
37              stmt.setString(1, vo.getTitle());
38              stmt.setString(2, vo.getWriter());
39              stmt.setString(3, vo.getContent());
40              stmt.executeUpdate();
41          }catch(Exception e){
42              e.printStackTrace();
43          }finally{
44              JDBCUtil.close(stmt, conn);
45          }
46      }
47
48      // 글 수정
49      public void updateBoard(BoardVO vo){
50          System.out.println("===> JDBC로 updateBoard() 기능 처리");
51          try{
52              conn = JDBCUtil.getConnection();
53              stmt = conn.prepareStatement(BOARD_UPDATE);
54              stmt.setString(1, vo.getTitle());
55              stmt.setString(2, vo.getContent());
56              stmt.setInt(3, vo.getSeq());
57              stmt.executeUpdate();
58          }catch(Exception e){
59              e.printStackTrace();
60          }finally{
61              JDBCUtil.close(stmt, conn);
62          }
63      }
64
```

```java

    // 글 삭제
    public void deleteBoard(BoardVO vo){
        System.out.println("===> JDBC로 deleteBoard() 기능 처리");
        try{
            conn = JDBCUtil.getConnection();
            stmt = conn.prepareStatement(BOARD_DELETE);
            stmt.setInt(1, vo.getSeq());
            stmt.executeUpdate();
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            JDBCUtil.close(stmt, conn);
        }
    }

    // 글 상세 조회
    public void getBoard(BoardVO vo){
        System.out.println("===> JDBC로 getBoard() 기능 처리");
        BoardVO board = null;
        try{
            conn = JDBCUtil.getConnection();
            stmt = conn.prepareStatement(BOARD_GET);
            stmt.setInt(1, vo.getSeq());
            rset = stmt.executeQuery();
            if(rset.next()){
                board = new BoardVO();
                board.setSeq(rset.getInt("SEQ"));
                board.setTitle(rset.getString("TITLE"));
                board.setWriter(rset.getString("WRITER"));
                board.setContent(rset.getString("CONTENT"));
                board.setRegDate(rset.getDate("REGDATE"));
                board.setCnt(rset.getInt("CNT"));
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            JDBCUtil.close(rset, stmt, conn);
        }
    }

```

```java
    // 글 목록 조회
    public List<BoardVO> getBoardList(BoardVO vo){
        System.out.println("===> JDBC로 getBoardList() 기능 처리");
        List<BoardVO> boardList = new ArrayList<BoardVO>();
        try{
            conn = JDBCUtil.getConnection();
            stmt = conn.prepareStatement(BOARD_LIST);
            rset = stmt.executeQuery();
            while(rset.next()){
                BoardVO board = new BoardVO();
                board.setSeq(rset.getInt("SEQ"));
                board.setTitle(rset.getString("TITLE"));
                board.setWriter(rset.getString("WRITER"));
                board.setContent(rset.getString("CONTENT"));
                board.setRegDate(rset.getDate("REGDATE"));
                board.setCnt(rset.getInt("CNT"));
                boardList.add(board);
            }
        }catch(Exception e){
            e.printStackTrace();
        }finally{
            JDBCUtil.close(rset, stmt, conn);
        }
        return boardList;
    }
}

```

BoardDAO 클래스

- 스프링 컨테이너가 생성할 수 있도록 @Repository 어노테이션 설정
  - @Component 를 사용해도 문제가 발생하지 않음
- DAO 기능의 클래스에는 @Component보다 @Repository가 적합
- CRUD 기능의 메소드 이름-일관성 유지

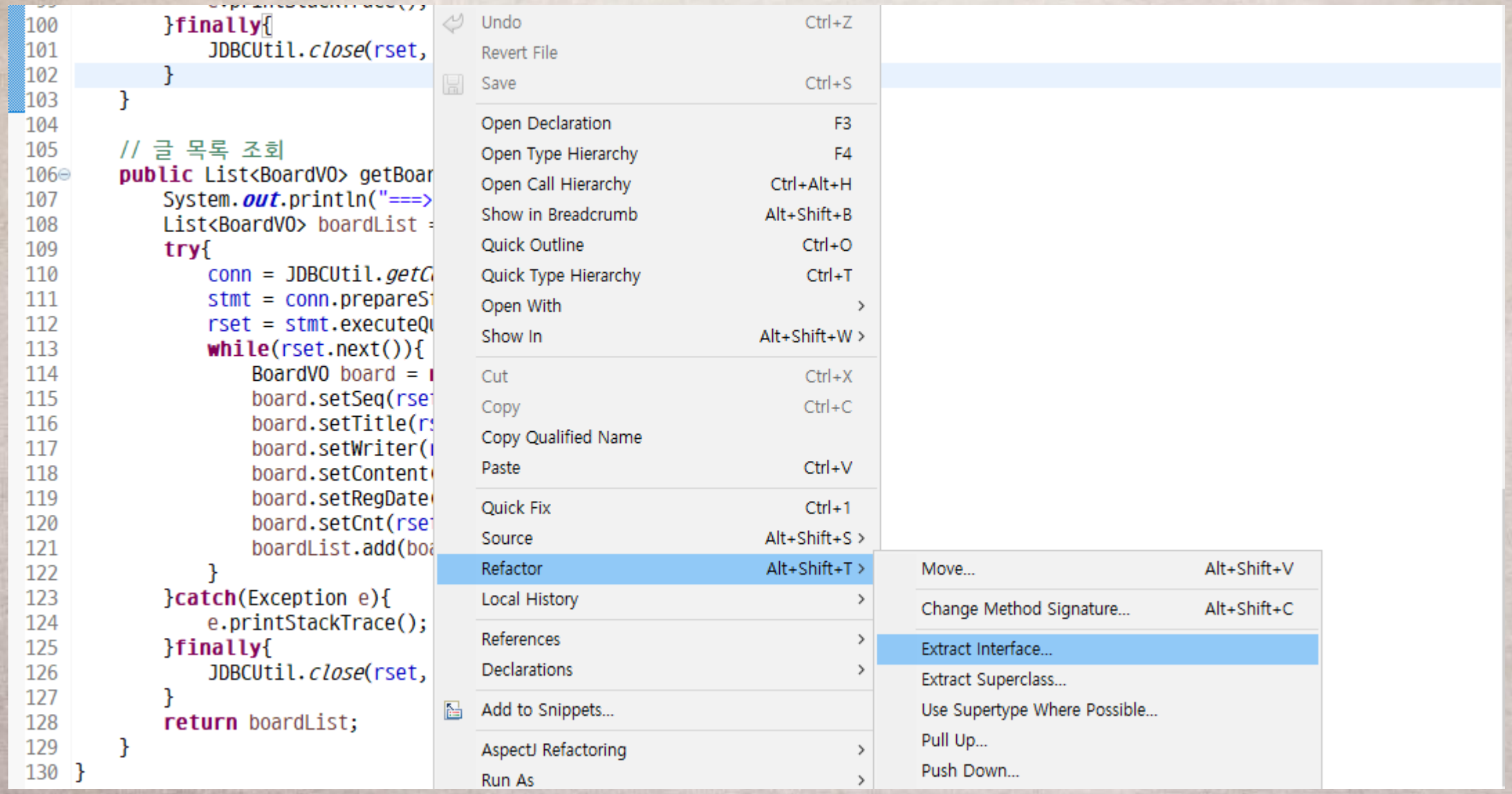
기능	메소드 이름
등록	insert테이블명
수정	update테이블명
삭제	delete테이블명
상세 조회	get테이블명 혹은 select테이블명
목록 검색	get테이블명List 또는 selec테이블명List

6.4 Service 인터페이스 작성

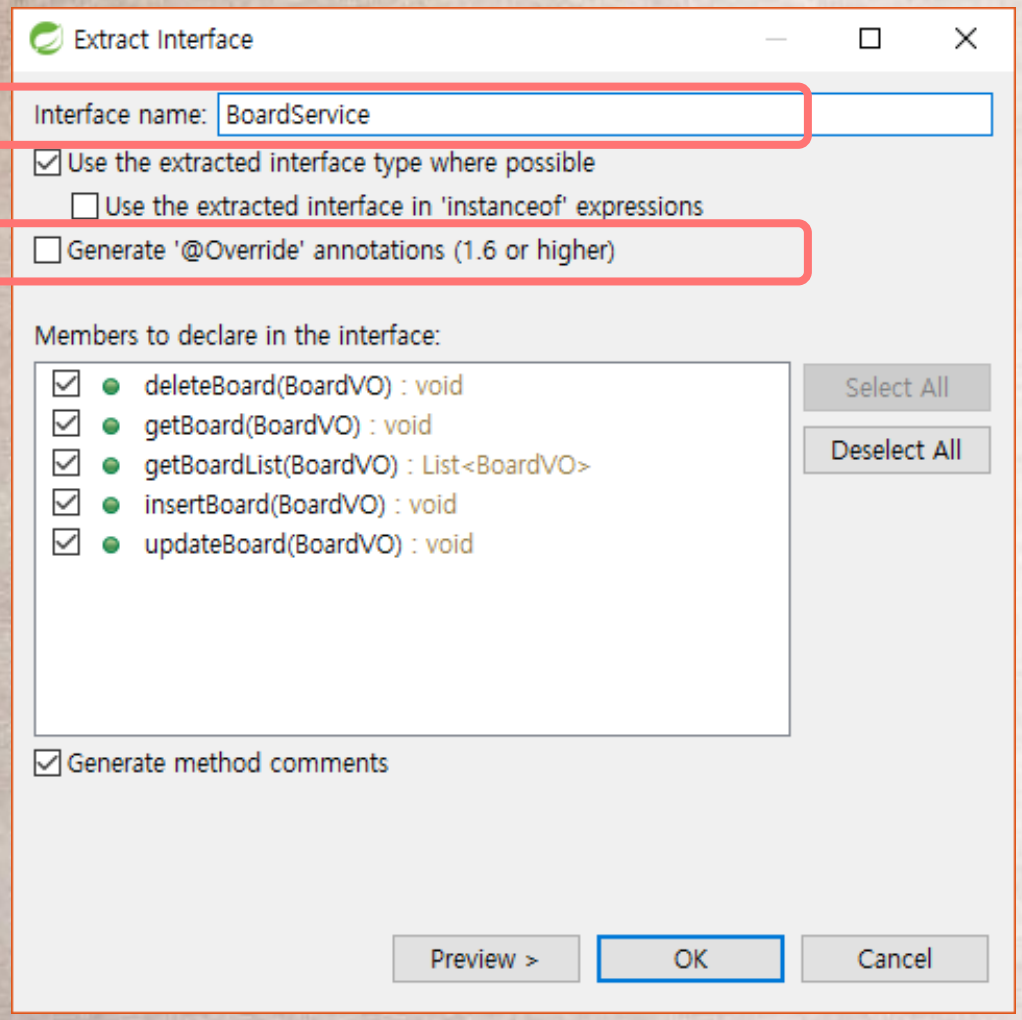
DAO 클래스에서 BoardService 인터페이스를 작성

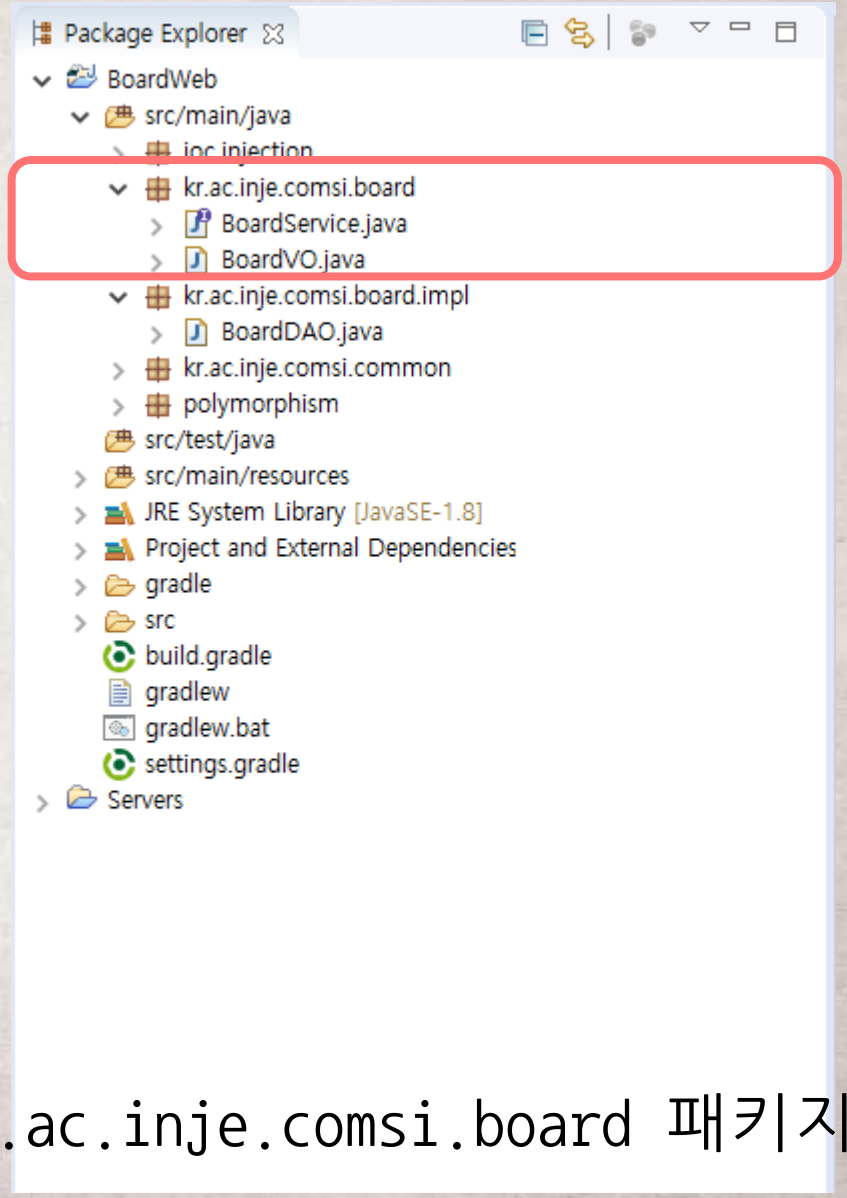
BoardService 인터페이스 작성

- BoardDAO 소스에 생성



BoardService 인터페이스 생성





kr.ac.inje.comsi.board 패키지로 이동

BoardService 인터페이스 생성

JDBCUtil.java | BoardDAO.java | BoardService.java

```java
package kr.ac.inje.comsi.board;

import java.util.List;

public interface BoardService {

    // CRUD 기능의 메소드 구현
    // 글 등록
    void insertBoard(BoardVO vo);

    // 글 수정
    void updateBoard(BoardVO vo);

    // 글 삭제
    void deleteBoard(BoardVO vo);

    // 글 상세 조회
    void getBoard(BoardVO vo);

    // 글 목록 조회
    List<BoardVO> getBoardList(BoardVO vo);

}
```

BoardDAO.java 소스 수정

```java
// DAO(Data Access Object)
@Repository("boardDAO")
public class BoardDAO implements BoardService {
    // JDBC 관련 변수
    private Connection conn = null;
    private PreparedStatement stmt = null;
    private ResultSet rset = null;

```

implements BoardService 삭제

6.5 Service 구현 클래스 작성

- BoardService 인터페이스를 구현한 BoardServiceImpl 클래스를 구현하면 비즈니스 컴포넌트는 마무리 됨
- BoardServiceImpl 클래스의 비즈니스 메소드를 구현할 때, 멤버변수로 선언된 BoardDAO를 이용함.

BoardServiceImpl.java 클래스 생성

BoardService.java | BoardServiceImpl.java

```java
package kr.ac.inje.comsi.board.impl;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import kr.ac.inje.comsi.board.BoardService;
import kr.ac.inje.comsi.board.BoardVO;

@Service("boardService")
public class BoardServiceImpl implements BoardService {

    @Autowired
    private BoardDAO boardDAO;

    @Override
    public void insertBoard(BoardVO vo) {
        boardDAO.insertBoard(vo);
    }

    @Override
    public void updateBoard(BoardVO vo) {
        boardDAO.updateBoard(vo);
    }

    @Override
    public void deleteBoard(BoardVO vo) {
        boardDAO.deleteBoard(vo);
    }
```

```
31
32      @Override
33      public void getBoard(BoardVO vo) {
34          boardDAO.getBoard(vo);
35      }
36
37      @Override
38      public List<BoardVO> getBoardList(BoardVO vo) {
39          return boardDAO.getBoardList(vo);
40      }
41
42  }
43
```

- BoardService 인터페이스의 모든 추상메소드를 Overriding 구현
- 클래스 선언부에 객체 생성을 위한 @Service가 선언
- 클라이언트 프로그램에서 boardService라는 이름으로 객체 요청
- BoardServiceImpl은 데이터베이스 연동을 포함된 비즈니스 로직 처리를 위해 BoardDAO 타입 객체를 멤버변수를 가짐
- BoardDAO 타입의 객체를 의존성 주입을 위해 @Autowired 설정

6.6 BoardService 컴포넌트 테스트

○

1. 스프링 설정 파일 수정

BoardService.java | BoardServiceImpl.java | applicationContext.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xmlns:context="http://www.springframework.org/schema/context"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        http://www.springframework.org/schema/context/spring-context-4.3.xsd">

    <context:component-scan base-package="kr.ac.inje.comsi"></context:component-scan>

</beans>
```

- 컴포넌트 스캔 범위를 ‘kr.ac.inje.comsi’로 지정
- BoardServiceImpl 클래스와 boardDAO 클래스가 스캔 범위에 포함되어 의존성 처리가 될 것임.

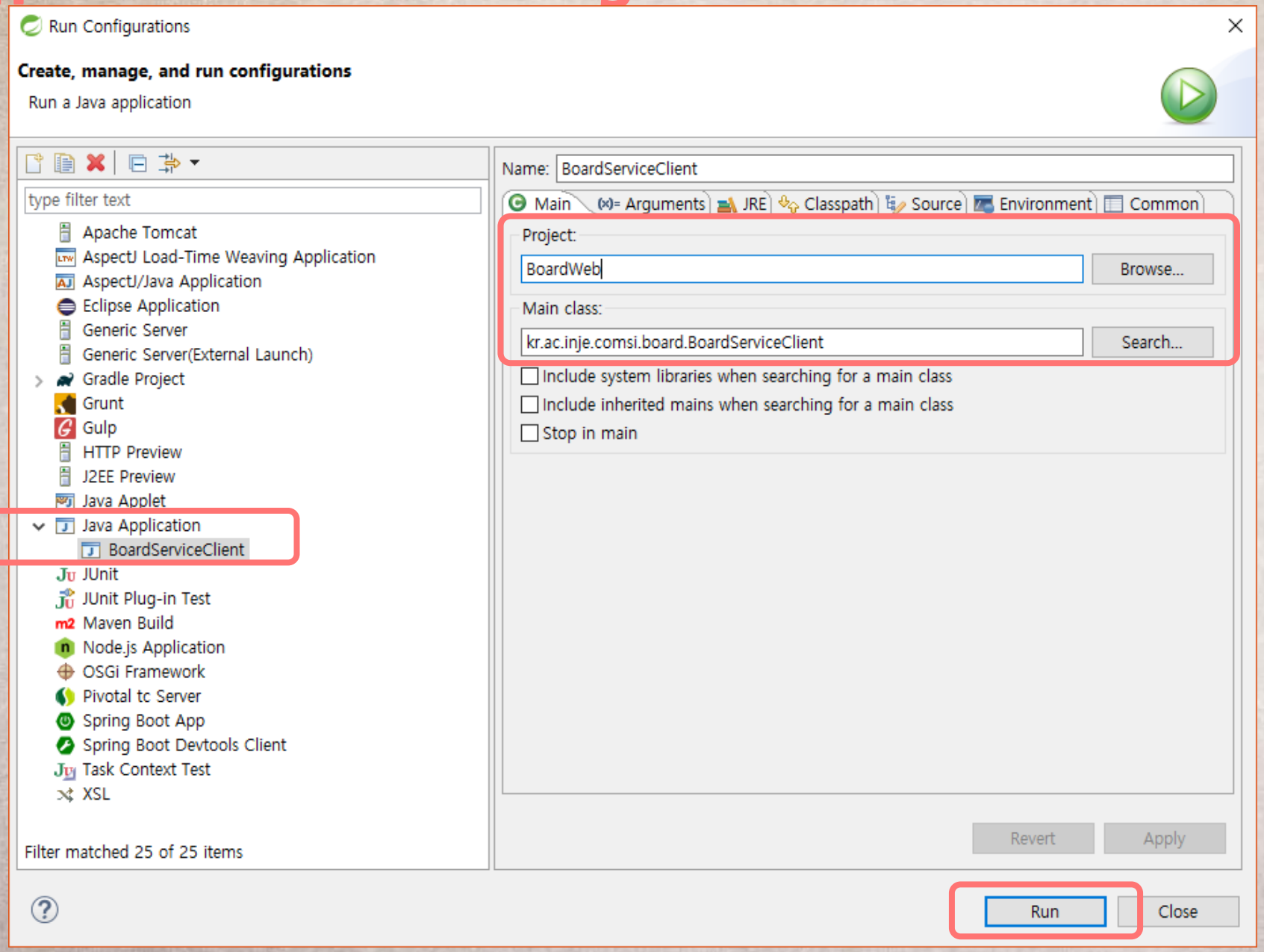
2. 클라이언트 작성 및 실행

BoardService.java | BoardServiceImpl.java | BoardServiceClient.java

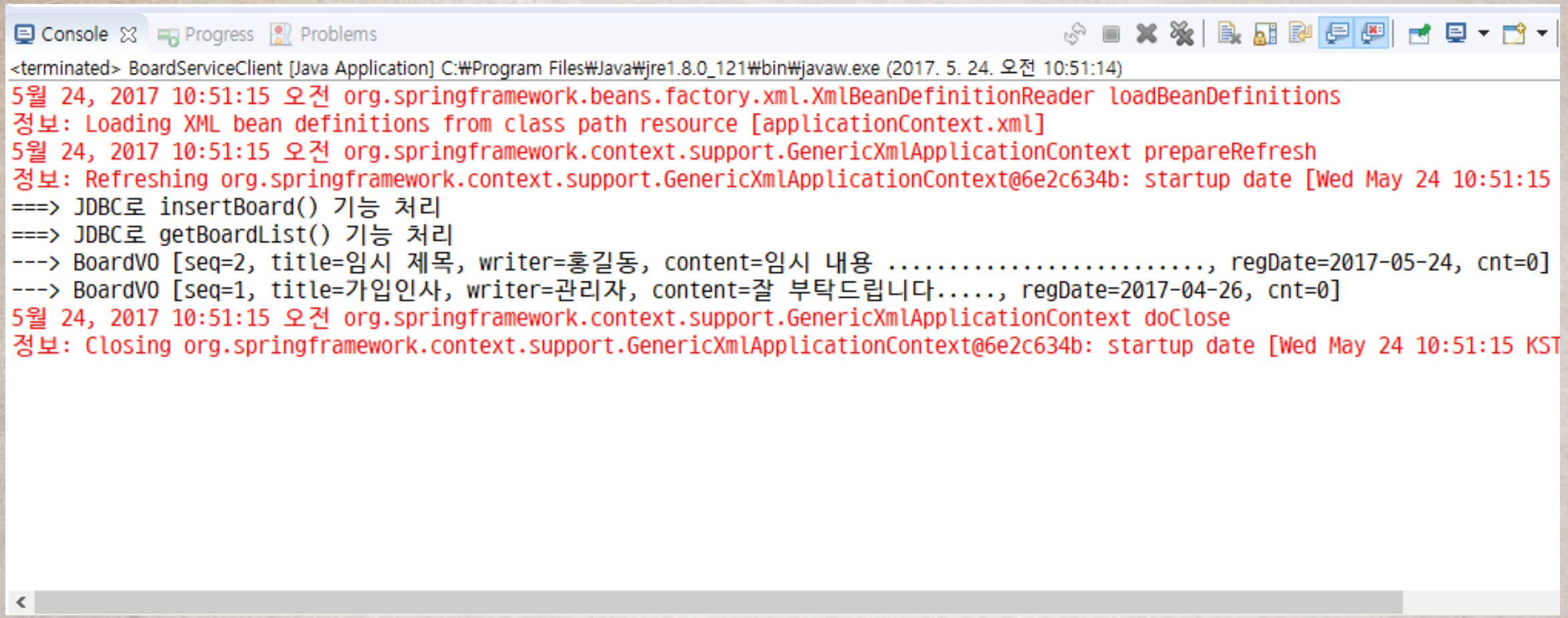
```java
package kr.ac.inje.comsi.board;

import java.util.List;

import org.springframework.context.support.AbstractApplicationContext;
import org.springframework.context.support.GenericXmlApplicationContext;

public class BoardServiceClient {

    public static void main(String[] args) {

        // 1. Spring 컨테이너를 구동
        AbstractApplicationContext  container = new GenericXmlApplicationContext("applicationContext.xml");

        // 2. Spring 컨테이너로부터 BoardServiceImpl 객체를 Lookup
        BoardService boardService = (BoardService) container.getBean("boardService");

        // 3. 글 등록 기능 테스트
        BoardVO vo = new BoardVO();
        vo.setTitle("임시 제목");
        vo.setWriter("홍길동");
        vo.setContent("임시 내용 ..........................");
        boardService.insertBoard(vo);

        // 4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList(vo);
        for(BoardVO board: boardList){
            System.out.println("---> " + board.toString());
        }

        // 5. Spring 컨테이너 종료
        container.close();
    }

}
```

실행

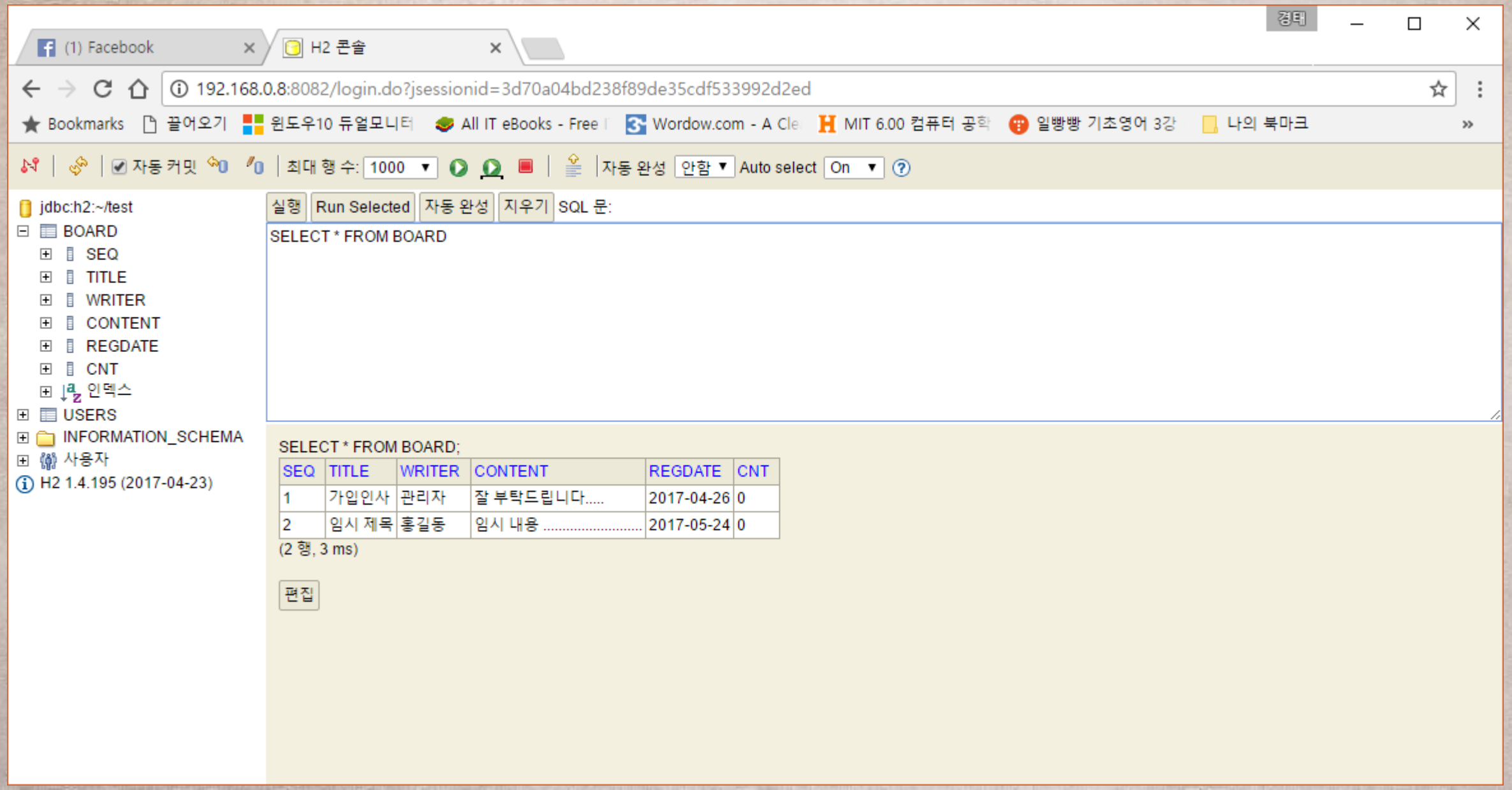
메뉴 Run→Run Configurations…



실행 결과

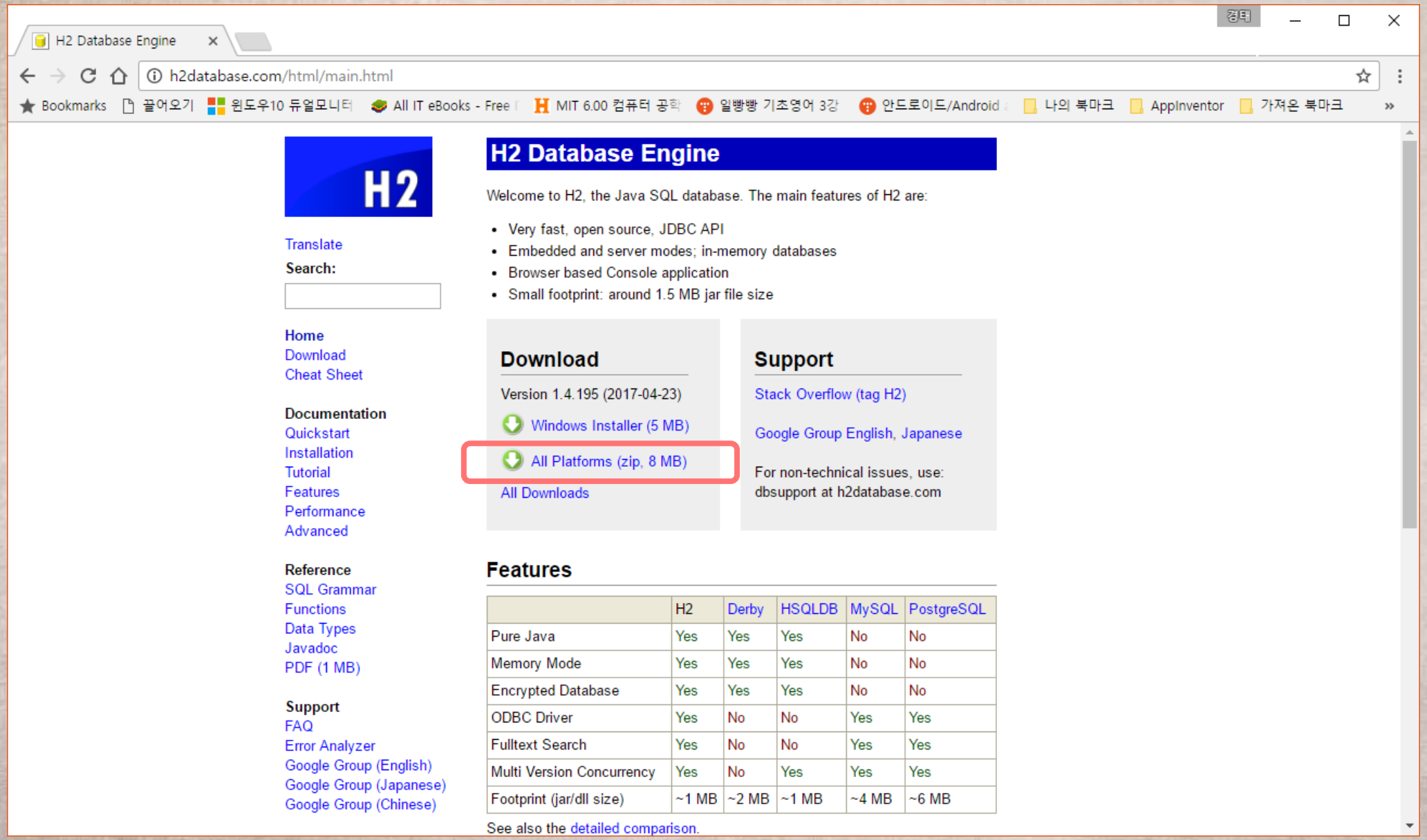

Console
Progress
Problems
<terminated> BoardServiceClient [Java Application] C:₩Program Files₩Java₩jre1.8.0_121₩bin₩javaw.exe (2017. 5. 24. 오전 10:51:14)
5월 24, 2017 10:51:15 오전 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 24, 2017 10:51:15 오전 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed May 24 10:51:15
===> JDBC로 insertBoard() 기능 처리
===> JDBC로 getBoardList() 기능 처리
---> BoardVO [seq=2, title=임시 제목, writer=홍길동, content=임시 내용 ....., regDate=2017-05-24, cnt=0]
---> BoardVO [seq=1, title=가입인사, writer=관리자, content=잘 부탁드립니다....., regDate=2017-04-26, cnt=0]
5월 24, 2017 10:51:15 오전 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Wed May 24 10:51:15 KST

실행 결과 - H2Database

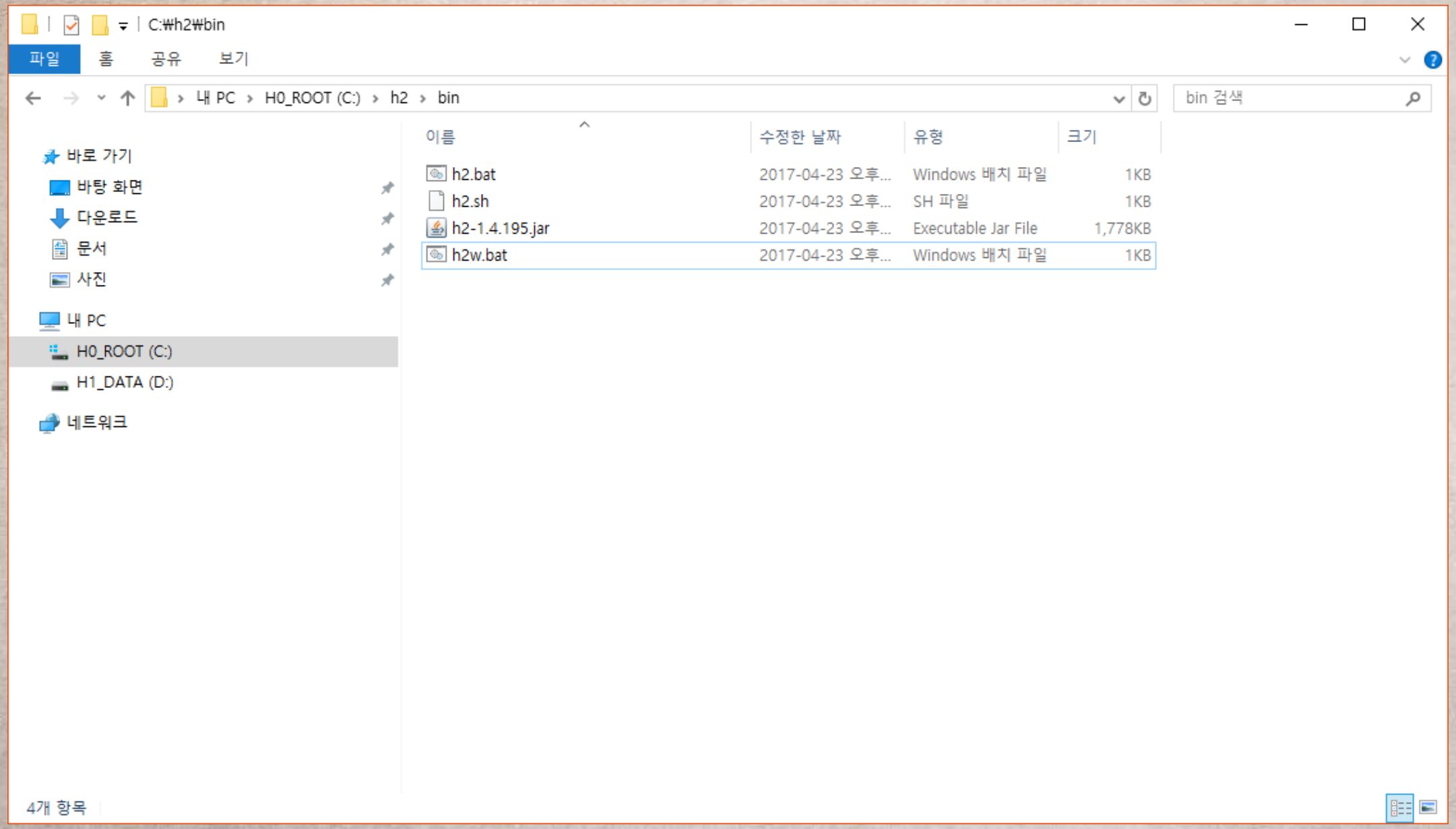


데이터베이스 구축

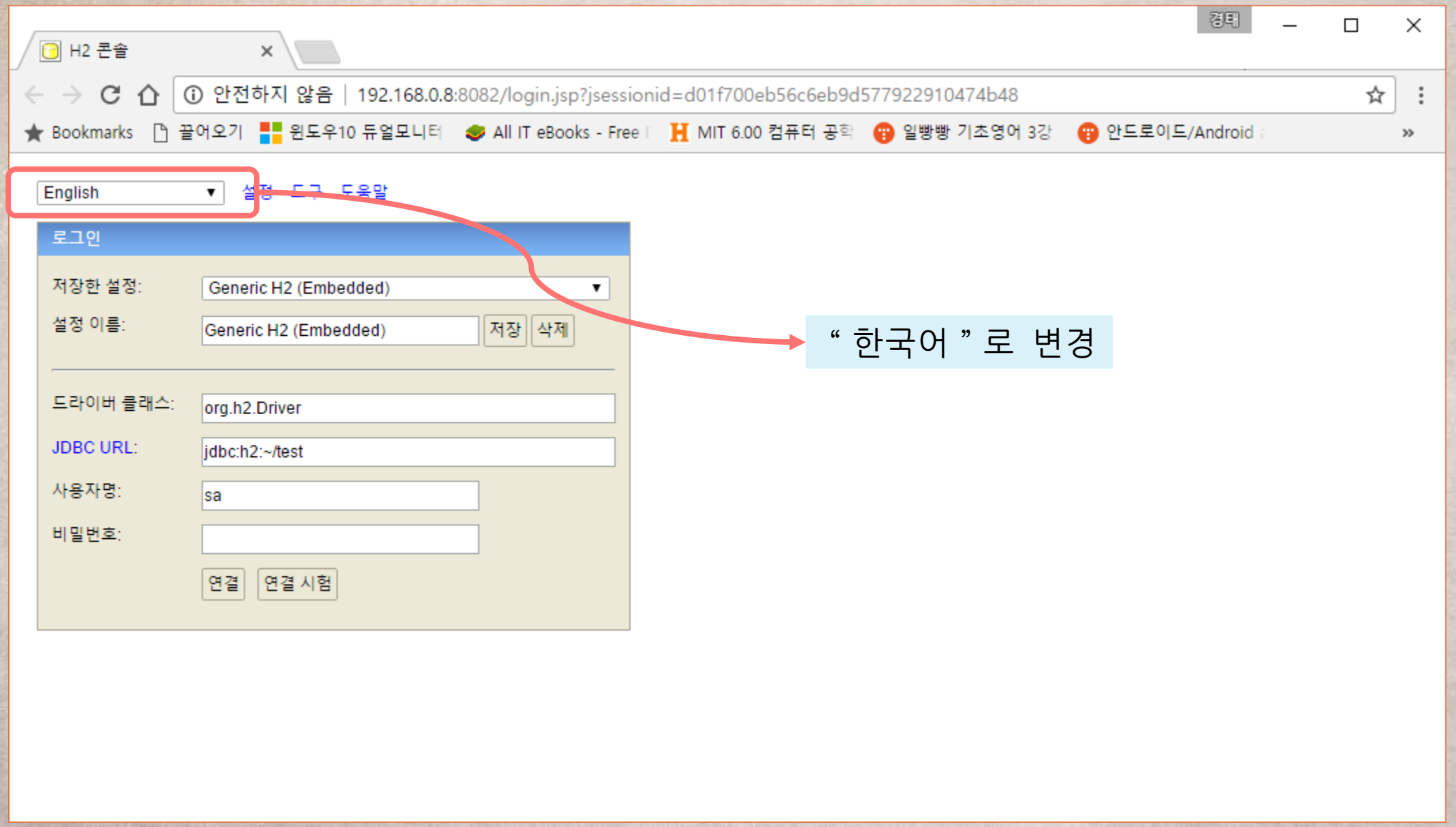
H2-http://h2database.com



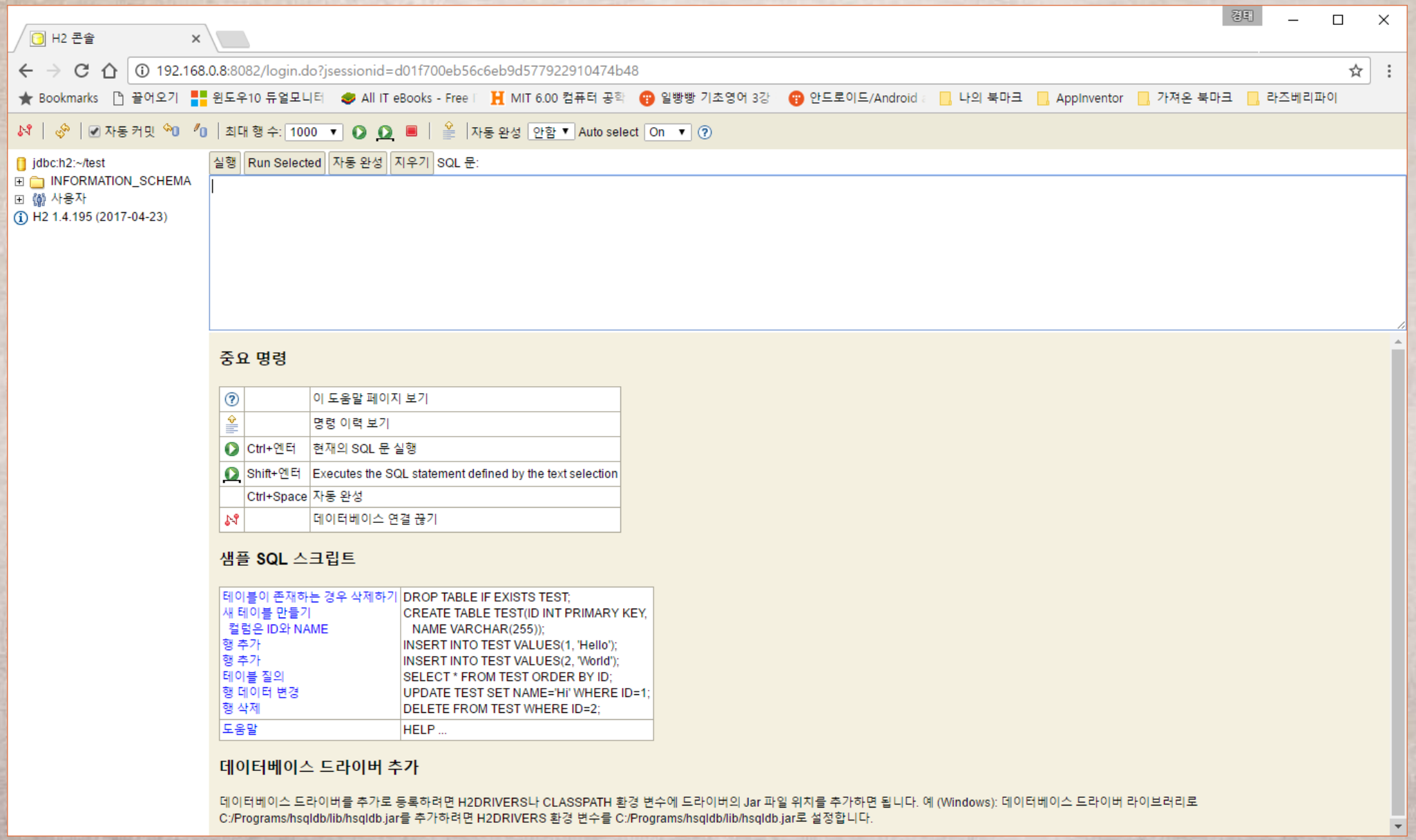
H2Database 실행 - h2→bin>h2w.bat 실행



H2 서버 - 구동


H2 콘솔
안전하지 않음 | 192.168.0.8:8082/login.jsp?jsessionid=d01f700eb56c6eb9d577922910474b48
Bookmarks
English
로그인
저장한 설정:
Generic H2 (Embedded)
설정 이름:
Generic H2 (Embedded)
저장
삭제
드라이버 클래스:
org.h2.Driver
JDBC URL:
jdbc:h2:~/test
사용자명:
sa
비밀번호:
연결
연결 시험
“ 한국어 ” 로 변경

H2 서버 - 연결


H2 콘솔
192.168.0.8:8082/login.do?jsessionid=d01f700eb56c6eb9d577922910474b48
Bookmarks
끌어오기
윈도우10 듀얼모니터
All IT eBooks - Free
MIT 6.00 컴퓨터 공학
일빵빵 기초영어 3강
안드로이드/Android
나의 북마크
AppInventor
가져온 북마크
라즈베리파이
자동 커밋
최대 행 수: 1000
자동 완성
안함
Auto select
On
jdbc:h2:~/test
INFORMATION_SCHEMA
사용자
H2 1.4.195 (2017-04-23)
실행
Run Selected
자동 완성
지우기
SQL 문:
중요 명령
이 도움말 페이지 보기
명령 이력 보기
Ctrl+엔터
현재의 SQL 문 실행
Shift+엔터
Executes the SQL statement defined by the text selection
Ctrl+Space
자동 완성
데이터베이스 연결 끊기
샘플 SQL 스크립트
테이블이 존재하는 경우 삭제하기
DROP TABLE IF EXISTS TEST;
새 테이블 만들기
컬럼은 ID와 NAME
CREATE TABLE TEST(ID INT PRIMARY KEY,
NAME VARCHAR(255));
행 추가
INSERT INTO TEST VALUES(1, 'Hello');
행 추가
INSERT INTO TEST VALUES(2, 'World');
테이블 질의
SELECT * FROM TEST ORDER BY ID;
행 데이터 변경
UPDATE TEST SET NAME='Hi' WHERE ID=1;
행 삭제
DELETE FROM TEST WHERE ID=2;
도움말
HELP ...
데이터베이스 드라이버 추가
데이터베이스 드라이버를 추가로 등록하려면 H2DRIVERS나 CLASSPATH 환경 변수에 드라이버의 Jar 파일 위치를 추가하면 됩니다. 예 (Windows): 데이터베이스 드라이버 라이브러리로 C:/Programs/hsqldb/lib/hsqldb.jar를 추가하려면 H2DRIVERS 환경 변수를 C:/Programs/hsqldb/lib/hsqldb.jar로 설정합니다.

H2 서버 - 테이블 생성 및 데이터 입력코드(sql문)

USER 테이블

```
CREATE TABLE USERS(
        ID VARCHAR2(8) PRIMARY KEY,
        PASSWORD VARCHAR2(8),
        NAME VARCHAR2(20),
        ROLE VARCHAR2(5)
);

INSERT INTO USERS VALUES('test','test123','관리자','Admin');
INSERT INTO USERS VALUES('user1','user1','홍길동','user');
```

BOARD 테이블

```
CREATE TABLE BOARD(
        SEQ NUMBER(5) PRIMARY KEY,
        TITLE VARCHAR2(200),
        WRITER VARCHAR2(20),
        CONTENT VARCHAR2(2000),
        REGDATE DATE DEFAULT SYSDATE,
        CNT NUMBER(5) DEFAULT 0
);

INSERT INTO BOARD(SEQ, TITLE, WRITER, CONTENT) VALUES(1,'가입인사','관리자','잘 부탁드립니다.....');
```

H2 서버 - 입력 데이터 확인