KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

JDBC 프로그래밍 (심화1)

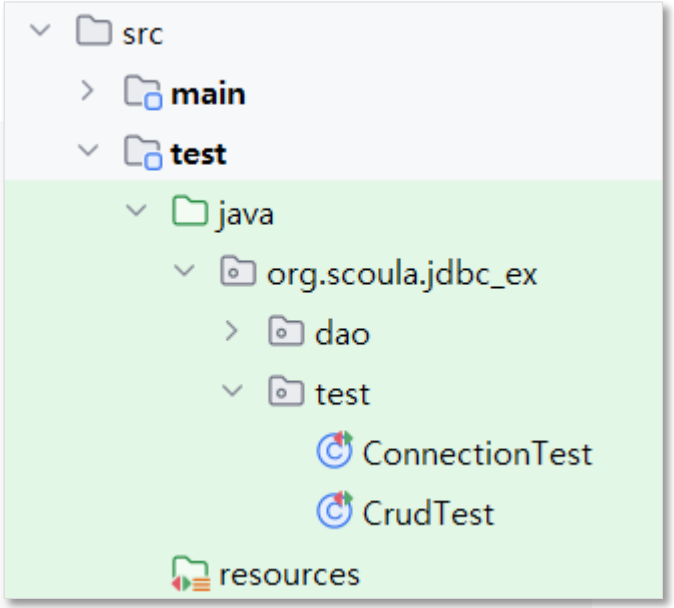
[KB] IT's Your Life

KB 국민은행

- **테스트 폴더에 CrudTest 클래스를 생성하세요.**
  - 패키지: org.scoula.jdbc_ex.test
  - 테스트 순서는 어노테이션으로 지정함을 설정
  - JDBCUtil을 이용하여 Connection 객체를 멤버 변수로 정의
  - 하나의 단위 테스트가 끝날때 마다 Connection을 닫는 tearDown() 메서드를 정의

CrudTest.java

```
package org.scoula.jdbc_ex.test;

import org.junit.jupiter.api.*;
import org.scoula.jdbc_ex.common.JDBCUtil;

import java.sql.*;

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
public class CrudTest {
    Connection conn = JDBCUtil.getConnection();

    @AfterAll
    static void tearDown() {
        JDBCUtil.close();
    }
}
```



- **CrudTest 클래스에 insertUser() 테스트 메서드를 작성하세요.**
  - 테스트 순서 1번
  - PreparedStatement를 이용하여 users 테이블에 데이터를 추가
  - insert 작업의 성공 여부를 단정문으로 확인

CrudTest.java

```
@Test
@DisplayName("새로운 user를 등록한다.")
@Order(1)
public void insertUser()  throws SQLException {
    String sql = "insert into users(id, password, name, role) values(?, ?, ?, ?)";
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setString(1, "scoula");
        pstmt.setString(2, "scoula3");
        pstmt.setString(3, "스콜라");
        pstmt.setString(4, "USER");

        int count = pstmt.executeUpdate();
        Assertions.assertEquals(1, count);
    }
}
```

- **CrudTest 클래스에 selectUser() 테스트 메서드를 작성하세요.**
  - 테스트 순서 2번 지정
  - Statement를 이용하여 users 테이블의 모든 데이터를 추출
  - ResultSet을 이용하여 추출한 모든 행의 name 컬럼 출력

CrudTest.java

```
@Test
@DisplayName("user 목록을 추출한다.")
@Order(2)
public void selectUser() throws SQLException {
    String sql ="select * from users";
    try(Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql);
        ) {
        while(rs.next()) {
            System.out.println(rs.getString("name"));
        }
    }
}
```

- **CrudTest 클래스에 selectUserById() 테스트 메서드를 작성하세요.**
  - 테스트 순서 3번 지정
  - PreparedStatement를 이용하여 users 테이블에서 id를 이용하여 한 개의 데이터를 추출
  - ResultSet을 이용하여 추출한 행의 name 컬럼 출력

CrudTest.java

```
@Test
@DisplayName("특정 user 검색한다.")
@Order(3)
public void selectUserById() throws SQLException {
    String userid = "scoula";
    String sql ="select * from users where id = ?";
    try(PreparedStatement stmt = conn.prepareStatement(sql)){
        stmt.setString(1, userid);
        try(ResultSet rs = stmt.executeQuery()) {
            if(rs.next()) {
                System.out.println(rs.getString("name"));
            } else {
                throw new SQLException("scoula not found");
            }
        }
    }
}
```

- **CrudTest 클래스에 updateUser() 테스트 메서드를 작성하세요.**
  - 테스트 순서 4번 지정
  - PreparedStatement를 이용하여 users 테이블에서 지정한 id의 name을 수정
  - 단정문을 이용하여 update문 실행 성공 여부를 판정

CrudTest.java

```
@Test
@DisplayName("특정 user 수정한다.")
@Order(4)
public void updateUser() throws SQLException {
    String userid = "scoula";
    String sql ="update users set name= ?  where id = ?";
    try(PreparedStatement stmt = conn.prepareStatement(sql)){
        stmt.setString(1, "스콜라 수정");
        stmt.setString(2, userid);
        int count = stmt.executeUpdate();
        Assertions.assertEquals(1, count);
    }
}
```

- **CrudTest 클래스에 deleteUser() 테스트 메서드를 작성하세요.**
  - 테스트 순서 5번 지정
  - PreparedStatement를 이용하여 users 테이블에서 지정한 id의 행 삭제
    단정문을 이용하여 delete문 실행 성공 여부를 판정

CrudTest.java

```
    @Test
    @DisplayName("지정한 사용자를 삭제한다.")
    @Order(5)
    public void deleteUser() throws SQLException {
        String userid = "scoula";
        String sql ="delete from users where id = ?";
        try(PreparedStatement stmt = conn.prepareStatement(sql)){
            stmt.setString(1, userid);
            int count = stmt.executeUpdate();
            Assertions.assertEquals(1, count);
        }
    }
}
```

KB금융그룹 | 국민의 평생 금융파트너

2025년 상반기 K-디지털 트레이닝

JDBC 프로그래밍 (심화2)

[KB] IT's Your Life

KB 국민은행

- **Users 테이블을 위한 VO 클래스를 정의하세요.**
  - 패키지: org.scoula.jdbc_ex.domain
  - 클래스명: UserVO
  - 롬복을 이용한 기본 정의

UserVO.java

```
package org.scoula.jdbc_ex.domain;

import lombok.AllArgsConstructor;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@NoArgsConstructor
@AllArgsConstructor
public class UserVO {
    private String id;
    private String password;
    private String name;
    private String role;
}
```

- **Users 테이블에 대한 기본 CRUD 연산을 수행하기 위한 인터페이스를 정의하세요.**
  - 패키지명: org.scoula.jdbc_ex.dao
  - 인터페이스명: UserDao
  - 메서드 목록
    - int create(UserVO user) throws SQLException;
    - List<UserVO> getList() throws SQLException;
    - Optional<UserVO> get(String id) throws SQLException;
    - int update(UserVO user) throws SQLException;
    - int delete(String id) throws SQLException;

UserDao.java

```
package org.scoula.jdbc_ex.dao;

import org.scoula.jdbc_ex.domain.UserVO;

import java.sql.SQLException;
import java.util.List;
import java.util.Optional;

public interface UserDao {
    // 회원 등록
    int create(UserVO user) throws SQLException;

    // 회원 목록 조회
    List<UserVO> getList() throws SQLException;

    // 회원 정보 조회
    Optional<UserVO> get(String id) throws SQLException;

    // 회원 수정
    int update(UserVO user) throws SQLException;

    // 회원 삭제
    int delete(String id) throws SQLException;
}
```

- UserDao의 구현 클래스 UserDaoImpl 클래스를 정의하세요.

UserDao.java

```
package org.scoula.jdbc_ex.dao;

import org.scoula.jdbc_ex.domain.UserVO;

import java.sql.SQLException;
import java.util.List;
import java.util.Optional;

public interface UserDao {
    // 회원 등록
    int create(UserVO user) throws SQLException;

    // 회원 목록 조회
    List<UserVO> getList() throws SQLException;

    // 회원 정보 조회
    Optional<UserVO> get(String id) throws SQLException;

    // 회원 수정
    int update(UserVO user) throws SQLException;

    // 회원 삭제
    int delete(String id) throws SQLException;
}
```

UserDaoImpl.java

```
package org.scoula.jdbc_ex.dao;

import org.scoula.jdbc_ex.common.JDBCUtil;
import org.scoula.jdbc_ex.domain.UserVO;

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

public class UserDaoImpl implements UserDao {
    Connection conn = JDBCUtil.getConnection();

    // USERS 테이블 관련 SQL 명령어
    private String USER_LIST   = "select * from users";
    private String USER_GET    = "select * from users where id = ?";
    private String USER_INSERT = "insert into users values(?, ?, ?, ?)";
    private String USER_UPDATE = "update users set name = ?, role = ? where id = ?";
    private String USER_DELETE = "delete from users where id = ?";
```

UserDaoImpl.java

```
private String USER_INSERT = "insert into users values(?, ?, ?, ?)";
```

```
// 회원 등록
@Override
public int create(UserVO user) throws SQLException {
    try (PreparedStatement stmt = conn.prepareStatement(USER_INSERT)) {
        stmt.setString(1, user.getId());
        stmt.setString(2, user.getPassword());
        stmt.setString(3, user.getName());
        stmt.setString(4, user.getRole());
        return stmt.executeUpdate();
    }
}
```

UserDaoImpl.java

```
private UserVO map(ResultSet rs) throws SQLException {
    UserVO user = new UserVO();
    user.setId(rs.getString("ID"));
    user.setPassword(rs.getString("PASSWORD"));
    user.setName(rs.getString("NAME"));
    user.setRole(rs.getString("ROLE"));
    return user;
}

// 회원 목록 조회
@Override
public List<UserVO> getList() throws SQLException{
    List<UserVO> userList = new ArrayList<UserVO>();
    Connection conn = JDBCUtil.getConnection();
    try (PreparedStatement stmt = conn.prepareStatement(USER_LIST);
         ResultSet rs = stmt.executeQuery()) {
        while(rs.next()) {
            UserVO user = map(rs);
            userList.add(user);
        }
    }
    return userList;
}
```

```
private String USER_LIST   = "select * from users";
```

UserDaoImpl.java

```
private String USER_GET     = "select * from users where id = ?";
```

```
// 회원 정보 조회
@Override
public Optional<UserVO> get(String id) throws SQLException{
    try (PreparedStatement stmt = conn.prepareStatement(USER_GET)) {
        stmt.setString(1, id);
        try(ResultSet rs = stmt.executeQuery()) {
            if(rs.next()) {
                return Optional.of(map(rs));
            }
        }
    }
    return Optional.empty();
}
```

UserDaoImpl.java

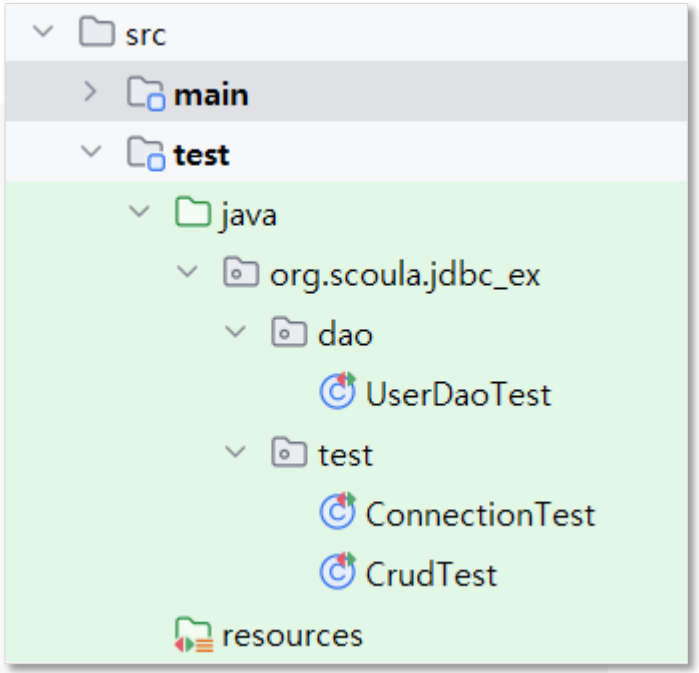
```
    // 회원 수정
    @Override
    public int update(UserVO user) throws SQLException{
        Connection conn = JDBCUtil.getConnection();
        try ( PreparedStatement stmt = conn.prepareStatement(USER_UPDATE)) {
            stmt.setString(1, user.getName());
            stmt.setString(2, user.getRole());
            stmt.setString(3, user.getId());
            return stmt.executeUpdate();
        }
    }

    // USERS 테이블 관련 CRUD 메소드
    // 회원 삭제
    @Override
    public int delete(String id) throws SQLException{
        try(PreparedStatement stmt = conn.prepareStatement(USER_DELETE)) {
            stmt.setString(1, id);
            return stmt.executeUpdate();
        }
    }
}
```

```
private String USER_UPDATE = "update users set name = ?, role = ? where id = ?";
```

```
private String USER_DELETE = "delete from users where id = ?";
```

- UserDaoTest 클래스를 작성하여 UserDaoImpl 클래스에 대해 단위 테스트를 진행하세요.

UserDaoTest.java




```
package org.scoula.jdbc_ex.dao;

import org.junit.jupiter.api.*;
import org.scoula.jdbc_ex.common.JDBCUtil;
import org.scoula.jdbc_ex.domain.UserVO;

import java.sql.SQLException;
import java.util.List;
import java.util.NoSuchElementException;

@TestMethodOrder(MethodOrderer.OrderAnnotation.class)
class UserDaoTest {
    UserDao dao = new UserDaoImpl();

    @AfterAll
    static void tearDown() {
        JDBCUtil.close();
    }
```

UserDaoTest.java

```
@Test
@DisplayName("user를 등록합니다.")
@Order(1)
void create() throws SQLException {
    UserVO user = new UserVO("ssamz3", "ssamz123", "쌤즈", "ADMIN");
    int count = dao.create(user);
    Assertions.assertEquals(1, count);
}

@Test
@DisplayName("UserDao User 목록을 추출합니다.")
@Order(2)
void getList() throws SQLException {
    List<UserVO> list = dao.getList();
    for(UserVO vo: list) {
        System.out.println(vo);
    }
}
```

UserDaoTest.java

```
    @Test
    @DisplayName("특정 user 1건을 추출합니다.")
    @Order(3)
    void get() throws SQLException {
        UserVO user = dao.get("ssamz3").orElseThrow(NoSuchElementException::new);
        Assertions.assertNotNull(user);
    }

    @Test
    @DisplayName("user의 정보를 수정합니다.")
    @Order(4)
    void update() throws SQLException {
        UserVO user = dao.get("ssamz3").orElseThrow(NoSuchElementException::new);
        user.setName("쌤즈3");
        int count = dao.update(user);
        Assertions.assertEquals(1, count);
    }

    @Test
    @DisplayName("user를 삭제합니다.")
    @Order(5)
    void delete() throws SQLException {
        int count = dao.delete("ssamz3");
        Assertions.assertEquals(1, count);
    }
}
```