

0B장 스프링 퀵 스타트 - 첫째 날 (IoC)

B.1 스프링 프레임워크 시작하기

B.1.1 개발 환경 구축

(1) JDK 설치

- JDK는 1.8 버전(jdk-8u111-windows-x64.exe)을 설치한다.

(2) 이클립스와 STS Plugin 설치

- 이클립스는 자바 언어로 개발된 툴로 JDK 필요하다.
- 다운로드 사이트: <https://www.eclipse.org/downloads/>
 - Eclipse IDE for Java EE Developers Version 2020-09
 - 웹 애플리케이션 등의 Enterprise (Network) 환경에서 실행
 - CPU 사양에 맞게 다운로드 (eclipse-inst-jre-win64.exe)
- Preferences 설정
 - 에디터의 행 번호 표기하기: Eclipse > Windows > Preferences > General/Editors/Text Editors > "Show line numbers"를 체크
 - 에디터의 인코딩 설정: Eclipse > Windows > Preferences > General/Workspace > Text file encoding > UTF-8 로 선택
- STS(Spring Tool Suite) 플로그인 설치
 - 이클립스에 STS 플러그인을 설치하면 편하고 쉽게 스프링 관련 실습을 수행할 수 있다.
 - 이클립스의 'Help > Eclipse Marketplace' 메뉴를 선택하여 'Spring Tools'이라는 키워드로 검색하여 "Spring Tools 3 (Standalone Edition) 3.9.14.RELEASE"에서 <Install> 버튼을 클릭하여 설치한다.

[꿀팁] STS 설치 (선택사항)

- STS(Spring Tool Suite)는 스프링 개발을 편하게 하기 위한 개발 툴, 전체적인 모습은 이클립스와 동일하다.
- 다운로드 사이트: <https://github.com/spring-projects/toolsuite-distribution/wiki/Spring-Tool-Suite-3>
- 스프링 개발을 위해 STS 3.9.1 버전(spring-tool-suite-3.9.1.RELEASE-e4.7.1a-win32-x86_64.zip)을 다운로드를 받아 설치한다.

(3) 톰캣 서버 설치와 STS 연동

- 톰캣(Tomcat) 서버는 아파치 홈페이지(<http://tomcat.apache.org/>)에서 내려받을 수 있다.
- 실습에 사용할 아파치 톰캣 파일(apache-tomcat-9.0.41-windows-x64.zip)을 내려 받는다.

- STS를 실행하여 [Servers] 뷰에 있는 링크를 클릭하여 톰캣 서버를 등록한다.

(4) 데이터베이스 구축

- H2 데이터베이스
 - 실습에 사용할 DBMS는 별도의 설치 과정도 필요없고 용량도 1.7M 정도로 가벼운 H2 데이터베이스이다.
 - H2 데이터베이스 홈페이지(<http://www.h2database.com/html/download.html>)에서 Version 1.4.199 인 Window Installer(h2-setup-2019-03-13.exe) 버전으로 설치한다.
 - 설치후 윈도우 메뉴 > 모든 프로그램 > H2 > H2 Console 로 접속한다.
 - 접속후 SQL을 입력하는 창에 다음과 같이 실습에 필요한 테이블 생성코드와 입력 코드를 작성하여 실행한다.

```
// WEB-INT/sql/ddl.sql

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');

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');

SELECT * FROM BOARD;
SELECT * FROM USERS;
```

B.1.2 실습 프로젝트 생성

(1) 프로젝트 생성

- 이클립스의 'File > New' 메뉴로 이동한 후에 'Spring Legacy Project'를 선택한다.
- 프로젝트 설정을 다음과 같다.
 - Project name: BoardWeb
 - Templates: 'Spring MVC Project'를 선택
 - Top-level package: com.springbook.biz

(2) 프로젝트 설정 변경

- BoardWebLab 프로젝트를 선택한 상태에서 마우스 오른쪽 버튼을 클릭하여 맨 아래에 'Properties'를 선택한다.
- 'Project Facets'를 선택하여 Java 버전을 우리가 설치한 1.8 버전으로 수정한다. 그리고 오른쪽에 'Runtimes' 탭을 선택하여 'Apache Tomcat v9.0'을 체크한다.
- 'Java Buil Path'를 클릭하고 'Libaries' 탭을 클릭하여 'Apache Tomcat v9.0 [Apache Tomcat v9.0]'과 JRE System Library [jre]' 설정이 개발 환경에 맞는지 확인한다.
- 마지막으로 pom.xml 파일의 Java 버전과 Spring 버전을 변경한다.

// pom.xml의 일부

```
<properties>
    <java-version>1.8</java-version>
    <org.springframework-version>4.2.4.RELEASE</org.springframework-version>
    <org.aspectj-version>1.6.10</org.aspectj-version>
    <org.slf4j-version>1.6.6</org.slf4j-version>
</properties>
```

[꿀팁] pom.xml의 에러 문제과 해결

- 문제: 프로젝트 생성 직후 다음과 같이 pom.xml에 에러가 발생한다.

// pom.xml의 에러

```
Multiple annotations found at this line:
  - CoreException: Could not get the value for parameter compilerId for plugin execution default-compile: PluginResolutionException: Plugin
    org.apache.maven.plugins:maven-compiler-plugin:2.5.1 or one of its dependencies could not be resolved: Failed to collect dependencies at
    org.apache.maven.plugins:maven-compiler-plugin:jar:2.5.1 -> org.apache.maven:maven-core:jar:2.0.9:
ArtifactDescriptorException: Failed to read artifact
  descriptor for org.apache.maven:maven-core:jar:2.0.9: ArtifactResolutionException: Failure to transfer org.apache.maven:maven-core:pom:2.0.9 from https://
    repo.maven.apache.org/maven2 was cached in the local repository, resolution will not be reattempted until the update interval of central has elapsed or
    updates are forced. Original error: Could not transfer artifact org.apache.maven:maven-core:pom:2.0.9 from/to central (https://repo.maven.apache.org/
    maven2): connect timed out
  - Plugin execution not covered by lifecycle configuration: org.apache.maven.plugins:maven-compiler-plugin:2.5.1:compile (execution: default-compile,
    phase: compile)
```

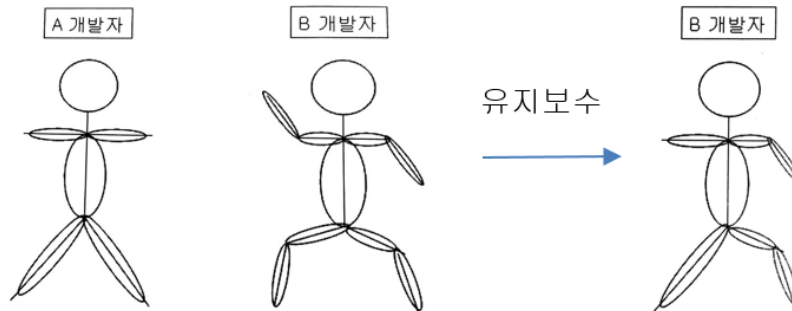
- 해결
 1. 프로젝트 우클릭 > Run As > Maven Install
 2. 이클립스 프로젝트 탐색기에서 해당 프로젝트 클릭 후 F5(새로고침)
 3. 프로젝트 우클릭 > Maven > Update Project

B.2 프레임워크 개요

B.2.1 프레임워크 개념

(1) 프레임워크의 등장 배경

- 프레임워크의 사전적 의미는 뼈대 혹은 틀로서 이 의미를 소프트웨어 관점에서 접근하면 아키텍처에 해당하는 골격 코드이다.
- 개발자에게 모든 것을 위임하는 것이 아니라 애플리케이션의 기본 아키텍처는 프레임워크가 제공하고, 그 뼈대에 살을 붙이는 작업만 개발자가 하는 것이다. → 따라서 유지보수가 쉽다.
- 예를들어 두 명의 개발자에게 인형을 만들도록 지시하고 모든 권한을 각 개발자에게 위임하여 자유롭게 만들도록 하게되면 각 개발자는 다음과 같이 자신이 가진 경험과 기술을 활용하여 각각 인형을 만들 것이다. --> 고객은 수리 과정에서 처음에 구매한 인형과 다른 모습으로 변형된 인형을 보면서 불만을 제기할 것이다



[그림] 유지보수의 어려움

(2) 프레임워크의 장점

- 빠른 구현 시간: 프레임워크를 사용하면 아키텍처에 해당하는 골격 코드를 프레임워크에서 제공한다. 따라서 개발자는 비즈니스 로직만 구현하면 되므로 제한된 시간에 많은 기능을 구현할 수 있다.
- 쉬운 관리: 같은 프레임워크가 적용된 애플리케이션들은 아키텍처가 같으므로 관리하기가 쉽다. 결과적으로 유지보수에 들어가는 인력과 시간도 줄일 수 있다.
- 개발자들의 역량 획일화: 프레임워크를 사용하면 숙련된 개발자와 초급 개발자가 생성한 코드가 비슷해진다. 결과적으로 관리자 입장에서 개발 인력을 더 효율적으로 구성할 수 있다.
- 검증된 아키텍처의 재사용과 일관성 유지: 아키텍처에 관한 별다른 고민이나 검증 없이 소프트웨어를 개발할 수 있다.

(3) 자바 기반의 프레임워크

처리 영역	프레임워크	설명
Presentation (화면)	Struts	UI Layer에 중점을 두고 개발된 MVC 프레임워크이다.
	Spring	Struts와 동일하게 MVC 아키텍처를 제공하는 UI Layer 프레임워크이다. 하

	(MVC)	지만 Struts처럼 독립된 프레임워크는 아니고 Spring 프레임워크에 포함되어 있다.
Business (로직)	Spring (IoC, AOP)	Spring은 컨테이너 성격을 가지는 프레임워크이다. Spring의 IoC의 AOP 모듈을 이용하여 Spring 컨테이너에서 동작하는 엔터프라이즈 비즈니스 컴포넌트를 개발할 수 있다.
Persistence (데이터베이스)	Hibernate or JPA	Hibernate는 완벽한 ORM(Object Relation Mapping) 프레임워크이다. JPA는 Hibernate를 비롯한 모든 ORM의 공통 인터페이스를 제공하는 자바 표준 API이다.
	Ibatis or Mybatis	Ibatis 프레임워크는 개발자가 작성한 SQL 명령어와 자바 객체(VO 혹은 DTO)를 매핑해주는 기능을 제공한다. Mybatis는 Ibatis에서 파생된 프레임워크로서 기본 개념과 문법은 거의 같다.

(4) 스프링 프레임워크가 개발의 대세가 된 이유

- 복잡함(Enterprise JavaBeans 기반의 개발)에 반기를 들어서 만들어진 프레임워크
- 프로젝트의 전체 구조를 설계할 때 유용한 프레임워크
- 다른 프레임워크들의 포용
- 개발 생산성과 개발 도구의 지원

B.2.2 스프링 프레임워크의 특징

(1) 경량(Lightweight)

- 몇 개의 JAR 파일만 있으면 개발과 실행이 모두 가능하다. 따라서 스프링을 이용해서 만든 애플리케이션의 배포 역시 매우 빠르고 쉽다.
- POJO(Plain Old Java Object) 형태의 객체를 관리한다. POJO는 클래스를 구현하는 데 특별한 규칙이 없는 단순하고 가벼운 객체이므로 POJO를 관리하는 것은 기존의 EJB((Enterprise Java Bean) 객체를 관리하는 것보다 훨씬 가볍고 빠르다.

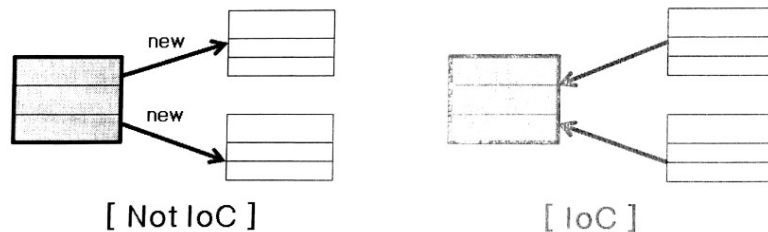
[꿀팁] Servlet 클래스 = Not POJO

- Servlet 클래스는 우리 마음대로 만들 수 없으며, 반드시 Servlet에서 요구하는 규칙에 맞게 클래스를 만들어야 한다.
- 다음은 Servlet 클래스 작성 규칙이다.
 - javax.servlet, javax.servlet.http 패키지를 import해야 한다.
 - public 클래스로 선언되어야 한다.
 - Servlet, GenericServlet, HttpServlet 중 하나를 상속해야 한다.
 - 기본 생성자(Default Constructor)가 있어야 한다.
 - 생명주기에 해당하는 메소드를 재정의(Overriding) 한다.

(2) 제어의 역행(IoC, Inversion of Control)

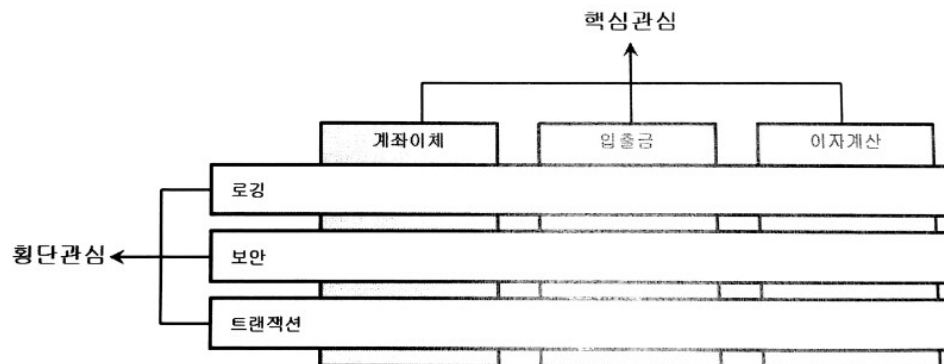
- 스프링은 제어의 역행을 통해 애플리케이션을 구성하는 객체 간의 느슨한 결합, 즉 낮은 결합도를 유지한다.

- IoC가 적용되지 않은 상황에서는 의존관계에 있는 객체를 변경할 때, 반드시 자바 코드를 수정해야 한다. 하지만 IoC가 적용되면 객체 생성을 자바 코드로 직접 처리하는 것이 아니라 컨테이너가 대신 처리하여 결합도가 떨어져서 유지보수가 편리해진다.
 - 제어의 역행(IoC, Inversion Of Control)은 메소드나 객체의 호출작업을 개발자가 결정하는 것이 아니라 외부에서 결정되는 것을 의미한다.
 - 의존성 주입(DI, Dependency Injection)은 제어의 역행으로 특정 객체에 필요한 객체를 결정한다.



(3) 관점지향 프로그래밍(AOP, Aspect Oriented Programming)

- AOP(Aspect Oriented Programming)는 횡단 관심사를 모듈로 분리하는 프로그래밍의 패러다임이다. 즉, 비즈니스 메소드를 개발할 때, 핵심 비즈니스 로직과 각 비즈니스 메소드마다 반복해서 등장하는 공통 로직을 분리함으로써 응집도가 높게 개발할 수 있도록 지원하는 것이다.



(4) 컨테이너(Container)

- 컨테이너는 특정 객체의 생성과 관리를 담당하여 객체 운용에 필요한 다양한 기능을 제공한다.
- 대표적인 컨테이너로 Servlet 객체를 생성하고 관리하는 'Servlet 컨테이너'와 EJB 객체를 생성하고 관리하는 'EJB 컨테이너'가 있다. 그리고 Servlet 컨테이너는 우리가 사용하는 톰캣 서버에도 포함되어 있다.
- 애플리케이션 운용에 필요한 객체를 생성하고 객체 간의 의존관계를 관리한다는 점에서 스프링도 일종의 컨테이너라고 할 수 있다.

B.2.3 IoC(Inversion of Control) 컨테이너

- 스프링 프레임워크를 이해하는데 가장 중요한 개념이 바로 컨테이너이다. 컨테이너의 개념은 스프링에서 처음 사용된 것이 아니며, 기존의 서블릿이나 EJB 기술에서 이미 사용해 왔다. 다음과 같이 간단한 서블릿 클래스를 만들었다고 가정하자.

[BoardWeb/src/main/java/hello/HelloServlet.java]

```
01 package hello;
02
03 import java.io.IOException;
04 import javax.servlet.ServletException;
05 import javax.servlet.http.HttpServlet;
06 import javax.servlet.http.HttpServletRequest;
07 import javax.servlet.http.HttpServletResponse;
08
09 public class HelloServlet extends HttpServlet {
10     private static final long serialVersionUID = 1L;
11
12     public HelloServlet() {
13         System.out.println("==> HelloServlet 객체 생성");
14     }
15
16     protected void doGet(HttpServletRequest request, HttpServletResponse response)
17         throws ServletException, IOException {
18         System.out.println("doGet() 메소드 호출");
19     }
20
21 }
```

// web.xml에 코드 추가

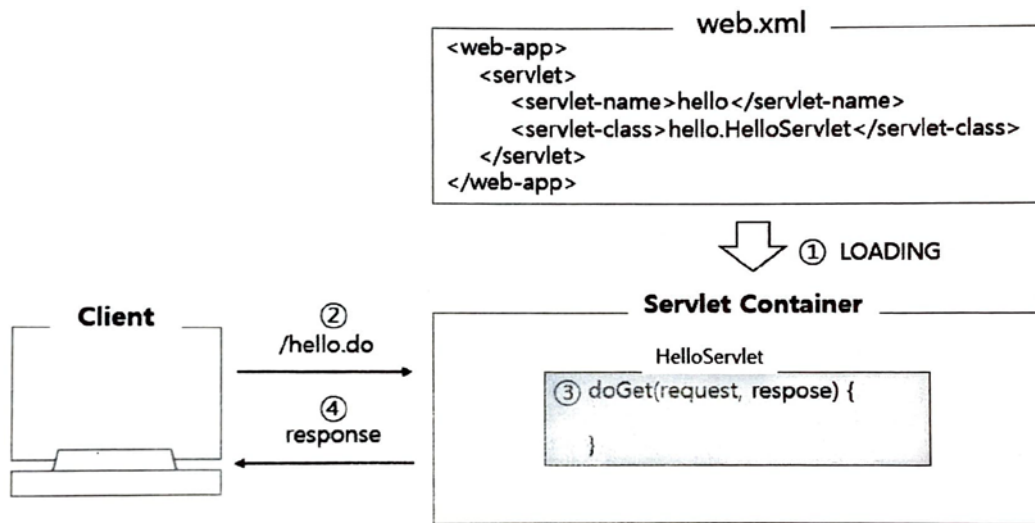
```
...
    <!-- lab begin -->
    <servlet>
        <description></description>
        <display-name>hello</display-name>
        <servlet-name>hello</servlet-name>
        <servlet-class>hello.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>hello</servlet-name>
        <url-pattern>/hello.do</url-pattern>
    </servlet-mapping>
    <!-- lab end -->
...
```

- 실행: <http://localhost:8080/biz/hello.do>

// 실행결과

```
12월 17, 2020 11:49:36 오전 org.apache.catalina.startup.Catalina start
INFO: 서버가 [33829] 밀리초 내에 시작되었습니다.
==> HelloServlet 객체 생성
doGet() 메소드 호출
```

- 서블릿 컨테이너는 다음 순서에 따라 동작한다.
 - WEB-INF/web.xml 파일을 로딩하여 구동
 - 브라우저로부터 /hello.do 요청 수신
 - hello.HelloServlet 클래스를 찾아 객체를 생성하고 doGet() 메소드 호출
 - doGet() 메소드 실행 결과를 클라이언트 브라우저로 전송



- 이렇듯 컨테이너는 자신이 관리할 클래스들이 등록된 XML 설정 파일을 로딩하여 구동한다. 그리고 클라이언트의 요청이 들어오는 순간 XML 설정 파일을 참조하여 객체를 생성하고, 객체의 생명주기를 관리한다.
- 스프링 컨테이너 역시 서블릿 컨테이너와 유사하게 동작하여 객체를 생성하고 객체들 사이의 의존관계를 처리하는 것에 대한 책임을 전적으로 개발자 소스코드에 의하지 않고 컨테이너로 처리하도록 하는데 이것을 제어의 역행이라 한다.
- 따라서 제어의 역행을 이용하면 소스에서 객체 생성과 의존성 관계에 대한 코드가 사라져 결과적으로 낮은 결합도의 컴포넌트를 구현할 수 있게 한다. --> 유지보수가 쉬워짐!

(1) 결합도(Coupling)가 높은 프로그램

- 결합도란 하나의 클래스가 다른 클래스와 얼마나 많이 연결되어 있는지를 나타내는 표현이며, 결합도가 높은 프로그램은 유지보수가 어렵다.
- 아래 코드에서 SamsungTVCoupling과 LgTVCoupling는 메소드 시그니처(signature)가 다르므로 TVUserCoupling 코드 대부분을 수정해야 TV를 교체할 수 있다. 만약 TVUserCoupling과 같은 클라이언트 프로그램이 하나가 아니라 여러 개라면 유지보수는 더욱더 힘들 것이며, TV 교체를 결정하기가 쉽지 않을 것이다.

[src/main/java/polymorphism/SamsungTVCoupling.java]

```

01 package polymorphism;
02
03 public class SamsungTVCoupling {
04
05     public void powerOn() {
06         System.out.println("SamsungTV---전원을 켜다.");
07     }
08
09     public void powerOff() {
10         System.out.println("SamsungTV---전원을 끈다.");
11     }
12
13     public void volumeUp() {

```



```

14         System.out.println("SamsungTV---소리 올린다.");
15     }
16
17     public void volumeDown() {
18         System.out.println("SamsungTV---소리 내린다.");
19     }
20 }

```

[src/main/java/polymorphism/LgTVCoupling.java]

```

01 package polymorphism;
02
03 public class LgTVCoupling {
04
05     public void turnOn() {
06         System.out.println("LgTV---전원 켜다.");
07     }
08
09     public void turnOff() {
10         System.out.println("LgTV---전원 끈다.");
11     }
12
13     public void soundUp() {
14         System.out.println("LgTV---소리 올린다.");
15     }
16
17     public void soundDown() {
18         System.out.println("LgTV---소리 내린다.");
19     }
20 }

```

[src/main/java/polymorphism/TVUserCoupling.java]

```

01 package polymorphism;
02
03 public class TVUserCoupling {
04     public static void main(String[] args) {
05         SamsungTVCoupling tv = new SamsungTVCoupling();
06         tv.powerOn();
07         tv.volumeUp();
08         tv.volumeDown();
09         tv.powerOff();
10
11         LgTVCoupling tv1 = new LgTVCoupling();
12         tv1.turnOn();
13         tv1.soundUp();
14         tv1.soundDown();
15         tv1.turnOff();
16     }
17 }

```

```

// 실행결과
SamsungTV---전원을 켜다.
SamsungTV---소리 올린다.
SamsungTV---소리 내린다.
SamsungTV---전원을 끈다.
LgTV---전원 켜다.
LgTV---소리 올린다.
LgTV---소리 내린다.
LgTV---전원 끈다.

```

(2) 다형성 이용하기

- 결합도를 낮추기 위해서 다양한 방법을 사용할 수 있겠지만, 가장 쉽게 생각할 수 있는 것이 객체지향 언어의 핵심 개념인 다형성(Polymorphism)을 이용하는 것이다.
- 이 방법 역시 TV를 변경하고자 할 때, TV 클래스 객체를 생성하는 소스를 수정해야만 한다.

[TV.java]

```
01 package polymorphism;
02
03 public interface TV {
04     public void powerOn();
05
06     public void powerOff();
07
08     public void volumeUp();
09
10     public void volumeDown();
11
12 }
```

[SamsungTV.java]

```
01 package polymorphism;
02
03 public class SamsungTV implements TV {
04     private Speaker speaker;
05     private int price;
06
07     public SamsungTV() {
08         System.out.println("=> SamsungTV(1) 객체 생성");
09     }
10
11     public void setSpeaker(Speaker speaker) {
12         System.out.println("=> setSpeaker() 호출");
13         this.speaker = speaker;
14     }
15
16     public void setPrice(int price) {
17         System.out.println("=> setPrice() 호출");
18         this.price = price;
19     }
20
21     public void powerOn() {
22         System.out.println("SamsungTV---전원을 켜다. (가격 : " + price + ")");
23     }
24
25     public void powerOff() {
26         System.out.println("SamsungTV---전원을 끈다.");
27     }
28
29     public void volumeUp() {
30         //speaker.volumeUp();
31         System.out.println("SamsungTV---소리 올린다.");
32     }
33
34     public void volumeDown() {
35         //speaker.volumeDown();
36         System.out.println("SamsungTV---소리 내린다.");
37     }
38 }
```

```
37     }
38 }
```

[LgTV.java]

```
01 package polymorphism;
02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.stereotype.Component;
05
06 @Component("tv")
07 public class LgTV implements TV {
08     @Autowired
09     private Speaker speaker;
10
11     public LgTV() {
12         System.out.println("==> LgTV 객체 생성됨");
13     }
14
15     public void powerOn() {
16         System.out.println("LgTV---전원 켜다.");
17     }
18     public void powerOff() {
19         System.out.println("LgTV---전원 끈다.");
20     }
21     public void volumeUp() {
22         //speaker.volumeUp();
23         System.out.println("LgTV---소리 올린다.");
24     }
25     public void volumeDown() {
26         //speaker.volumeDown();
27         System.out.println("LgTV---소리 내린다.");
28     }
29 }
```

[TVUserPolymorphism.java]

```
01 package polymorphism;
02
03 public class TVUserPolymorphism {
04     public static void main(String[] args) {
05
06         TV tv = new SamsungTV();
07         tv.powerOn();
08         tv.volumeUp();
09         tv.volumeDown();
10         tv.powerOff();
11
12         tv = new LgTV(); // TV를 변경시, TV 객체를 생성하는 소스를 수정해야 한다.
13         tv.powerOn();
14         tv.volumeUp();
15         tv.volumeDown();
16         tv.powerOff();
17
18     }
19 }
```

// 실행결과

```
==> SamsungTV(1) 객체 생성
SamsungTV---전원을 켜다. (가격 : 0)
```

```

SamsungTV---소리 올린다.
SamsungTV---소리 내린다.
SamsungTV---전원을 끈다.
==> LgTV 객체 생성됨
LgTV---전원 켜다.
LgTV---소리 올린다.
LgTV---소리 내린다.
LgTV---전원 끈다.

```

(3) 디자인 패턴 이용하기

- 결합도를 낮추기 위한 또 다른 방법으로 디자인 패턴을 이용하는 방법이 있다.
- TV를 교체할 때, 클라이언트 소스를 수정하지 않고 명령행 매개변수만 수정하여 TV를 교체할 수 있어 유지보수가 더욱 편리하다.

[BeanFactory.java]

```

01  package polymorphism;
02
03  public class BeanFactory {
04      public Object getBean(String beanName){
05          if(beanName.equals("samsung")){
06              return new SamsungTV();
07          } else if(beanName.equals("lg")){
08              return new LgTV();
09          }
10          return null;
11      }
12  }
13  }

```

[TVUserDesignPattern.java]

```

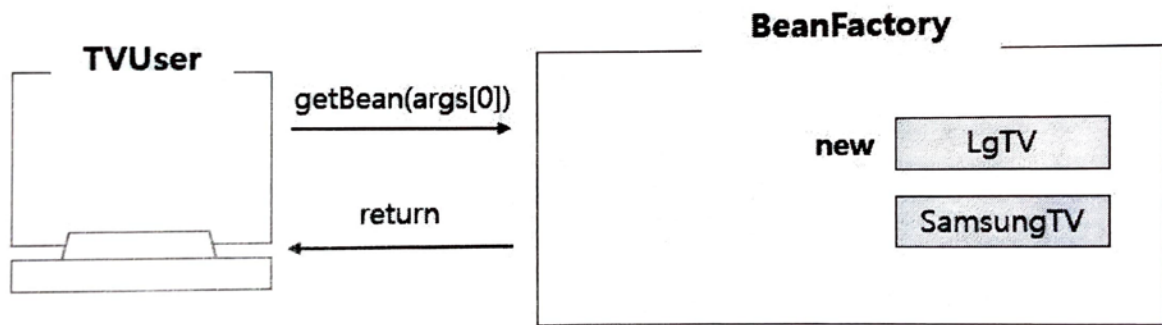
01  package polymorphism;
02
03  public class TVUserDesignPattern {
04      public static void main(String[] args) {
05
06          BeanFactory factory = new BeanFactory();
07          TV tv = (TV) factory.getBean(args[0]);
08          tv.powerOn();
09          tv.volumeUp();
10          tv.volumeDown();
11          tv.powerOff();
12
13      }
14  }

```

```

// 실행결과
==> LgTV 객체 생성됨
LgTV---전원 켜다.
LgTV---소리 올린다.
LgTV---소리 내린다.
LgTV---전원 끈다.

```



B.3 스프링 컨테이너 및 설정 파일

B.3.1 스프링 IoC 시작하기

(1) 스프링 설정 파일 생성

- "메뉴 > New > Other > Spring > Spring Bean Configuration File > Next"를 클릭하여 아래와 같이 입력하고 "Finish" 버튼을 누른다.
 - Enter or select the parent folder: BoardWebLab/src/main/resources
 - File name: context
- 앞에서 작성한 TV 예제를 스프링 기반으로 테스트하기 위해서 SamsungTV 클래스를 스프링 설정 파일에 등록한다.
 - <bean> 엘리먼트의 class 속성값에 "sams"이라고 간략히 입력 후, <Ctrl + Space>를 누르면 자동으로 SamsungTV 클래스가 등록된다.

[BoardWeb/src/main/resources/context.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04       xsi:schemaLocation="http://www.springframework.org/schema/beans
05       http://www.springframework.org/schema/beans/spring-beans.xsd">
06
07
08     <bean id="tv" class="polymorphism.SamsungTV" />
09
10 </beans>

```

(2) 스프링 컨테이너 구동 및 테스트

[TVUser.java]

```

01 package polymorphism;
02
03 import org.springframework.context.support.AbstractApplicationContext;
04 import org.springframework.context.support.GenericXmlApplicationContext;
05
06 public class TVUser {
07     public static void main(String[] args) {
08         // 1. Spring 컨테이너를 구동한다.
09         AbstractApplicationContext factory =

```

```

10         new GenericXmlApplicationContext("context.xml");
11
12         // 2. Spring 컨테이너로부터 필요한 객체를 요청(Lookup)한다.
13         TV tv = (TV)factory.getBean("tv");
14         tv.powerOn();
15         tv.volumeUp();
16         tv.volumeDown();
17         tv.powerOff();
18
19         // 3. Spring 컨테이너를 종료한다.
20         factory.close();
21
22     }
23 }

```

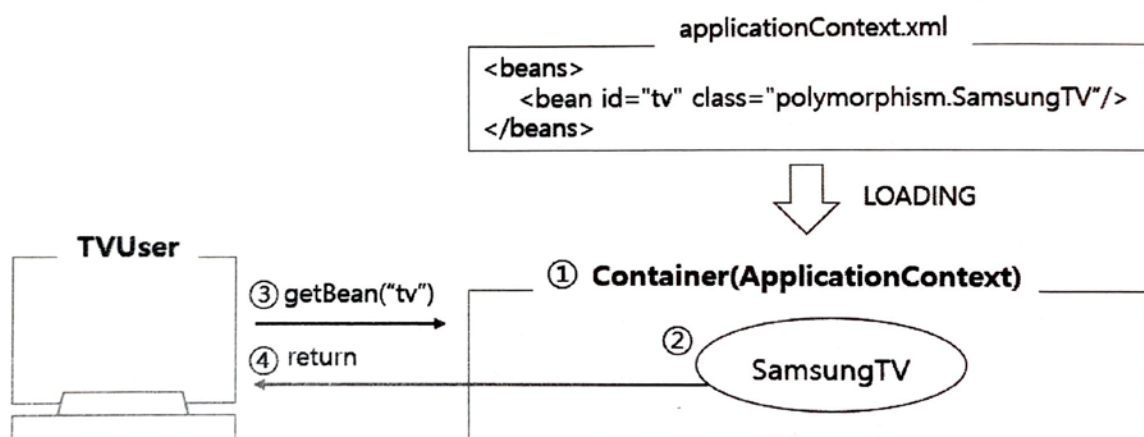
```

// 실행결과
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 20:01:30
KST 2018]; root of context hierarchy
==> SamsungTV(1) 객체 생성
SamsungTV---전원을 켜다. (가격 : 28000)
SamsungTV---소리 올린다.
SamsungTV---소리 내린다.ㅋ
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 20:01:30
KST 2018]; root of context hierarchy

```

■ 스프링 컨테이너의 동작 순서

1. TVUser 클라이언트가 스프링 설정 파일을 로딩하여 컨테이너 구동
2. 스프링 설정 파일에 <bean> 등록된 SamsungTV 객체 생성
3. getBean() 메소드로 이름이 'tv'인 객체를 요청(Lookup)
4. SamsungTV 객체 반환



(3) 스프링 컨테이너의 종류

- 스프링에서는 BeanFactory와 이를 상속한 ApplicationContext 두 가지 유형의 컨테이너를 제공한다. 일반적으로 스프링 프로젝트에서 BeanFactory를 사용할 일은 없다. 반면에

ApplicationContext는 BeanFactory가 제공하는 <bean> 객체 관리 기능 외에도 트랜잭션 관리나 메시지 기반의 다국어 처리 등 다양한 기능을 지원한다.

- ApplicationContext의 구현 클래스는 매우 다양하나 실제로 가장 많이 사용하는 두 개의 클래스만 알고 있으면 된다.

구현 클래스	기능
GenericXmlApplicationContext	파일 시스템이나 클래스 경로에 있는 XML 설정 파일을 로딩하여 구동하는 컨테이너이다.
XmlWebApplicationContext	웹 기반의 스프링 애플리케이션을 개발할 때 사용하는 컨테이너이다.

B.3.2 스프링 XML 설정

(1) <beans> 루트 엘리먼트

- 스프링 컨테이너는 <bean> 저장소에 해당하는 XML 설정 파일을 참조하여 <bean>의 생명주기를 관리하고 여러 가지 서비스를 제공한다.

```
// applicationContext.xml
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
</beans>
```

(2) <import> 엘리먼트

- 스프링 기반의 애플리케이션은 단순한 <bean> 등록 외에도 트랜잭션 관리, 예외 처리, 다국어 처리 등 복잡하고 다양한 설정이 필요하다.
- 기능별 여러 XML 파일로 나누어 설정하고 하나로 통합할 때 <import> 엘리먼트를 사용한다.

```
// applicationContext.xml
<beans>
  <import resource="context-datasource.xml" />
  <import resource="context-transaction.xml" />
</beans>
```

(3) <bean> 엘리먼트

- 스프링 설정 파일에 클래스를 등록하려면 <bean> 엘리먼트를 사용한다.

```
// applicationContext.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"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
```

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

</beans>
```

(4) <bean> 엘리먼트 속성

■ init-method 속성

- Servlet 컨테이너는 web.xml 파일에 등록된 Servlet 클래스의 객체를 생성할 때 디폴트 생성자만 인식한다. 따라서 생성자로 Servlet 객체의 멤버변수를 초기화할 수 없다. 그래서 서블릿은 init() 메소드를 재정의(Overriding)하여 멤버변수를 초기화한다.
- 스프링 컨테이너 역시 스프링 설정 파일에 등록된 클래스를 객체 생성할 때 디폴트 생성자를 호출한다. 따라서 멤버변수 초기화 작업이 필요하다면, <bean> 엘리먼트에 init-method 속성을 사용한다.

```
// applicationContext.xml
<bean id="tv" class="polymorphism.SamsungTV" init-method="initMethod" />
```

```
// SamsungTV.java
package polymorphism;

public class SamsungTV implements TV {
    private Speaker speaker;
    private int price;

    public SamsungTV() {
        System.out.println("==> SamsungTV(1) 객체 생성");
    }

    public void initMethod() {
        System.out.println("객체 초기화 작업 처리...");
    }
}
...(생략)...
```

```
// 실행결과 (TVUser.java)
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 20:48:06
KST 2018]; root of context hierarchy
==> SamsungTV(1) 객체 생성
객체 초기화 작업 처리...
SamsungTV---전원을 켜다. (가격 : 0)
SamsungTV---소리 올린다.
SamsungTV---소리 내린다.
SamsungTV---전원을 끈다.
```

■ destroy-method 속성

- <bean> 엘리먼트에서 destroy-method 속성을 이용하여 스프링 컨테이너가 객체를 삭제하기 직전에 호출될 임의의 메소드를 지정할 수 있다.

```
// applicationContext.xml
<bean id="tv" class="polymorphism.SamsungTV" init-method="initMethod" destroy-method="destroyMethod" />
```



```
// SamsungTV.java
package polymorphism;

public class SamsungTV implements TV {
    private Speaker speaker;
    private int price;

    public SamsungTV() {
        System.out.println("==> SamsungTV(1) 객체 생성");
    }

    public void initMethod() {
        System.out.println("객체 초기화 작업 처리...");
    }

    public void destroyMethod() {
        System.out.println("객체 삭제 전에 처리할 로직 처리...");
    }
}
...(생략)...
```

```
// 실행결과 (TVUser.java)
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 21:00:14
KST 2018]; root of context hierarchy
==> SamsungTV(1) 객체 생성
객체 초기화 작업 처리...
SamsungTV---전원을 켜다. (가격 : 0)
SamsungTV---소리 올린다.
SamsungTV---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 21:00:14
KST 2018]; root of context hierarchy
객체 삭제 전에 처리할 로직 처리...
```

■ lazy-init 속성

- 스프링에서는 컨테이너가 구동되는 시점이 아닌 해당 <bean>이 사용되는 시점에 객체를 생성하도록 lazy-init 속성을 제공한다.

```
// applicationContext.xml
<bean id="tv" class="polymorphism.SamsungTV" lazy-init="true" />
```

■ scope 속성

- 하나의 객체만 생성하도록 제어하기 위해 싱글톤 패턴(singleton pattern) 사용한다. 그러나 싱글톤 패턴을 구현하려면 일일이 클래스에 패턴 관련 코드를 작성해야 하므로 매우 귀찮은 일이다.
- 스프링 컨테이너는 컨테이너가 생성한 bean을 어느 범위에서 사용할 수 있는지 지정할 수 있는데, 이때 scope 속성을 사용한다. scope 속성값은 기본이 싱글톤이다. 이는 해당 bean이 스프링 컨테이너에 의해 단 하나만 생성되어 운용되도록 한다. 일반적으로 scope은 생략하므로 스프링 컨테이너가 관리하는 <bean>들은 대부분 싱글톤으로 운영된다.
- scope 속성을 "prototype"으로 지정하면 해당 <bean>이 요청될 때마다 매번 새로운 객체를 생성하여 반환한다.

```
// applicationContext.xml
<bean id="tv" class="polymorphism.SamsungTV" scope="singleton" />
<bean id="tv" class="polymorphism.SamsungTV" scope="prototype" />
```

```
// 실행결과 (TVUserSingleton.java)
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [applicationContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 21:21:58
KST 2018]; root of context hierarchy
==> SamsungTV(1) 객체 생성
==> SamsungTV(1) 객체 생성
==> SamsungTV(1) 객체 생성
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Thu May 10 21:21:58
KST 2018]; root of context hierarchy
```

B.4 의존성 주입

B.4.1 의존성 관리

(1) 스프링의 의존성 관리 방법

- 스프링은 IoC를 다음 두 가지 형태로 지원한다.
 - Dependency Lookup: 컨테이너가 애플리케이션 운용에 필요한 객체를 생성하고 클라이언트는 컨테이너가 생성한 객체를 검색하여 사용하는 방식
 - Dependency Injection: 객체 사이의 의존관계를 스프링 설정 파일에 등록된 정보를 바탕으로 컨테이너가 자동으로 처리해 주는 방식, 대부분 이 방식을 사용하여 개발한다.

(2) 의존성 관계

- 의존성(Dependency) 관계란 객체와 객체의 결합 관계이다. 즉, 하나의 객체에서 다른 객체의 변수나 메소드를 이용해야 한다면 이용하려는 객체에 대한 객체 생성과 객체의 레퍼런스 정보가 필요하다.

[SonySpeaker.java]

```
01 package polymorphism;
02
03 public class SonySpeaker implements Speaker {
04     public SonySpeaker() {
05         System.out.println("==> SonySpeaker 객체 생성");
06     }
07     public void volumeUp() {
08         System.out.println("SonySpeaker---소리 올린다.");
09     }
10     public void volumeDown() {
11         System.out.println("SonySpeaker---소리 내린다.");
12     }
13 }
14 }
```

[SamsungTVDependency.java]

```
01 package polymorphism;
02
03 public class SamsungTVDependency implements TV {
04     private SonySpeaker speaker;
05
06     public SamsungTVDependency() {
07         System.out.println("==> SamsungTV 객체 생성");
08     }
09
10     public void powerOn() {
11         System.out.println("SamsungTV---전원을 켜다.");
12     }
13
14     public void powerOff() {
15         System.out.println("SamsungTV---전원을 끈다.");
16     }
17
18     public void volumeUp() {
19         speaker = new SonySpeaker();
20         speaker.volumeUp();
21     }
22
23     public void volumeDown() {
24         speaker = new SonySpeaker();
25         speaker.volumeDown();
26     }
27 }
```

// TVUserDependency.java의 실행결과

```
==> SamsungTV 객체 생성
SamsungTV---전원을 켜다.
==> SonySpeaker 객체 생성
SonySpeaker---소리 올린다.
==> SonySpeaker 객체 생성
SonySpeaker---소리 내린다.
SamsungTV---전원을 끈다.
```

B.4.2 생성자 인젝션 이용하기

- 스프링 컨테이너는 XML 설정 파일에 등록된 클래스를 찾아서 객체 생성할 때 기본적으로 매개변수가 없는 기본(Default) 생성자를 호출한다.
- 하지만 컨테이너가 기본 생성자 말고 매개변수를 가지는 다른 생성자를 호출하도록 설정할 수 있는데, 이 기능을 이용하여 생성자 인젝션(Constructor Injection)을 처리한다.

[polymorphism/SamsungTV.java]

```
01 package polymorphism;
02
03 public class SamsungTV implements TV {
04     private Speaker speaker;
05     private int price;
06 }
```

```

07     public SamsungTV() {
08         System.out.println("==> SamsungTV(1) 객체 생성");
09     }
10
11     public SamsungTV(Speaker speaker) {
12         System.out.println("==> SamsungTV(2) 객체 생성");
13         this.speaker = speaker;
14     }
15
16     public void initMethod() {
17         System.out.println("객체 초기화 작업 처리...");
18     }
19
20     public void destroyMethod() {
21         System.out.println("객체 삭제 전에 처리할 로직 처리...");
22     }
23
24     public void setSpeaker(Speaker speaker) {
25         System.out.println("==> setSpeaker() 호출");
26         this.speaker = speaker;
27     }
28
29     public void setPrice(int price) {
30         System.out.println("==> setPrice() 호출");
31         this.price = price;
32     }
33
34     public void powerOn() {
35         System.out.println("SamsungTV---전원을 켜다. (가격 : " + price + ")");
36     }
37
38     public void powerOff() {
39         System.out.println("SamsungTV---전원을 끈다.");
40     }
41
42     public void volumeUp() {
43         speaker.volumeUp();
44         //System.out.println("SamsungTV---소리 올린다.");
45     }
46
47     public void volumeDown() {
48         speaker.volumeDown();
49         //System.out.println("SamsungTV---소리 내린다.");
50     }
51 }

```

[context.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04     xmlns:aop="http://www.springframework.org/schema/aop"
05     xmlns:context="http://www.springframework.org/schema/context"
06     xmlns:tx="http://www.springframework.org/schema/tx"
07     xmlns:p="http://www.springframework.org/schema/p"
08     xsi:schemaLocation="http://www.springframework.org/schema/beans
09         http://www.springframework.org/schema/beans/spring-beans.xsd
10         http://www.springframework.org/schema/context
11         http://www.springframework.org/schema/context/spring-context-4.2.xsd
12         http://www.springframework.org/schema/aop
13         http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
14         http://www.springframework.org/schema/tx
15         http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">
16

```

```

17      <!-- <bean id="tv" class="polymorphism.SamsungTV" scope="prototype" /> -->
18      <bean id="tv" class="polymorphism.SamsungTV">
19          <constructor-arg ref="sony"></constructor-arg>
20      </bean>
21
22      <bean id="sony" class="polymorphism.SonySpeaker"></bean>
23      <bean id="apple" class="polymorphism.AppleSpeaker"></bean>
24
25  </beans>

```

// TVUser.java의 실행결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [applicationContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Sat Nov 03 09:42:48
KST 2018]; root of context hierarchy
==> SonySpeaker 객체 생성
==> SamsungTV(2) 객체 생성
SamsungTV---전원을 켜다. (가격 : 0)
SonySpeaker---소리 올린다.
SonySpeaker---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplication

```

(1) 다중 변수 매칭

// SamsungTV.java의 일부

```

public SamsungTV(Speaker speaker, int price) {
    System.out.println("==> SamsungTV(3) 객체 생성");
    this.speaker = speaker;
    this.price = price;
}

public void powerOn() {
    System.out.println("SamsungTV---전원을 켜다. (가격 : " + price + ")");
}

```

// context.xml의 일부

```

<bean id="tv" class="polymorphism.SamsungTV">
    <constructor-arg index="0" ref="sony"></constructor-arg>
    <constructor-arg index="1" value="27000000"></constructor-arg>
</bean>
<bean id="sony" class="polymorphism.SonySpeaker"></bean>
<bean id="apple" class="polymorphism.AppleSpeaker"></bean>

```

// TVUser.java의 실행결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:18:15
KST 2018]; root of context hierarchy

```

```

==> SonySpeaker 객체 생성
==> SamsungTV(3) 객체 생성
==> AppleSpeaker 객체 생성
SamsungTV---전원을 켜다. (가격 : 27000000)
SonySpeaker---소리 올린다.
SonySpeaker---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:18:15
KST 2018]; root of context hierarchy

```

(2) 의존관계 변경

- 지금까지는 SamsungTV 객체가 SonySpeaker를 이용하여 동작했지만 유지보수 과정에서 다른 스피커로 교체하는 상황도 발행할 것이다. 의존성 주입은 이런 상황을 매우 효과적으로 처리해준다.

```

// context.xml의 일부

<bean id="tv" class="polymorphism.SamsungTV">
    <constructor-arg index="0" ref="apple"></constructor-arg>
    <constructor-arg index="1" value="27000000"></constructor-arg>
</bean>
<bean id="sony" class="polymorphism.SonySpeaker"></bean>
<bean id="apple" class="polymorphism.AppleSpeaker"></bean>

```

```

// TVUser.java의 실행결과

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:22:13
KST 2018]; root of context hierarchy
==> AppleSpeaker 객체 생성
==> SamsungTV(3) 객체 생성
==> SonySpeaker 객체 생성
SamsungTV---전원을 켜다. (가격 : 27000000)
AppleSpeaker---소리 올린다.
AppleSpeaker---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:22:13
KST 2018]; root of context hierarchy

```

B.4.3 Setter 인젝션 사용하기

- Setter 메소드를 호출하여 의존성 주입을 처리하는 방법이다.

(1) Setter 인젝션 기본

```

// SamsungTV.java의 일부

```

```

public void setSpeaker(Speaker speaker) {
    System.out.println("==> setSpeaker() 호출");
    this.speaker = speaker;
}

public void setPrice(int price) {
    System.out.println("==> setPrice() 호출");
    this.price = price;
}

```

// context.xml의 일부

```

<bean id="tv" class="polymorphism.SamsungTV">
    <constructor-arg ref="sony"></constructor-arg>
    <property name="speaker" ref="apple"></property>
    <property name="price" value="27000000"></property>
</bean>

<bean id="sony" class="polymorphism.SonySpeaker"></bean>
<bean id="apple" class="polymorphism.AppleSpeaker"></bean>

```

// TVUser.java의 실행결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [applicationContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Sat Nov 03 10:03:01
KST 2018]; root of context hierarchy
==> SonySpeaker 객체 생성
==> SamsungTV(2) 객체 생성
==> AppleSpeaker 객체 생성
==> setSpeaker() 호출
==> setPrice() 호출
SamsungTV---전원을 켜다. (가격 : 27000000)
AppleSpeaker---소리 올린다.
AppleSpeaker---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@7dc5e7b4: startup date [Sat Nov 03 10:03:01
KST 2018]; root of context hierarchy

```

(2) p 네임스페이스 사용하기

- Setter 인젝션을 설정할 때, p 네임스페이스를 이용하면 좀 더 효율적으로 의존성 주입을 처리할 수 있다.

// context.xml의 일부

```

<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:p="http://www.springframework.org/schema/p"
    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-4.2.xsd">

```

```

<!-- lab begin -->
<!-- <bean id="tv" class="polymorphism.SamsungTV">
    <constructor-arg index="0" ref="apple"></constructor-arg>
    <constructor-arg index="1" value="27000000"></constructor-arg>
</bean> -->
<bean id="tv" class="polymorphism.SamsungTV" p:speaker-ref="apple" p:price="28000"/>

```

// TVUser.java의 실행결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:35:43
KST 2018]; root of context hierarchy
==> SamsungTV(1) 객체 생성
==> AppleSpeaker 객체 생성
==> setPrice() 호출
==> setSpeaker() 호출
==> SonySpeaker 객체 생성
SamsungTV---전원을 켜다. (가격 : 28000)
AppleSpeaker---소리 올린다.
AppleSpeaker---소리 내린다.
SamsungTV---전원을 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sun Nov 04 11:35:43
KST 2018]; root of context hierarchy

```

B.4.4 컬렉션(Collection) 객체 설정

(1) List 타입 매핑

[com.springbook.ioc.injection.CollectionBean.java]

```

01 package com.springbook.ioc.injection;
02
03 import java.util.List;
04
05 public class CollectionBean {
06     private List<String> addressList;
07
08     public void setAddressList(List<String> addressList) {
09         this.addressList = addressList;
10     }
11     public List<String> getAddressList() {
12         return addressList;
13     }
14 }

```

[context.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <beans xmlns="http://www.springframework.org/schema/beans"
03     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04     xsi:schemaLocation="http://www.springframework.org/schema/beans
05     http://www.springframework.org/schema/beans/spring-beans.xsd">
06

```



```

07         <bean id="collectionBean" class="com.springbook.ioc.injection.CollectionBean">
08             <property name="addressList">
09                 <list>
10                     <value>서울시 강남구 역삼동</value>
11                     <value>서울시 성동구 행당동</value>
12                 </list>
13             </property>
14         </bean>
15
16     </beans>

```

[com.springbook.ioc.injection.CollectionBeanClient.java]

```

01 package com.springbook.ioc.injection;
02
03 import java.util.List;
04
05 import org.springframework.context.support.AbstractApplicationContext;
06 import org.springframework.context.support.GenericXmlApplicationContext;
07
08 import com.springbook.ioc.injection.CollectionBean;
09
10 public class CollectionBeanClient {
11     public static void main(String[] args) {
12         AbstractApplicationContext factory =
13             new GenericXmlApplicationContext("context.xml");
14
15         CollectionBean bean = (CollectionBean)
16             factory.getBean("collectionBean");
17         List<String> addressList = bean.getAddressList();
18         for (String address : addressList) {
19             System.out.println(address.toString());
20         }
21         factory.close();
22     }
23 }

```

// CollectionBeanClient.java의 실행결과

```

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sat Nov 03 13:07:37
KST 2018]; root of context hierarchy
서울시 강남구 역삼동
서울시 성동구 행당동
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@1e81f4dc: startup date [Sat Nov 03 13:07:37
KST 2018]; root of context hierarchy

```

B.5 어노테이션 기반 설정

B.5.1 어노테이션 설명 기초

(1) Context 네임스페이스 추가

- 어노테이션 설정을 추가하려면 다음과 같이 스프링 설정 파일의 루트 엘리먼트인 <beans>에

Context 관련 네임스페이스와 스키마 문서의 위치를 등록해야 한다.

```
// src/main/resources/context.xml의 일부

<?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:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:p="http://www.springframework.org/schema/p"
       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-4.2.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">

</beans>
```

(2) 컴포넌트 스캔(component-scan) 설정

- 스프링 설정 파일에 애플리케이션에서 사용할 객체들을 <bean> 등록하지 않고 자동으로 생성하려면 <context:component-scan/>이라는 엘리먼트를 정의해야 한다.

```
// src/main/resources/context.xml의 일부

<?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:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:p="http://www.springframework.org/schema/p"
       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-4.2.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">

    <context:component-scan base-package="polymorphism"></context:component-scan>

</beans>
```

(3) @Component

- <context:component-scan>를 설정했으면 이제 스프링 설정 파일에 클래스들을 일일이 <bean> 엘리먼트로 등록할 필요가 없다. @Component만 클래스 선언부 위에 설정하면 끝난다.

```
// src/main/java/polymorphism/LgTV.java의 일부

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 생성됨");
    }
}
```

```
// src/main/java/polymorphism/TVUser.java의 실행결과

INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [context.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@75a1cd57: startup date [Thu Jul 30 16:32:08
KST 2020]; root of context hierarchy
INFO : org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor - JSR-330
'javax.inject.Inject' annotation found and supported for autowiring
==> AppleSpeaker 객체 생성
==> LgTV 객체 생성됨
LgTV---전원 켜다.
AppleSpeaker---소리 올린다.
AppleSpeaker---소리 내린다.
LgTV---전원 끈다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@75a1cd57: startup date [Thu Jul 30 16:32:08
KST 2020]; root of context hierarchy
```

B.5.2 의존성 주입 설정

(1) 의존성 주입 어노테이션

어노테이션	설명
@Autowired	주로 변수 위에 설정하여 해당 타입의 객체를 찾아서 자동으로 할당한다.
@Qualifier	특정 객체의 이름을 이용하여 의존성 주입할 때 사용한다.
@Inject	@Autowired와 동일한 기능을 제공한다.
@Resource	@Autowired와 @Qualifier의 기능을 결합한 어노테이션이다.

(2) @Autowired

- @Autowired는 생성자나 메소드, 멤버변수 위에 모두 사용할 수 있다. 어디에 사용하든 결과가 같아서 상관없지만, 대부분은 멤버변수 위에 선언하여 사용한다.

```
[polymorphism.LgTV.java]

01 package polymorphism;
```

```

02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.stereotype.Component;
05
06 @Component("tv")
07 public class LgTV implements TV {
08     @Autowired
09     private Speaker speaker;
10
11     public LgTV() {
12         System.out.println("==> LgTV 객체 생성됨");
13     }
14
15     public void powerOn() {
16         System.out.println("LgTV---전원 켜다.");
17     }
18     public void powerOff() {
19         System.out.println("LgTV---전원 끈다.");
20     }
21     public void volumeUp() {
22         speaker.volumeUp();
23         //System.out.println("LgTV---소리 올린다.");
24     }
25     public void volumeDown() {
26         speaker.volumeDown();
27         //System.out.println("LgTV---소리 내린다.");
28     }
29 }

```

(3) @Qualifier

- @Autowired 대상이 되는 Speaker 타입의 객체가 AppleSpeaker, SonySpeaker과 같이 둘 다 메모리에 생성되어 있을 경우, 어떤 객체를 의존성 주입할지 모르기 때문에 예외가 발생한다. (NoUniqueBeanDefinitionException)
- 이런 문제를 해결하기 위해서 스프링은 @Qualifier 어노테이션을 제공한다. 단, @Autowired와 같이 사용해야 한다.

[polymorphism.LgTV.java]

```

01 package polymorphism;
02
03 import org.springframework.beans.factory.annotation.Autowired;
04 import org.springframework.beans.factory.annotation.Qualifier;
05 import org.springframework.stereotype.Component;
06
07 @Component("tv")
08 public class LgTV implements TV {
09     @Autowired
10     @Qualifier("apple")
11     private Speaker speaker;
12
13     public LgTV() {
14         System.out.println("==> LgTV 객체 생성됨");
15     }
16
17     public void powerOn() {
18         System.out.println("LgTV---전원 켜다.");
19     }
20     public void powerOff() {
21         System.out.println("LgTV---전원 끈다.");

```

```

22     }
23     public void volumeUp() {
24         //speaker.volumeUp();
25         System.out.println("LgTV---소리 올린다.");
26     }
27     public void volumeDown() {
28         //speaker.volumeDown();
29         System.out.println("LgTV---소리 내린다.");
30     }
31 }

```

(4) @Resource

- 앞에서 살펴본 @Autowired는 변수의 타입을 기준으로 객체를 검색하여 의존성 주입을 처리하지만, @Resource는 객체의 이름을 이용하여 의존성 주입을 처리한다.

[polymorphism.LgTV.java]

```

01 package polymorphism;
02
03 import javax.annotation.Resource;
04
05 import org.springframework.stereotype.Component;
06
07 @Component("tv")
08 public class LgTV implements TV {
09     // @Autowired
10     // @Qualifier("apple")
11     @Resource(name = "apple")
12     private Speaker speaker;
13
14     public LgTV() {
15         System.out.println("==> LgTV 객체 생성됨");
16     }
17
18     public void powerOn() {
19         System.out.println("LgTV---전원 켜다.");
20     }
21
22     public void powerOff() {
23         System.out.println("LgTV---전원 끈다.");
24     }
25
26     public void volumeUp() {
27         // speaker.volumeUp();
28         System.out.println("LgTV---소리 올린다.");
29     }
30
31     public void volumeDown() {
32         // speaker.volumeDown();
33         System.out.println("LgTV---소리 내린다.");
34     }
35 }

```

(5) 어노테이션과 XML 설정 병행하여 사용하기

- 스프링으로 의존성 주입을 처리할 때, XML 설정과 어노테이션 설정은 장단점이 서로 상충한다.

- XML 방식은 자바 소스를 수정하지 않고 XML 파일의 설정만 변경하면 실행되는 Speaker를 교체할 수 있어서 유지보수가 편하다. 하지만 XML 설정에 대한 부담 역시 존재한다.
- 어노테이션 기반 설정은 XML 설정에 대한 부담도 없고, 의존관계에 대한 정보가 자바 소스에 들어있어서 사용하기는 편하다. 하지만 자바소스를 수정하지 않고 Speaker를 교체할 수 없다는 문제가 생긴다.
- 이런 문제를 서로의 장점을 조합하는 것으로 해결할 수 있다.

```
// polymorphism.LgTV.java 의 일부

@Component("tv")
public class LgTV implements TV {
    @Autowired
    private Speaker speaker;

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

    public void powerOn() {
        System.out.println("LgTV---전원 켜다.");
    }

    public void powerOff() {
        System.out.println("LgTV---전원 끈다.");
    }

    public void volumeUp() {
        speaker.volumeUp();
        //System.out.println("LgTV---소리 올린다.");
    }

    public void volumeDown() {
        speaker.volumeDown();
        //System.out.println("LgTV---소리 내린다.");
    }
}
```

```
// src/main/resources/context.xml의 일부

<context:component-scan base-package="polymorphism"></context:component-scan>

<!-- <bean id="apple" class="polymorphism.AppleSpeaker"></bean> -->
<bean id="sony" class="polymorphism.SonySpeaker"></bean>
```

B.5.3 추가 어노테이션

- @Component를 이용하여 스프링 컨테이너가 해당 클래스 객체를 생성하도록 설정할 수 있다. 그런데 시스템을 구성하는 모든 클래스에 @Component를 할당하면 어떤 클래스가 어떤 클래스가 어떤 역할을 수행하는지 파악하기 어렵다.
- 스프링 프레임워크에서는 클래스들을 분류하기 위해서 @Component를 상속하여 다음과 같은

세 개의 어노테이션을 추가로 제공한다.

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

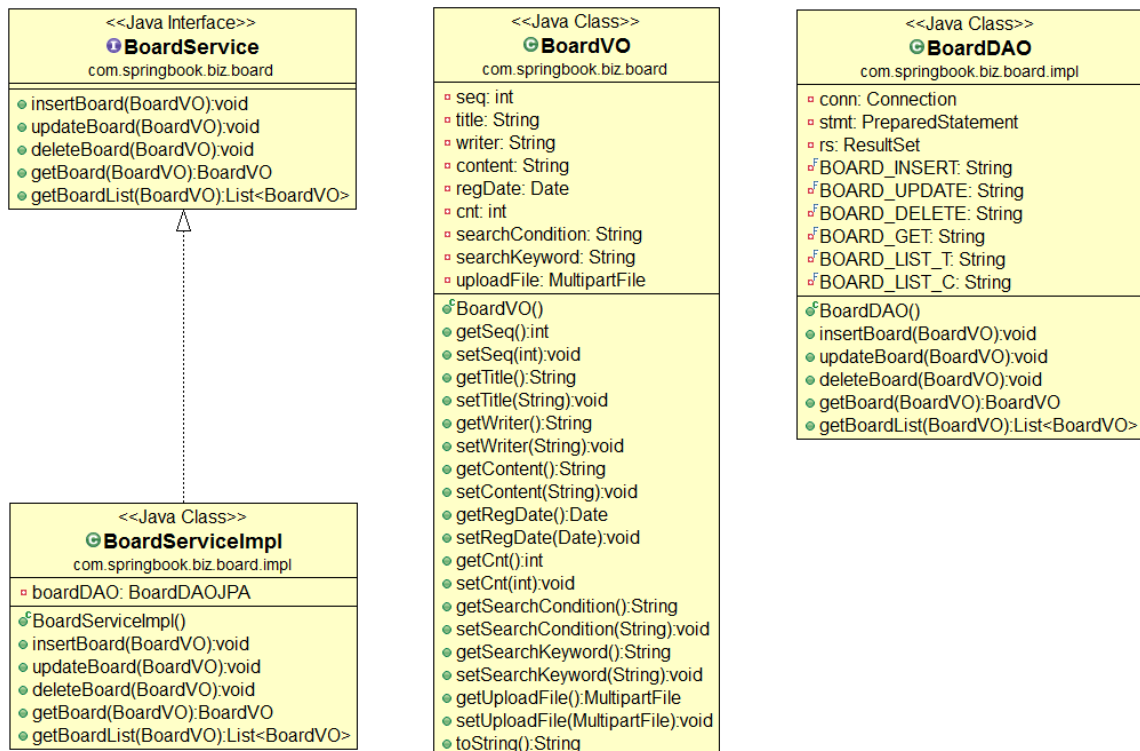
- 이 처럼 어노테이션을 나눈 이유는 단순히 해당 클래스를 분류하기 위해서 뿐만 아니라 MVC 아키텍처에서 @Controller는 컨트롤러 객체로 인식하도록 해주며, @Repository는 DB 연동 과정에서 발생하는 예외를 변환해주는 특별한 기능이 추가되어 있다.

B.6 비즈니스 컴포넌트 실습1

// 설정과정이 설명되어 있지 않아 코드가 갑자기 어렵다. 나중에 학습하도록 하자!

B.6.1 BoardService 컴포넌트 구조

- 프로젝트마다 조금씩은 다르겠지만 일반적으로 비즈니스 컴포넌트는 네 개의 자바 파일로 구성된다. 그리고 각 자바 파일을 작성하는 순서의 이름 규칙도 어느 정도는 정해져 있는 것이 일반적이다.
- 다음은 BOARD 테이블과 관련된 BoardService 컴포넌트에 대한 클래스 다이어그램이며, BoardVO, BoardDAO, BoardService, BoardServiceImpl 클래스로 구성되어 있다.



(1) 드라이버 내려받기

[BoardWeb/pom.xml]

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <project xmlns="http://maven.apache.org/POM/4.0.0"
03     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
04     xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/maven-
05 v4_0_0.xsd">
06     <modelVersion>4.0.0</modelVersion>
07     <groupId>com.springbook</groupId>
08     <artifactId>biz</artifactId>
09     <name>BoardWeb</name>
10     <packaging>war</packaging>
11     <version>1.0.0-BUILD-SNAPSHOT</version>
12     <properties>
13         <java-version>1.6</java-version>
14         <org.springframework-version>4.2.4.RELEASE</org.springframework-version>
15         <org.aspectj-version>1.6.10</org.aspectj-version>
16         <org.slf4j-version>1.6.6</org.slf4j-version>
17     </properties>
18     <dependencies>
19         <!-- JPA, 하이버네이트 -->
20         <dependency>
21             <groupId>org.hibernate</groupId>
22             <artifactId>hibernate-entitymanager</artifactId>
23             <version>5.1.0.Final</version>
24         </dependency>
25
26         <!-- H2 데이터베이스 -->
27         <dependency>
28             <groupId>com.h2database</groupId>
29             <artifactId>h2</artifactId>
30             <version>1.4.191</version>
31         </dependency>
32
33         <!-- Mybatis -->
34         <dependency>
35             <groupId>org.mybatis</groupId>
36             <artifactId>mybatis</artifactId>
37             <version>3.3.1</version>
38         </dependency>
39
40         <!-- Mybatis Spring -->
41         <dependency>
42             <groupId>org.mybatis</groupId>
43             <artifactId>mybatis-spring</artifactId>
44             <version>1.2.4</version>
45         </dependency>
46
47         <!-- Spring -->
48         <dependency>
49             <groupId>org.springframework</groupId>
50             <artifactId>spring-context</artifactId>
51             <version>${org.springframework-version}</version>
52             <exclusions>
53                 <!-- Exclude Commons Logging in favor of SLF4j -->
54                 <exclusion>
55                     <groupId>commons-logging</groupId>
56                     <artifactId>commons-logging</artifactId>
57                 </exclusion>
58             </exclusions>
59         </dependency>
60
61         <dependency>
62             <groupId>org.springframework</groupId>
63             <artifactId>spring-jdbc</artifactId>
64             <version>${org.springframework-version}</version>

```



```

65         </dependency>
66
67         <dependency>
68             <groupId>org.springframework</groupId>
69             <artifactId>spring-orm</artifactId>
70             <version>${org.springframework-version}</version>
71         </dependency>
72
73         <dependency>
74             <groupId>org.springframework</groupId>
75             <artifactId>spring-webmvc</artifactId>
76             <version>${org.springframework-version}</version>
77         </dependency>
78
79         <!-- DBCP -->
80         <dependency>
81             <groupId>commons-dbcp</groupId>
82             <artifactId>commons-dbcp</artifactId>
83             <version>1.4</version>
84         </dependency>
85
86         <!-- FileUpload -->
87         <dependency>
88             <groupId>commons-fileupload</groupId>
89             <artifactId>commons-fileupload</artifactId>
90             <version>1.3.1</version>
91         </dependency>
92
93         <!-- Jackson2 -->
94         <dependency>
95             <groupId>com.fasterxml.jackson.core</groupId>
96             <artifactId>jackson-databind</artifactId>
97             <version>2.7.2</version>
98         </dependency>
99
100
101         <!-- AspectJ -->
102         <dependency>
103             <groupId>org.aspectj</groupId>
104             <artifactId>aspectjrt</artifactId>
105             <version>${org.aspectj-version}</version>
106         </dependency>
107         <dependency>
108             <groupId>org.aspectj</groupId>
109             <artifactId>aspectjweaver</artifactId>
110             <version>1.8.8</version>
111         </dependency>
112
113         <!-- Logging -->
114         <dependency>
115             <groupId>org.slf4j</groupId>
116             <artifactId>slf4j-api</artifactId>
117             <version>${org.slf4j-version}</version>
118         </dependency>
119         <dependency>
120             <groupId>org.slf4j</groupId>
121             <artifactId>jcl-over-slf4j</artifactId>
122             <version>${org.slf4j-version}</version>
123             <scope>runtime</scope>
124         </dependency>
125         <dependency>
126             <groupId>org.slf4j</groupId>
127             <artifactId>slf4j-log4j12</artifactId>
128             <version>${org.slf4j-version}</version>
129             <scope>runtime</scope>

```

```

130         </dependency>
131     <dependency>
132         <groupId>log4j</groupId>
133         <artifactId>log4j</artifactId>
134         <version>1.2.15</version>
135         <exclusions>
136             <exclusion>
137                 <groupId>javax.mail</groupId>
138                 <artifactId>mail</artifactId>
139             </exclusion>
140             <exclusion>
141                 <groupId>javax.jms</groupId>
142                 <artifactId>jms</artifactId>
143             </exclusion>
144             <exclusion>
145                 <groupId>com.sun.jdmk</groupId>
146                 <artifactId>jmxtools</artifactId>
147             </exclusion>
148             <exclusion>
149                 <groupId>com.sun.jmx</groupId>
150                 <artifactId>jmxri</artifactId>
151             </exclusion>
152         </exclusions>
153         <scope>runtime</scope>
154     </dependency>
155
156     <!-- @Inject -->
157     <dependency>
158         <groupId>javax.inject</groupId>
159         <artifactId>javax.inject</artifactId>
160         <version>1</version>
161     </dependency>
162
163     <!-- Servlet -->
164     <dependency>
165         <groupId>javax.servlet</groupId>
166         <artifactId>servlet-api</artifactId>
167         <version>2.5</version>
168         <scope>provided</scope>
169     </dependency>
170     <dependency>
171         <groupId>javax.servlet.jsp</groupId>
172         <artifactId>jsp-api</artifactId>
173         <version>2.1</version>
174         <scope>provided</scope>
175     </dependency>
176     <dependency>
177         <groupId>javax.servlet</groupId>
178         <artifactId>jstl</artifactId>
179         <version>1.2</version>
180     </dependency>
181
182     <!-- Test -->
183     <dependency>
184         <groupId>junit</groupId>
185         <artifactId>junit</artifactId>
186         <version>4.7</version>
187         <scope>test</scope>
188     </dependency>
189
190     <!-- JDK11버전 이상에서 javax.xml.bind 사용시 -->
191     <dependency>
192         <groupId>javax.xml.bind</groupId>
193         <artifactId>jaxb-api</artifactId>
194         <version>2.3.1</version>

```

```

195         </dependency>
196     </dependency>
197     <groupId>com.sun.xml.bind</groupId>
198     <artifactId>jaxb-core</artifactId>
199     <version>2.3.0.1</version>
200 </dependency>
201 <dependency>
202     <groupId>com.sun.xml.bind</groupId>
203     <artifactId>jaxb-impl</artifactId>
204     <version>2.3.1</version>
205 </dependency>
206
207 </dependencies>
208 <build>
209     <plugins>
210         <plugin>
211             <artifactId>maven-eclipse-plugin</artifactId>
212             <version>2.9</version>
213             <configuration>
214                 <additionalProjectnatures>
215
216 <projectnature>org.springframework.ide.eclipse.core.springnature</projectnature>
217                 </additionalProjectnatures>
218                 <additionalBuildcommands>
219
220 <buildcommand>org.springframework.ide.eclipse.core.springbuilder</buildcommand>
221                 </additionalBuildcommands>
222                 <downloadSources>true</downloadSources>
223                 <downloadJavadocs>true</downloadJavadocs>
224             </configuration>
225         </plugin>
226         <plugin>
227             <groupId>org.apache.maven.plugins</groupId>
228             <artifactId>maven-compiler-plugin</artifactId>
229             <version>2.5.1</version>
230             <configuration>
231                 <source>1.6</source>
232                 <target>1.6</target>
233                 <compilerArgument>-Xlint:all</compilerArgument>
234                 <showWarnings>true</showWarnings>
235                 <showDeprecation>true</showDeprecation>
236             </configuration>
237         </plugin>
238         <plugin>
239             <groupId>org.codehaus.mojo</groupId>
240             <artifactId>exec-maven-plugin</artifactId>
241             <version>1.2.1</version>
242             <configuration>
243                 <mainClass>org.test.int1.Main</mainClass>
244             </configuration>
245         </plugin>
246     </plugins>
247 </build>
248 </project>

```

B.6.2 Value Object 클래스 작성

- VO(Value Object) 클래스는 레이어와 레이어 사이에서 관련된 데이터를 한꺼번에 주고받을 목적으로 사용하는 클래스이다.
- DTO(Data Transfer Object)라 하기도 하는데, 데이터 전달을 목적으로 사용하는 객체이므로

결국 같은 의미의 용어라고 생각하면 된다.

[WEB-INF/sql/ddl.sql]

```
DROP TABLE USERS;
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');

DROP TABLE BOARD;
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, '가임인사', '관리자', '잘 부탁드립니다');

SELECT * FROM BOARD;
SELECT * FROM USERS;
```

[com.springbook.biz.board.BoardVO.java]

```
01 package com.springbook.biz.board;
02
03 import java.util.Date;
04
05 import javax.persistence.Entity;
06 import javax.persistence.GeneratedValue;
07 import javax.persistence.Id;
08 import javax.persistence.Table;
09 import javax.persistence.Temporal;
10 import javax.persistence.TemporalType;
11 import javax.persistence.Transient;
12
13 import org.springframework.web.multipart.MultipartFile;
14
15 // VO(Value Object)
16 @Entity
17 @Table(name="BOARD")
18 public class BoardVO {
19     @Id
20     @GeneratedValue
21     private int seq;
22     private String title;
23     private String writer;
24     private String content;
25     @Temporal(TemporalType.DATE)
26     private Date regDate = new Date();
27     private int cnt;
28     @Transient
29     private String searchCondition;
30     @Transient
31     private String searchKeyword;
32     @Transient
```

```
33     private MultipartFile uploadFile;
34
35     // Getter/Setter 메소드
36     public int getSeq() {
37         return seq;
38     }
39
40     public void setSeq(int seq) {
41         this.seq = seq;
42     }
43
44     public String getTitle() {
45         return title;
46     }
47
48     public void setTitle(String title) {
49         this.title = title;
50     }
51
52     public String getWriter() {
53         return writer;
54     }
55
56     public void setWriter(String writer) {
57         this.writer = writer;
58     }
59
60     public String getContent() {
61         return content;
62     }
63
64     public void setContent(String content) {
65         this.content = content;
66     }
67
68     public Date getRegDate() {
69         return regDate;
70     }
71
72     public void setRegDate(Date regDate) {
73         this.regDate = regDate;
74     }
75
76     public int getCnt() {
77         return cnt;
78     }
79
80     public void setCnt(int cnt) {
81         this.cnt = cnt;
82     }
83
84     public String getSearchCondition() {
85         return searchCondition;
86     }
87
88     public void setSearchCondition(String searchCondition) {
89         this.searchCondition = searchCondition;
90     }
91
92     public String getSearchKeyword() {
93         return searchKeyword;
94     }
95
96     public void setSearchKeyword(String searchKeyword) {
97         this.searchKeyword = searchKeyword;
98     }
99 }
```

```

98         }
99
100        public MultipartFile getUploadFile() {
101            return uploadFile;
102        }
103
104        public void setUploadFile(MultipartFile uploadFile) {
105            this.uploadFile = uploadFile;
106        }
107
108        @Override
109        public String toString() {
110            return "BoardVO [seq=" + seq + ", title=" + title + ", writer="
111                + writer + ", content=" + content + ", regDate=" + regDate
112                + ", cnt=" + cnt + " ]";
113        }
114    }

```

B.6.3 DAO 클래스 작성

(1) 드라이버 내려받기

```

// pom.xml의 일부

<!-- H2 데이터베이스 -->
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
    <version>1.4.191</version>
</dependency>

```

(2) JDBC Utility 클래스

[com.springbook.biz.common.JDBCUtil.java]

```

01 package com.springbook.biz.common;
02
03 import java.sql.Connection;
04 import java.sql.DriverManager;
05 import java.sql.PreparedStatement;
06 import java.sql.ResultSet;
07
08 public class JDBCUtil {
09     public static Connection getConnection() {
10         try {
11             Class.forName("org.h2.Driver");
12             return DriverManager.getConnection("jdbc:h2:tcp://localhost/~:/test", "sa", "");
13         } catch (Exception e) {
14             e.printStackTrace();
15         }
16         return null;
17     }
18
19     public static void close(PreparedStatement stmt, Connection conn) {
20         if(stmt != null) {
21             try {
22                 if(!stmt.isClosed()) stmt.close();
23             } catch (Exception e) {

```

```

24         e.printStackTrace();
25     } finally {
26         stmt = null;
27     }
28 }
29
30 if(conn != null) {
31     try {
32         if(!conn.isClosed()) conn.close();
33     } catch (Exception e) {
34         e.printStackTrace();
35     } finally {
36         conn = null;
37     }
38 }
39 }
40
41 public static void close(ResultSet rs, PreparedStatement stmt, Connection conn) {
42     if(rs != null) {
43         try {
44             if(!rs.isClosed()) rs.close();
45         } catch (Exception e) {
46             e.printStackTrace();
47         } finally {
48             rs = null;
49         }
50     }
51
52     if(stmt != null) {
53         try {
54             if(!stmt.isClosed()) stmt.close();
55         } catch (Exception e) {
56             e.printStackTrace();
57         } finally {
58             stmt = null;
59         }
60     }
61
62     if(conn != null) {
63         try {
64             if(!conn.isClosed()) conn.close();
65         } catch (Exception e) {
66             e.printStackTrace();
67         } finally {
68             conn = null;
69         }
70     }
71 }
72 }

```

(3) DAO 클래스 작성

- 이 클래스 객체를 스프링 컨테이너가 생성할 수 있도록 클래스 선언부에 @Repository 어노테이션을 설정한다.
- 물론 @Component를 사용한다고 해서 문제가 발생하지는 않는다. 다만 DAO 기능의 클래스에는 @Component보다 @Repository를 사용하는 것이 여러 이유에서 적합하다.

```
[com.springbook.biz.board.BoardDAO.java]
```

```
01 package com.springbook.biz.board;
```

```

02
03 import java.sql.Connection;
04 import java.sql.PreparedStatement;
05 import java.sql.ResultSet;
06 import java.util.ArrayList;
07 import java.util.List;
08
09 import org.springframework.stereotype.Repository;
10
11 import com.springbook.biz.common.JDBCUtil;
12
13 // DAO(Data Access Object)
14 @Repository("boardDAO")
15 public class BoardDAO {
16     // JDBC 관련 변수
17     private Connection conn = null;
18     private PreparedStatement stmt = null;
19     private ResultSet rs = null;
20
21     // SQL 명령어들
22     private final String BOARD_INSERT = "insert into board(seq, title, writer, content)
23 values((select nvl(max(seq), 0)+1 from board),?,?,?)";
24     private final String BOARD_UPDATE = "update board set title=?, content=? where seq=?";
25     private final String BOARD_DELETE = "delete board where seq=?";
26     private final String BOARD_GET = "select * from board where seq=?";
27     private final String BOARD_LIST_T = "select * from board where title like '%||?||%' order by
28 seq desc";
29     private final String BOARD_LIST_C = "select * from board where content like '%||?||%' order
30 by seq desc";
31
32     // CRUD 기능의 메소드 구현
33     // 글 등록
34     public void insertBoard(BoardVO vo) {
35         System.out.println("==> JDBC로 insertBoard() 기능 처리");
36         try {
37             conn = JDBCUtil.getConnection();
38             stmt = conn.prepareStatement(BOARD_INSERT);
39             stmt.setString(1, vo.getTitle());
40             stmt.setString(2, vo.getWriter());
41             stmt.setString(3, vo.getContent());
42             stmt.executeUpdate();
43         } catch (Exception e) {
44             e.printStackTrace();
45         } finally {
46             JDBCUtil.close(stmt, conn);
47         }
48     }
49
50     // 글 수정
51     public void updateBoard(BoardVO vo) {
52         System.out.println("==> JDBC로 updateBoard() 기능 처리");
53         try {
54             conn = JDBCUtil.getConnection();
55             stmt = conn.prepareStatement(BOARD_UPDATE);
56             stmt.setString(1, vo.getTitle());
57             stmt.setString(2, vo.getContent());
58             stmt.setInt(3, vo.getSeq());
59             stmt.executeUpdate();
60         } catch (Exception e) {
61             e.printStackTrace();
62         } finally {
63             JDBCUtil.close(stmt, conn);
64         }
65     }
66

```



```

67 // 글 삭제
68 public void deleteBoard(BoardVO vo) {
69     System.out.println("==> JDBC로 deleteBoard() 기능 처리");
70     try {
71         conn = JDBCUtil.getConnection();
72         stmt = conn.prepareStatement(BOARD_DELETE);
73         stmt.setInt(1, vo.getSeq());
74         stmt.executeUpdate();
75     } catch (Exception e) {
76         e.printStackTrace();
77     } finally {
78         JDBCUtil.close(stmt, conn);
79     }
80 }
81
82 // 글 상세 조회
83 public BoardVO getBoard(BoardVO vo) {
84     System.out.println("==> JDBC로 getBoard() 기능 처리");
85     BoardVO board = null;
86     try {
87         conn = JDBCUtil.getConnection();
88         stmt = conn.prepareStatement(BOARD_GET);
89         stmt.setInt(1, vo.getSeq());
90         rs = stmt.executeQuery();
91         if (rs.next()) {
92             board = new BoardVO();
93             board.setSeq(rs.getInt("SEQ"));
94             board.setTitle(rs.getString("TITLE"));
95             board.setWriter(rs.getString("WRITER"));
96             board.setContent(rs.getString("CONTENT"));
97             board.setRegDate(rs.getDate("REGDATE"));
98             board.setCnt(rs.getInt("CNT"));
99         }
100     } catch (Exception e) {
101         e.printStackTrace();
102     } finally {
103         JDBCUtil.close(rs, stmt, conn);
104     }
105     return board;
106 }
107
108 // 글 목록 조회
109 public List<BoardVO> getBoardList(BoardVO vo) {
110     System.out.println("==> JDBC로 getBoardList() 기능 처리");
111     List<BoardVO> boardList = new ArrayList<BoardVO>();
112     try {
113         conn = JDBCUtil.getConnection();
114         if (vo.getSearchCondition().equals("TITLE")) {
115             stmt = conn.prepareStatement(BOARD_LIST_T);
116         } else if (vo.getSearchCondition().equals("CONTENT")) {
117             stmt = conn.prepareStatement(BOARD_LIST_C);
118         }
119         stmt.setString(1, vo.getSearchKeyword());
120         rs = stmt.executeQuery();
121         while (rs.next()) {
122             BoardVO board = new BoardVO();
123             board.setSeq(rs.getInt("SEQ"));
124             board.setTitle(rs.getString("TITLE"));
125             board.setWriter(rs.getString("WRITER"));
126             board.setContent(rs.getString("CONTENT"));
127             board.setRegDate(rs.getDate("REGDATE"));
128             board.setCnt(rs.getInt("CNT"));
129             boardList.add(board);
130         }
131     } catch (Exception e) {

```

```

132             e.printStackTrace();
133         } finally {
134             JDBCUtil.close(rs, stmt, conn);
135         }
136         return boardList;
137     }
138 }
139 }

```

B.6.4 Service 인터페이스 작성

- DAO 클래스를 작성했으면 이제 DAO 클래스에서 <Alt + Shift + T> 단축키를 이용하여 "Extract Interface"를 선택하고 BoardService 인터페이스를 작성한다.
- 이때 인터페이스가 만들어지는 동시에 BoardDAO 클래스에는 implements와 @Override 코드가 작동으로 설정되는데, 이 코드는 삭제해야 한다.

[com.springbook.biz.board.BoardService.java]

```

01 package com.springbook.biz.board;
02
03 import java.util.List;
04
05 public interface BoardService {
06
07     // CRUD 기능의 메소드
08     // 글 등록
09     void insertBoard(BoardVO vo);
10
11     // 글 수정
12     void updateBoard(BoardVO vo);
13
14     // 글 삭제
15     void deleteBoard(BoardVO vo);
16
17     // 글 상세 조회
18     BoardVO getBoard(BoardVO vo);
19
20     // 글 목록 조회
21     List<BoardVO> getBoardList(BoardVO vo);
22
23 }

```

B.6.5 Service 구현 클래스 작성

[com.springbook.biz.board.BoardServiceImpl.java]

```

01 package com.springbook.biz.board;
02
03 import java.util.List;
04
05 import org.springframework.beans.factory.annotation.Autowired;
06 import org.springframework.stereotype.Service;
07
08 @Service("boardService")
09 public class BoardServiceImpl implements BoardService {
10     @Autowired

```

```

11         private BoardDAO boardDAO;
12
13         @Override
14         public void insertBoard(BoardVO vo) {
15             boardDAO.insertBoard(vo);
16         }
17
18         @Override
19         public void updateBoard(BoardVO vo) {
20             boardDAO.updateBoard(vo);
21         }
22
23         @Override
24         public void deleteBoard(BoardVO vo) {
25             boardDAO.deleteBoard(vo);
26         }
27
28         @Override
29         public BoardVO getBoard(BoardVO vo) {
30             return boardDAO.getBoard(vo);
31         }
32
33         @Override
34         public List<BoardVO> getBoardList(BoardVO vo) {
35             return boardDAO.getBoardList(vo);
36         }
37     }
38
39 }
40
41 }

```

B.6.6 BoardService 컴포넌트 테스트

(1) 스프링 설정 파일 수정

[src/main/resources/applicationContext.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:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       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-4.2.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">

    <!-- lab begin -->
    <context:component-scan base-package="com.springbook.biz">
    </context:component-scan>
    <!-- lab end -->

</beans>

```

(2) 클라이언트 작성 및 실행

[BoardWeb/src/main/java/lab/BoardServiceClient.java]

```
package lab;

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 구동한다.
        AbstractApplicationContext container = new
        GenericXmlApplicationContext("applicationContext.xml");

        // 2. Spring 컨테이너로부터 BoardServiceImpl 객체를 Lookup한다.
        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.....");
        vo.setSearchCondition("TITLE");
        vo.setSearchKeyword("임시");
        boardService.insertBoard(vo);

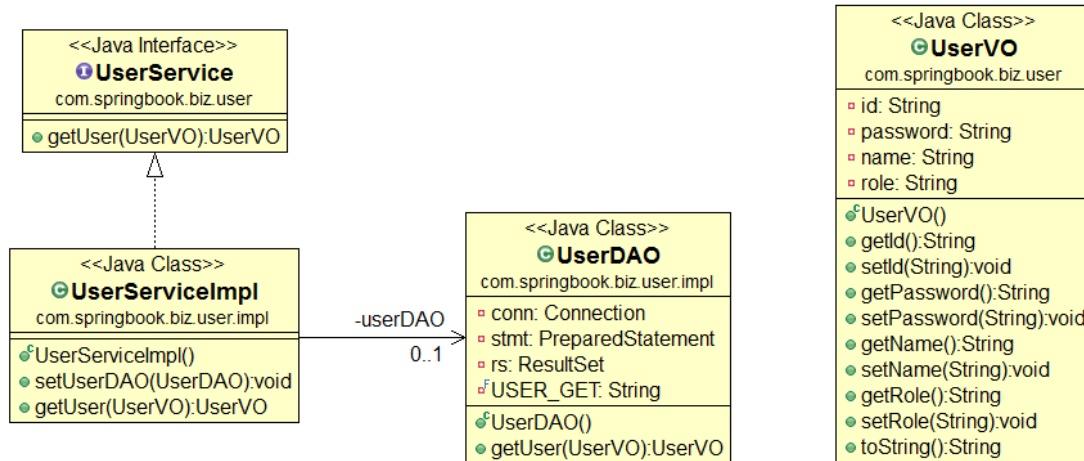
        // 4. 글 목록 검색 기능 테스트
        List<BoardVO> boardList = boardService.getBoardList(vo);
        for (BoardVO board : boardList) {
            System.out.println("---> " + board.toString());
        }
        // 5. Spring 컨테이너 종료
        container.close();
    }
}
```

// BoardServiceClient.java의 실행결과

```
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [boardServiceContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@6f2b958e: startup date [Sun Nov 04 14:01:27
KST 2018]; root of context hierarchy
INFO : org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor - JSR-330
'javax.inject.Inject' annotation found and supported for autowiring
==> JDBC로 insertBoard() 기능 처리
==> JDBC로 getBoardList() 기능 처리
---> BoardVO [seq=1, title=임시 제목, writer=홍길동, content=임시 내용....., regDate=2018-11-04,
cnt=0]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@6f2b958e: startup date [Sun Nov 04 14:01:27
KST 2018]; root of context hierarchy
```


B.7 비즈니스 컴포넌트 실습2

B.7.1 UserService 컴포넌트 구조



B.7.2 Value Object 클래스 작성

- VO(Value Object) 클래스를 작성하기 위해서 USERS 테이블의 구조를 확인한다.

// WEB-INF/sql/ddl.sql의 일부

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

[com.springbook.biz.user.UserVO.java]

```
01 package com.springbook.biz.user;
02
03 // VO(Value Object)
04 public class UserVO {
05     private String id;
06     private String password;
07     private String name;
08     private String role;
09
10     public String getId() {
11         return id;
12     }
13     public void setId(String id) {
14         this.id = id;
15     }
16     public String getPassword() {
17         return password;
18     }
19     public void setPassword(String password) {
20         this.password = password;
21     }
22 }
```

```

22     public String getName() {
23         return name;
24     }
25     public void setName(String name) {
26         this.name = name;
27     }
28     public String getRole() {
29         return role;
30     }
31     public void setRole(String role) {
32         this.role = role;
33     }
34     @Override
35     public String toString() {
36         return "UserVO [id=" + id + ", password=" + password + ", name=" + name
37             + ", role=" + role + "]\n";
38     }
39 }

```

B.7.3 DAO 클래스 작성

[com.springbook.biz.user.UserDAO.java]

```

package com.springbook.biz.user.impl;

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

import org.springframework.stereotype.Repository;

import com.springbook.biz.common.JDBCUtil;
import com.springbook.biz.user.UserVO;

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

    // SQL 명령어들
    private final String USER_GET = "select * from users where id=? and password=?";

    // CRUD 기능의 메소드 구현
    // 회원 등록
    public UserVO getUser(UserVO vo){
        UserVO user = null;
        try {
            System.out.println("=> JDBC로 getUser() 기능 처리");
            conn = JDBCUtil.getConnection();
            stmt = conn.prepareStatement(USER_GET);
            stmt.setString(1, vo.getId());
            stmt.setString(2, vo.getPassword());
            rs = stmt.executeQuery();
            if(rs.next()){
                user = new UserVO();
                user.setId(rs.getString("ID"));
                user.setPassword(rs.getString("PASSWORD"));
                user.setName(rs.getString("NAME"));
                user.setRole(rs.getString("ROLE"));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```

```

    }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(rs, stmt, conn);
    }
    return user;
}
}

```

B.7.4 Service 인터페이스 작성

[com.springbook.biz.user.UserService.java]

```

package com.springbook.biz.user;

public interface UserService {

    // CRUD 기능의 메소드
    // 회원 등록
    UserVO getUser(UserVO vo);

}

```

B.7.5 Service 구현 클래스 작성

[com.springbook.biz.user.impl.UserServiceImpl.java]

```

package com.springbook.biz.user.impl;

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

import com.springbook.biz.user.UserService;
import com.springbook.biz.user.UserVO;

@Service("userService")
public class UserServiceImpl implements UserService {
    @Autowired
    private UserDAO userDAO;

    public void setUserDAO(UserDAO userDAO) {
        this.userDAO = userDAO;
    }

    public UserVO getUser(UserVO vo) {
        return userDAO.getUser(vo);
    }
}

```

B.7.6 UserService 컴포넌트 테스트

(1) 스프링 설정 파일 수정

[src/main/resources/applicationContext.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:aop="http://www.springframework.org/schema/aop"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:p="http://www.springframework.org/schema/p"
       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-4.2.xsd
                           http://www.springframework.org/schema/aop
                           http://www.springframework.org/schema/aop/spring-aop-4.2.xsd
                           http://www.springframework.org/schema/tx
                           http://www.springframework.org/schema/tx/spring-tx-4.2.xsd">

    <!-- lab begin -->
    <context:component-scan base-package="com.springbook.biz">
    </context:component-scan>
    <!-- lab end -->

</beans>

```

(2) 클라이언트 작성 및 실행

[BoardWeb/src/main/java/lab/UserServiceClient.java]

```

package com.springbook.biz.user;

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

public class UserServiceClient {
    public static void main(String[] args) {
        // 1. Spring 컨테이너를 구동한다.
        AbstractApplicationContext container =
            new GenericXmlApplicationContext("applicationContext.xml");

        // 2. Spring 컨테이너로부터 UserServiceImpl 객체를 Lookup 한다.
        UserService userService =
            (UserService) container.getBean("userService");

        // 3. 로그인 기능 테스트
        UserVO vo = new UserVO();
        vo.setId("test");
        vo.setPassword("test123");

        UserVO user = userService.getUser(vo);
        if(user != null) {
            System.out.println(user.getName() + "님 환영합니다.");
        } else {
            System.out.println("로그인 실패");
        }

        // 4. Spring 컨테이너를 종료한다.
        container.close();
    }
}

```

// UserServiceClient.java의 실행결과

```
INFO : org.springframework.beans.factory.xml.XmlBeanDefinitionReader - Loading XML bean definitions from
class path resource [applicationContext.xml]
INFO : org.springframework.context.support.GenericXmlApplicationContext - Refreshing
org.springframework.context.support.GenericXmlApplicationContext@6f2b958e: startup date [Wed Jun 26 22:20:16
KST 2019]; root of context hierarchy
INFO : org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor - JSR-330
'javax.inject.Inject' annotation found and supported for autowiring
==> JDBC로 getUser() 기능 처리
관리자님 환영합니다.
INFO : org.springframework.context.support.GenericXmlApplicationContext - Closing
org.springframework.context.support.GenericXmlApplicationContext@6f2b958e: startup date [Wed Jun 26 22:20:16
KST 2019]; root of context hierarchy
```