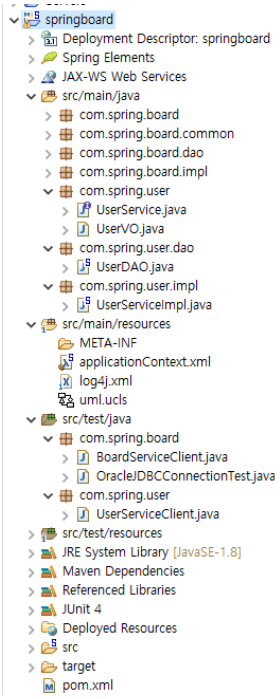
10. 비즈니스 컴포넌트 실습2

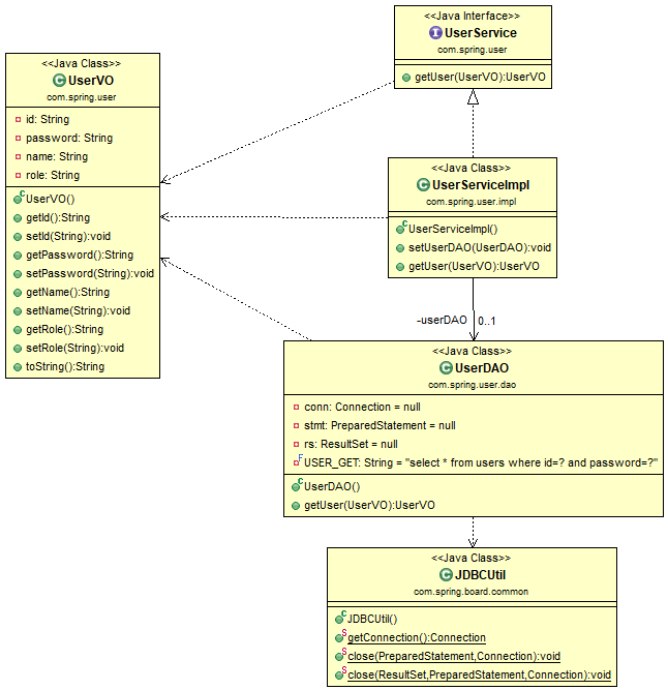
회원 정보를 관리하는 UserService 컴포넌트를 추가한다. 여기서는 어노테이션을 사용하지 않고, Setter 인젝션으로 의존성 주입을 처리하고 나서 어노테이션으로 변경하는 과정으로 실습한다.

10.1 UserService 컴포넌트 구조

파일 구조



UserService 컴포넌트에 대한 클래스 다이어그램



10.2 Value Object 클래스

USERS 테이블의 구조

```
create table users(
	id varchar2(8) primary key,
	password varchar2(8) not null,
	name varchar2(20),
	role varchar2(5)
);
```

USERS 테이블의 칼럼 이름과 메핑되는 멤버변수를 가진 UserVO 클래스를 작성한다.

UserVO.java

```
package com.spring.user;

// VO(Value Object)
public class UserVO {
		private String id;
		private String password;
		private String name;
		private String role;
```

```
	public String getId() {
		return id;
	}
	public void setId(String id) {
		this.id = id;
	}
	public String getPassword() {
		return password;
	}
	public void setPassword(String password) {
		this.password = password;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getRole() {
		return role;
	}
	public void setRole(String role) {
		this.role = role;
	}

	@Override
	public String toString() {
		return "UserVO [id=" + id + ", password=" + password + ", name=" + name + ", role="
+ role + "]";
	}
}
```

10.3 DAO 클래스

JDBCUtil 클래스를 이용하여 UserDAO 클래스의 메소드를 구현한다. 여기서는 UserDAO 클래스에는 회원 정보 하나를 검색하는 getUser() 메소드만 구현한다.

UserDAO.java

```
package com.spring.user.dao;

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

```
import com.spring.board.common.JDBCUtil;
import com.spring.user.UserVO;

//DAO(Data Access Object)
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();
         } finally {
               JDBCUtil.close(rs, stmt, conn);
         }
         return user;
   }
}
```

UserDAO 클래스는 <bean> 등록으로 객체를 생성할 것이므로 어노테이션은 설정하지 않는다.

10.4 Service 인터페이스

UserService.java

```
package com.spring.user;

public interface UserService {
	// CRUD 기능의 메소드 구현
	// 회원 등록
	public UserVO getUser(UserVO vo);
}
```

10.5 Service 구현 클래스

UserService 인터페이스를 구현하는 UserServiceImpl 클래스를 만들면 비즈니스 컴포넌트가 마무리된다.

UserServiceImpl.java

```
package com.spring.user.impl;

import com.spring.user.UserService;
import com.spring.user.UserVO;
import com.spring.user.dao.UserDAO;

public class UserServiceImpl implements UserService {
	private UserDAO userDAO;

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

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

UserServiceImpl 클래스의 비즈니스 메소드를 구현할 때, 멤버변수로 선언된 UserDAO 객체를 이용하여 DB 연동을 처리하면 된다. UserServiceImpl 클래스에는 Setter 인젝션 처리를 위한 Setter 메소드를 추가한다.

10.6 UserService 컴포넌트 테스트

UserService 컴포넌트를 테스트하기 위해서 우선 스프링 설정 파일인 applicationContext.xml에 UserServiceImpl와 UserDAO 클래스를 각각 <bean>을 등록한다. 그리고 UserServiceImpl 클래스에서 UserDAO 객체를 의존성 주입하기 위한 <property> 설정을 추가하면 된다.

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: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-4.3.xsd">

	<context:component-scan base-package="com.spring">
	</context:component-scan>

	<bean id="userService" class="com.spring.user.impl.UserServiceImpl">
		<property name="userDAO" ref="userDAO" />
	</bean>
	<bean id="userDAO" class="com.spring.user.dao.UserDAO" />
</beans>
```

테스트 데이터 추가

```
insert into users values('test', 'test1234', '관리자', 'Admin');
commit;
```

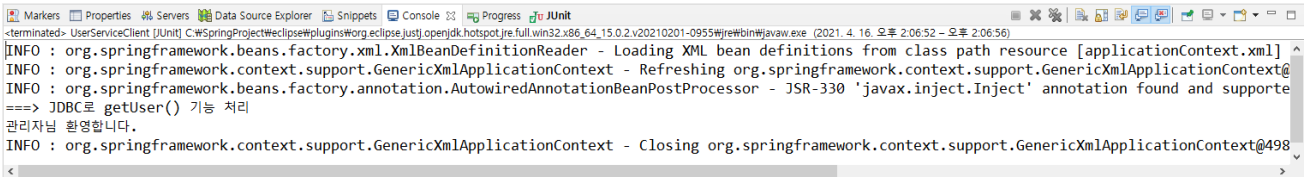
src/test/java

UserServiceClient.java

```
package com.spring.user;

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

public class UserServiceClient {

	@Test
	public void userTest() {
		// 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("test1234");

		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();
	}
}
```

Run As -> JUnit Test



10.7 어노테이션 적용

Setter 인젝션 설정으로 테스트한 UserService 컴포넌트를 어노테이션 설정으로 변경한다. 우선 스프링 설정으로 Setter 인젝션 관련 설정을 모두 주석 처리한다.

applicationContext.xml

```
<!--
	<bean id="userService" class="com.spring.user.impl.UserServiceImpl">
		<property name="userDAO" ref="userDAO" />
	</bean>
	<bean id="userDAO" class="com.spring.user.dao.UserDAO" />
 -->
```

그리고 UserServiceImpl과 UserDAO 클래스에 각각 관련된 어노테이션을 추가한다.

UserServiceImpl.java

```
package com.spring.user.impl;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import com.spring.user.UserService;
import com.spring.user.UserVO;
```

```
import com.spring.user.dao.UserDAO;

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

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

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

UserDAO.java

```
package com.spring.user.impl;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import org.springframework.stereotype.Repository;
import com.spring.board.common.JDBCUtil;
import com.spring.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;

    생략
}
```

수정 후 UserServiceClient을 실행해도 결과는 같다.