



박 경 태

comsi.java@gmail.com

고급 자바 프로그래밍 : STS를 이용한 Spring 프로그래밍

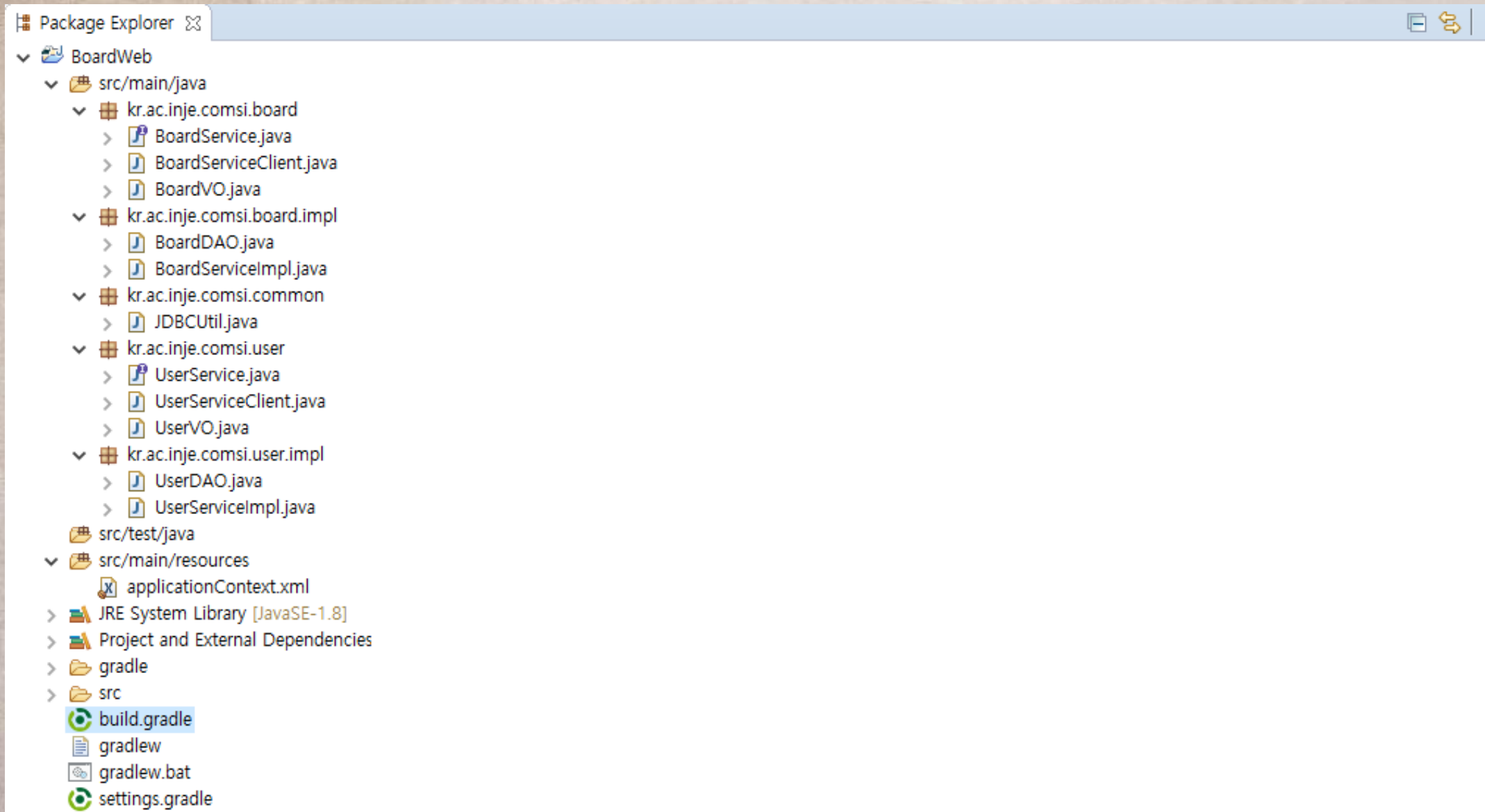
강의 내용

순서	내 용
1	<ul style="list-style-type: none">• Spring IoC를 이용한 비즈니스 컴포넌트 만들기
2	<ul style="list-style-type: none">• Spring AOP(Aspect Oriented Programming)를 이용한 공통 서비스 만들기• Spring DAO(Data Access Object)를 이용한 데이터베이스 연동 및 트랜잭션 처리
3	<ul style="list-style-type: none">• Spring MVC를 이용한 MVC 아키텍처 적용하기
4	<ul style="list-style-type: none">• Spring MVC의 부가 기능 사용하기(파일 업로드, 다국어, 예외 처리 등)
5	<ul style="list-style-type: none">• Spring과 MyBatis 연동하기• Spring과 JPA 연동하기

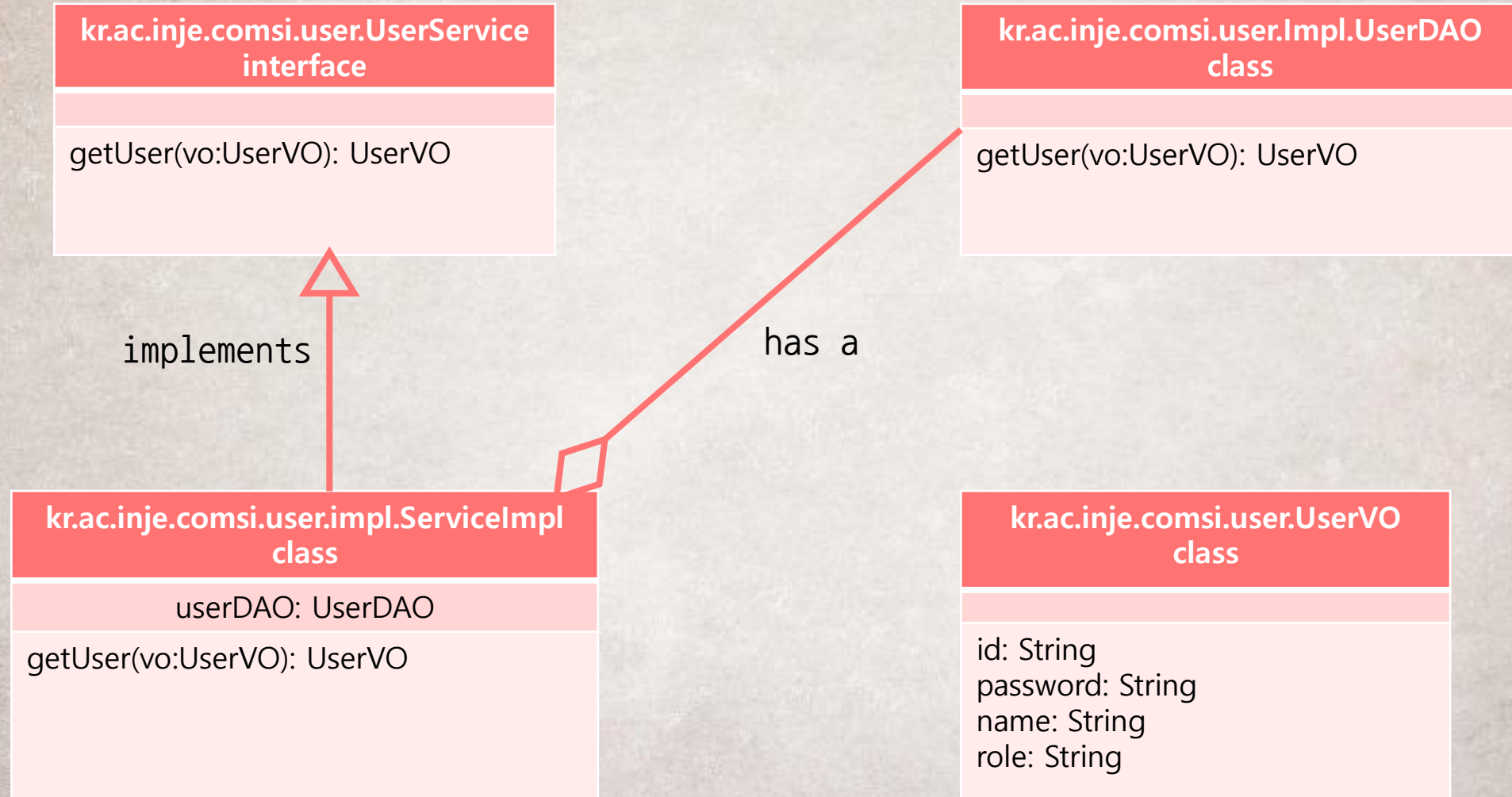
7. 비즈니스 컴포넌트 실습2

- BoardService 컴포넌트를 만들었으면 회원 정보 관리하는 UserService 컴포넌트를 추가로 개발해보자
- BoardService와 동일한 절차로 개발
- Setter 인젝션으로 의존성 주입을 처리한 후 어노테이션으로 변경하는 과정을 사용

프로젝트 구조



1. UserService 컴포넌트 구조



UserService 클래스 다이어그램

2.UserVO.java - 유저 정보 VO 클래스

UserVO.java

```
1 package kr.ac.inje.comsi.user;
2
3 // VO(Value Object)
4 public class UserVO {
5     private String id;
6     private String password;
7     private String name;
8     private String role;
9
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    public String getName() {
23        return name;
24    }
25 }
```

```
26- public void setName(String name) {  
27     this.name = name;  
28 }  
29- public String getRole() {  
30     return role;  
31 }  
32- public void setRole(String role) {  
33     this.role = role;  
34 }  
35- @Override  
36 public String toString() {  
37     return "UserVO [id=" + id + ", password=" + password  
38         + ", name=" + name + ", role=" + role + "];"  
39 }  
40  
41 }  
42  
43
```

3.DAO 클래스 작성 - JDBCUtil 클래스를 이용

```
UserServiceImpl.java  applicationContext.xml  UserServiceClient.java  UserDao.java ✕
1 package kr.ac.inje.comsi.user.impl;
2
3 import java.sql.Connection;
9
10 // DAO(Data Access Object)
11 public class UserDao {
12
13     // JDBC 관련 변수
14     private Connection conn = null;
15     private PreparedStatement stmt = null;
16     private ResultSet rset = null;
17
18     // SQL 명령어들
19     private final String USER_GET = "select * from users where id=? and password=?";
20
21     // CRUD 기능의 메소드
22     // 회원 등록
23 public UserVO getUser(UserVO vo){
24     UserVO user = null;
25     try{
26         System.out.println("==> JDBC로 getUser() 기능 처리");
27         conn = JDBCUtil.getConnection();
28         stmt = conn.prepareStatement(USER_GET);
29         stmt.setString(1, vo.getId());
30         stmt.setString(2, vo.getPassword());
31         rset = stmt.executeQuery();
```



```

32
33     if (rset.next()){
34         user = new UserVO();
35         user.setId(rset.getString("ID"));
36         user.setName(rset.getString("NAME"));
37         user.setPassword(rset.getString("PASSWORD"));
38         user.setRole(rset.getString("ROLE"));
39     }
40 }catch(Exception e){
41     e.printStackTrace();
42 }finally{
43     JDBCUtil.close(rset, stmt, conn);
44 }
45
46     return user;
47 }
48
49 }
50
51 |

```

- UserDAO 클래스는 정상적인 <bean> 등록으로 객체를 생성하고 어노테이션은 설정하지 않는다.

4. Service 인터페이스 작성

UserVO.java UserDao.java UserService.java

```
1 package kr.ac.inje.comsi.user;
2
3 public interface UserService {
4
5     // CRUD 기능의 메소드
6     // 회원 등록
7     UserVO getUser(UserVO vo);
8
9 }
```

5. Service 구현 클래스 작성

```
UserVO.java  UserDao.java  UserService.java  UserServiceImpl.java ✕
1 package kr.ac.inje.comsi.user.impl;
2
3 import kr.ac.inje.comsi.user.UserService;
4 import kr.ac.inje.comsi.user.UserVO;
5
6 public class UserServiceImpl implements UserService {
7
8     private UserDao userDao;
9
10    public void setUserDAO(UserDao userDao) {
11        this.userDao = userDao;
12    }
13
14    @Override
15    public UserVO getUser(UserVO vo) {
16        return userDao.getUser(vo);
17    }
18
19 }
20
21
```

- UserDao 객체를 이용해 DB 연동 처리
- UserServiceImpl 클래스에는 setter 인젝션 처리를 위한 Setter 메소드가 추가됨.

6. UserService 컴포넌트 테스트

```
UserVO.java  UserDao.java  UserService.java  UserServiceImpl.java  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="kr.ac.inje.comsi"></context:component-scan>
12
13     <bean id="userService" class="kr.ac.inje.comsi.user.impl.UserServiceImpl">
14         <property name="userDAO" ref="userDAO"></property>
15     </bean>
16
17     <bean id="userDAO" class="kr.ac.inje.comsi.user.impl.UserDAO"></bean>
18
19 </beans>
20
21
22
```

- applicationContext.xml에 UserServiceImpl와 UserDAO 클래스를 <bean> 등록
- UserServiceImpl 클래스에서 UserDAO객체를 의존성 주입하기 위한 <property> 설정 사용

UserServiceClient.java - 클라이언트 프로그램 수정

```
UserServiceImpl.java  applicationContext.xml  UserServiceClient.java  BoardDAO.java  UserDao.java
1 package kr.ac.inje.comsi.user;
2
3 import org.springframework.context.support.AbstractApplicationContext;
4 import org.springframework.context.support.GenericXmlApplicationContext;
5
6 public class UserServiceClient {
7
8     public static void main(String[] args) {
9
10         // 1. Spring 컨테이너를 구동
11         AbstractApplicationContext container = new GenericXmlApplicationContext("applicationContext.xml");
12
13         // 2. Spring 컨테이너로부터 UserServiceImpl 객체를 Lookup
14         UserService userService = (UserService) container.getBean("userService");
15
16         // 3. 로그인 기능 테스트
17         UserVO vo = new UserVO();
18         vo.setId("test");
19         vo.setPassword("test123");
20
21         UserVO user = userService.getUser(vo);
22         if (user != null){
23             System.out.println(user.getName() + "님 환영합니다.");
24         }else{
25             System.out.println("로그인 실패");
26         }
27
28         // 4. Spring 컨테이너 종료
29         container.close();
30     }
31 }
32
33
```

실행 결과



The screenshot shows an IDE console window with the following content:

```
<terminated> BoardWeb [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (2017. 5. 28. 오후 11:19:11)
5월 28, 2017 11:19:11 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 28, 2017 11:19:12 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Sun May 28 23:19:12
==> JDBC로 getUser() 기능 처리
관리자님 환영합니다.
5월 28, 2017 11:19:12 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Sun May 28 23:19:12 KST
```


setter 인젝션을
annotation으로 변경

7. 어노테이션 적용 - Setter 인젝션 설정을 수정

```
UserServiceImpl.java  applicationContext.xml  UserServiceClient.java  UserDao.java
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="kr.ac.inje.comsi"></context:component-scan>
12      <!--
13      <bean id="userService" class="kr.ac.inje.comsi.user.impl.UserServiceImpl">
14          <property name="userDAO" ref="userDAO"></property>
15      </bean>
16
17      <bean id="userDAO" class="kr.ac.inje.comsi.user.impl.UserDAO"></bean>
18      -->
19  </beans>
20
21
22
```

주석처리

UserDAO.java 수정



The screenshot shows an IDE with four tabs: UserServiceImpl.java, applicationContext.xml, UserServiceClient.java, and UserDAO.java. The UserDAO.java file is open and contains the following code:

```
1 package kr.ac.inje.comsi.user.impl;
2
3 import java.sql.Connection;
4 import java.sql.PreparedStatement;
5 import java.sql.ResultSet;
6
7 import org.springframework.stereotype.Repository;
8
9 import kr.ac.inje.comsi.common.JDBCUtil;
10 import kr.ac.inje.comsi.user.UserVO;
11
12 // DAO(Data Access Object)
13 @Repository("userDAO")
14 public class UserDAO {
15     // JDBC 관련 변수
16     private Connection conn = null;
17     private PreparedStatement stmt = null;
18     private ResultSet rset = null;
19
20     // SQL 명령어들
21     private final String USER_GET = "select * from users where id=? and password=?";
```

Two red arrows point to specific lines of code with Korean annotations:

- An arrow points to line 7, `import org.springframework.stereotype.Repository;`, with the annotation "추가됨" (Added).
- An arrow points to line 13, `@Repository("userDAO")`, with the annotation "UserDAO 객체 생성" (UserDAO object creation).

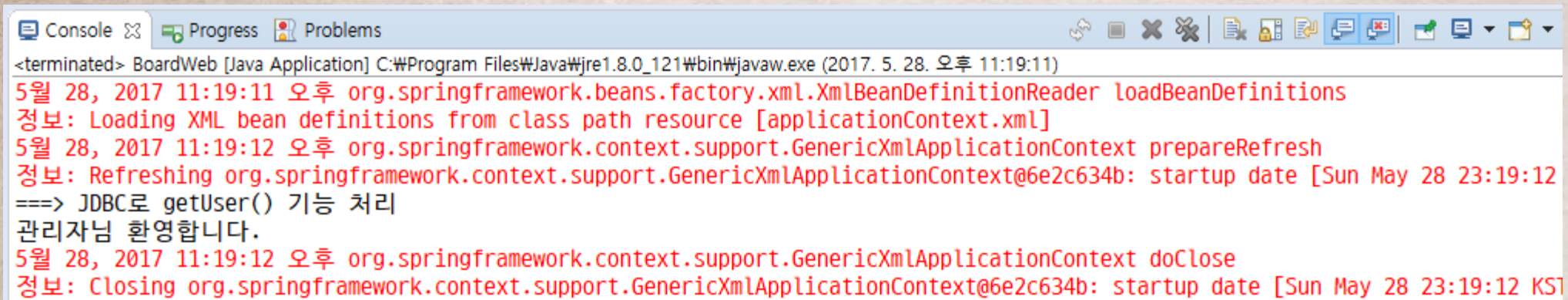
UserServiceImpl.java 수정하여 어노테이션 추가

```
UserServiceImpl.java applicationContext.xml UserServiceClient.java UserDAO.java
1 package kr.ac.inje.comsi.user.impl;
2
3 import org.springframework.beans.factory.annotation.Autowired;
4 import org.springframework.stereotype.Service;
5
6 import kr.ac.inje.comsi.user.UserService;
7 import kr.ac.inje.comsi.user.UserVO;
8
9 @Service("userService")
10 public class UserServiceImpl implements UserService {
11     @Autowired
12     private UserDAO userDAO;
13
14     public void setUserDAO(UserDAO userDAO) {
15         this.userDAO = userDAO;
16     }
17
18     @Override
19     public UserVO getUser(UserVO vo) {
20         return userDAO.getUser(vo);
21     }
22
23 }
24
25
```

이름이 userService인 Service객체 생성

UserDAO 객체 주입

실행 결과




The screenshot shows an IDE console window with the following content:

```
<terminated> BoardWeb [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (2017. 5. 28. 오후 11:19:11)
5월 28, 2017 11:19:11 오후 org.springframework.beans.factory.xml.XmlBeanDefinitionReader loadBeanDefinitions
정보: Loading XML bean definitions from class path resource [applicationContext.xml]
5월 28, 2017 11:19:12 오후 org.springframework.context.support.GenericXmlApplicationContext prepareRefresh
정보: Refreshing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Sun May 28 23:19:12
==> JDBC로 getUser() 기능 처리
관리자님 환영합니다.
5월 28, 2017 11:19:12 오후 org.springframework.context.support.GenericXmlApplicationContext doClose
정보: Closing org.springframework.context.support.GenericXmlApplicationContext@6e2c634b: startup date [Sun May 28 23:19:12 KST
```

데이터베이스 구축

H2-http://h2database.com



[Translate](#)

Search:

[Home](#)
[Download](#)
[Cheat Sheet](#)

[Documentation](#)
[Quickstart](#)
[Installation](#)
[Tutorial](#)
[Features](#)
[Performance](#)
[Advanced](#)

[Reference](#)
[SQL Grammar](#)
[Functions](#)
[Data Types](#)
[Javadoc](#)
[PDF \(1 MB\)](#)

[Support](#)
[FAQ](#)
[Error Analyzer](#)
[Google Group \(English\)](#)
[Google Group \(Japanese\)](#)
[Google Group \(Chinese\)](#)

H2 Database Engine

Welcome to H2, the Java SQL database. The main features of H2 are:

- Very fast, open source, JDBC API
- Embedded and server modes; in-memory databases
- Browser based Console application
- Small footprint: around 1.5 MB jar file size

Download

Version 1.4.195 (2017-04-23)

- [Windows Installer \(5 MB\)](#)
- [All Platforms \(zip, 8 MB\)](#)
- [All Downloads](#)

Support

[Stack Overflow \(tag H2\)](#)

[Google Group English, Japanese](#)

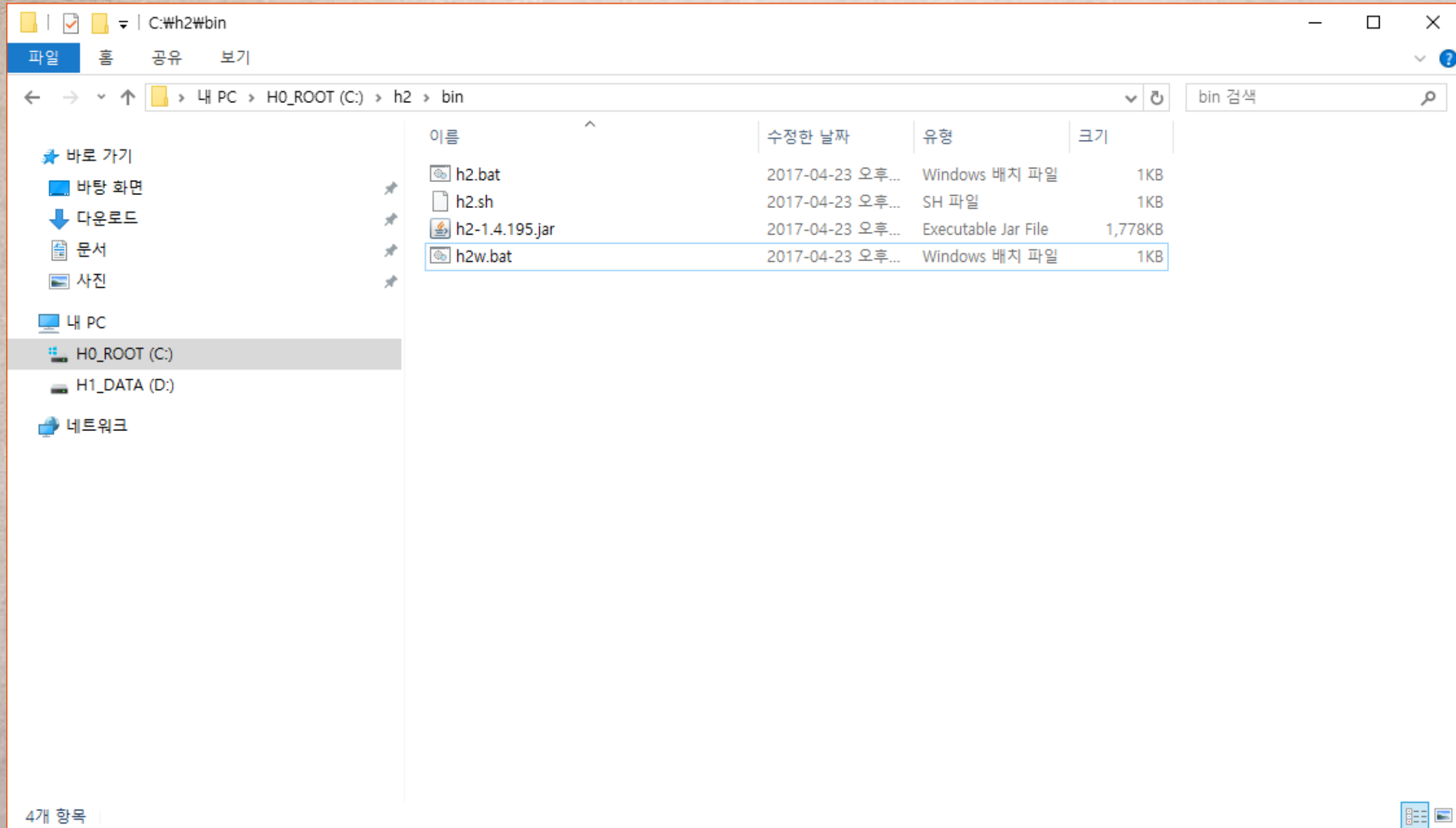
For non-technical issues, use:
dbsupport at h2database.com

Features

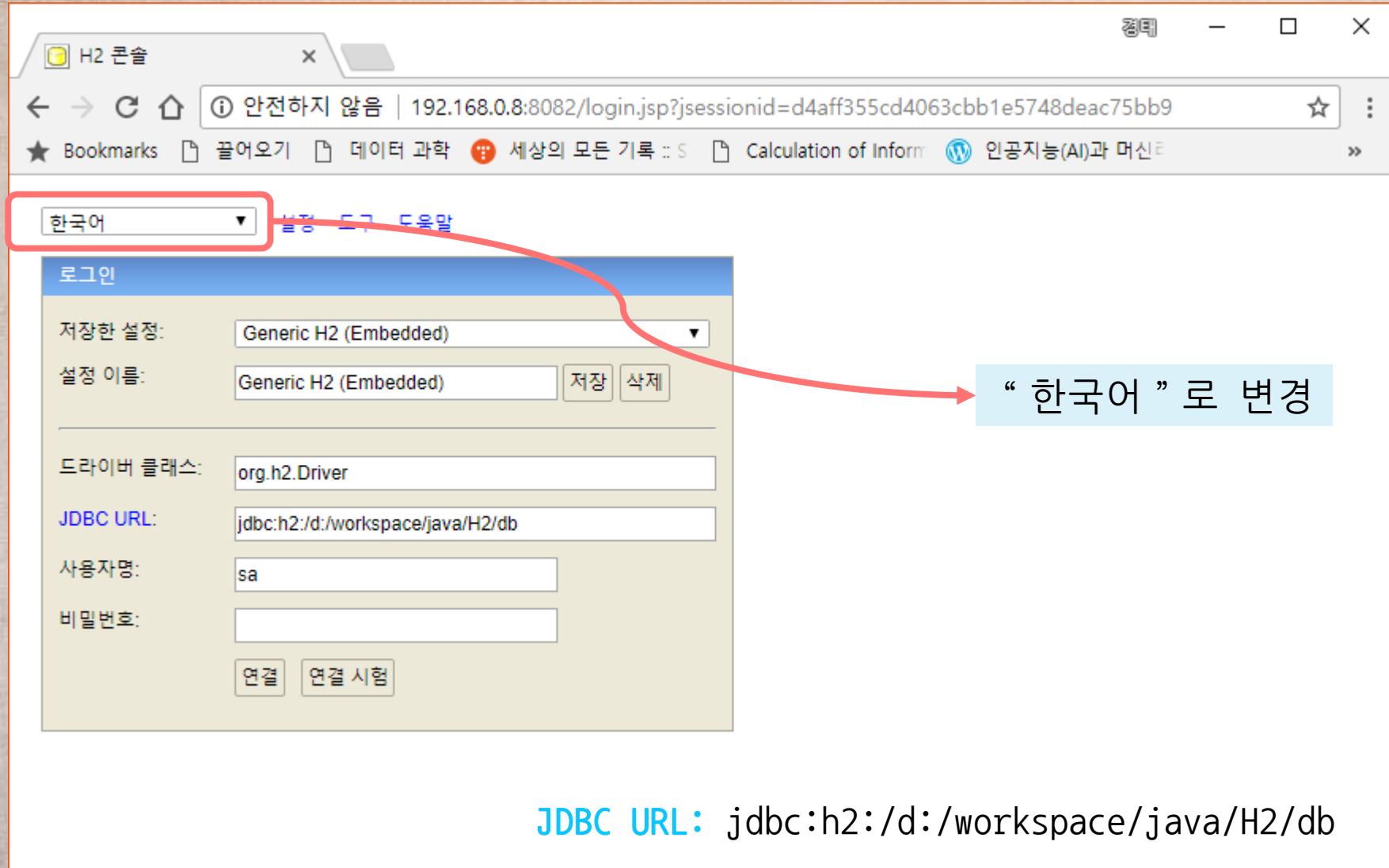
	H2	Derby	HSQldb	MySQL	PostgreSQL
Pure Java	Yes	Yes	Yes	No	No
Memory Mode	Yes	Yes	Yes	No	No
Encrypted Database	Yes	Yes	Yes	No	No
ODBC Driver	Yes	No	No	Yes	Yes
Fulltext Search	Yes	No	No	Yes	Yes
Multi Version Concurrency	Yes	No	Yes	Yes	Yes
Footprint (jar/dll size)	~1 MB	~2 MB	~1 MB	~4 MB	~6 MB

See also the [detailed comparison](#).

H2Database 실행 - h2→bin>h2w.bat 실행



H2 서버 - 구동



The screenshot shows the H2 console login page in a web browser. The browser's address bar displays the URL `192.168.0.8:8082/login.jsp?jsessionId=d4aff355cd4063cbb1e5748deac75bb9`. The page title is "H2 콘솔". A red box highlights the language dropdown menu, which is currently set to "한국어". A red arrow points from this dropdown to a blue callout box containing the text "“ 한국어 ” 로 변경". The login form itself is titled "로그인" and includes the following fields and buttons:

- 저장한 설정:** A dropdown menu showing "Generic H2 (Embedded)".
- 설정 이름:** A text input field containing "Generic H2 (Embedded)", with "저장" (Save) and "삭제" (Delete) buttons next to it.
- 드라이버 클래스:** A text input field containing "org.h2.Driver".
- JDBC URL:** A text input field containing "jdbc:h2:/d:/workspace/java/H2/db".
- 사용자명:** A text input field containing "sa".
- 비밀번호:** An empty text input field.
- Buttons at the bottom: "연결" (Connect) and "연결 시험" (Test Connection).

JDBC URL: jdbc:h2:/d:/workspace/java/H2/db

H2 서버 - 연결

H2 콘솔

192.168.0.8:8082/login.do?sessionId=d01f700eb56c6eb9d577922910474b48

★ Bookmarks | 끌어오기 | 윈도우10 듀얼모니터 | All IT eBooks - Free | MIT 6.00 컴퓨터 공학 | 일행행 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)

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

USER 테이블

```
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  
);
```

BOARD 테이블

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

H2 서버 - 입력 데이터 확인

The screenshot shows the H2 Console web interface in a browser window. The address bar shows the URL `192.168.0.8:8082/login.do?jsessionid=d01f700eb56c6eb9d577922910474b48`. The left sidebar shows the database structure: `jdbc:h2:~/test`, `BOARD`, `USERS`, `INFORMATION_SCHEMA`, `사용자`, and `H2 1.4.195 (2017-04-23)`. The main area displays the SQL queries and their results.

실행 Run Selected 자동 완성 지우기 SQL 문:

```
select * from users;
select * from board;
```

select * from users;

ID	PASSWORD	NAME	ROLE
test	test123	관리자	Admin
user1	user1	홍길동	user

(2 행, 0 ms)

select * from board;

SEQ	TITLE	WRITER	CONTENT	REGDATE	CNT
1	가입인사	관리자	잘 부탁드립니다.....	2017-04-26	0

(1 row, 0 ms)