KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

JDBC 프로그래밍

[KB] IT's Your Life

KB 국민은행

- MySQL에 jdbc_ex 데이터베이스를 생성하세요.
- jdbc_ex라는 이름의 사용자를 생성하세요(단, 비밀번호는 jdbc_ex)
- jdbc_ex 데이터베이스에 대한 모든 권한을 사용자 jdbc_ex에게 부여하세요.

데이터베이스 준비

```
CREATE DATABASE jdbc_ex;
```

사용자 준비

```
CREATE USER 'jdbc_ex'@'%' IDENTIFIED BY 'jdbc_ex';
GRANT ALL PRIVILEGES ON jdbc_ex.* TO 'jdbc_ex'@'%';
FLUSH PRIVILEGES;
```

- jdbc_ex 데이터베이스에서 다음과 같은 필드 속성을 가지는 users 테이블을 생성하세요.

컬럼명	타입	필수 여부	비고
ID	가변길이(12)	필수	기본 키
PASSWORD	가변길이(12)	필수	
NAME	가변길이(30)	필수	
ROLE	가변길이(6)	필수	

- 다음 데이터를 users 테이블에 추가하세요.

ID	PASSWORD	NAME	ROLE
guest	guest123	방문자	USER
admin	admin123	ADMIN	ADMIN
member	member123	일반회원	USER

- 위에서 추가한 users 테이블 내용을 확인하세요.

데이터베이스 준비

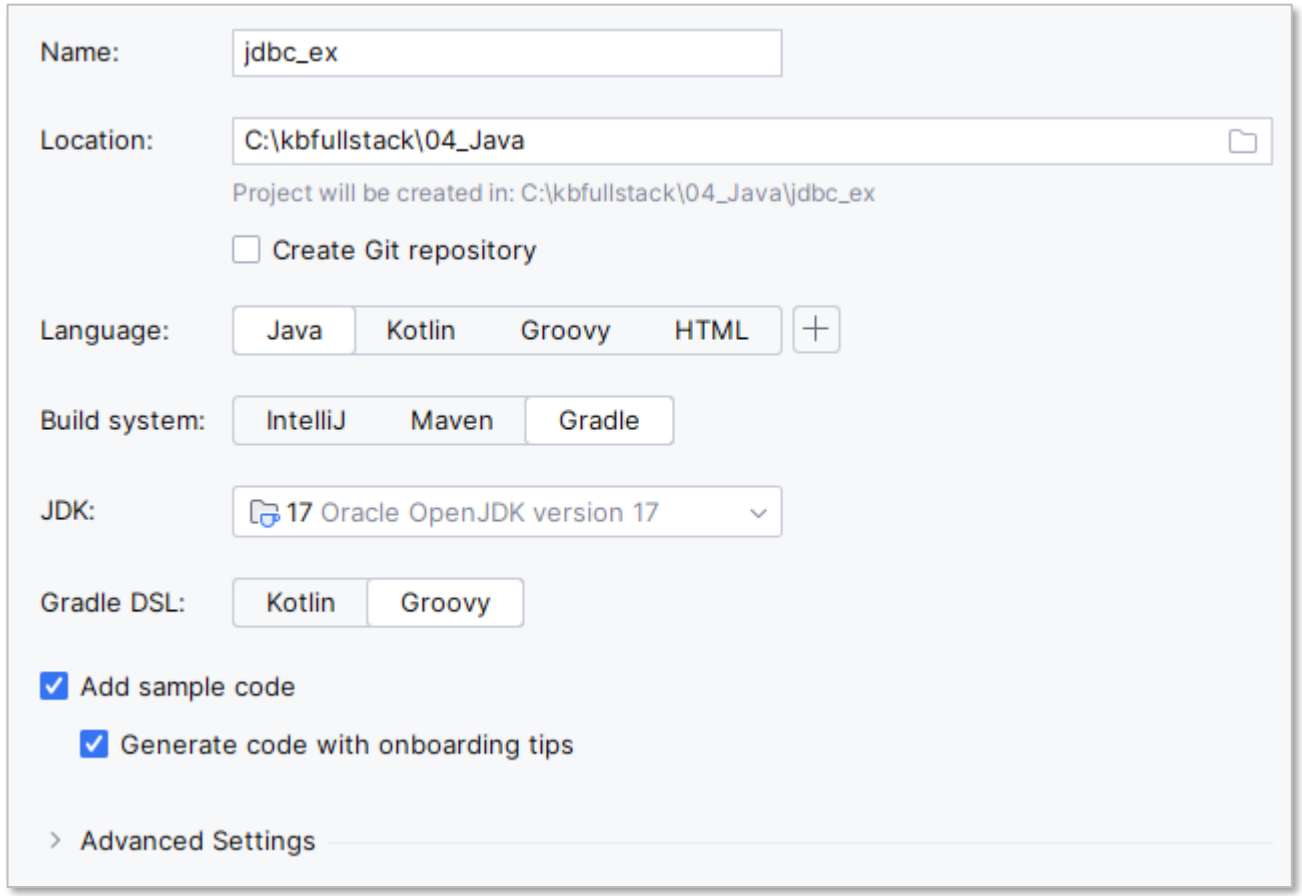
- jdbc_ex 계정 작업

```
CREATE TABLE USERS (
  ID VARCHAR(12) NOT NULL PRIMARY KEY,
  PASSWORD VARCHAR(12) NOT NULL,
  NAME VARCHAR(30) NOT NULL,
  ROLE VARCHAR(6) NOT NULL
);


INSERT INTO USERS(ID, PASSWORD, NAME, ROLE)
VALUES('guest', 'guest123', '방문자', 'USER');

INSERT INTO USERS(ID, PASSWORD, NAME, ROLE)
VALUES('admin', 'admin123', '관리자', 'ADMIN');

INSERT INTO USERS(ID, PASSWORD, NAME, ROLE)
VALUES('member', 'member123', '일반회원', 'USER');

SELECT * FROM USERS;
```

- **jdbc_ex 프로젝트를 생성하세요.**
  - gradle 시스템을 이용함
  - MySQL JDBC 드라이버, Lombok 라이브러리를 추가함

프로젝트 생성

- Name: jdbc_ex
- Build System : gradle



build.gradle

```
…
dependencies {
    implementation 'com.mysql:mysql-connector-j:8.3.0'
    compileOnly 'org.projectlombok:lombok:1.18.30'

    testImplementation platform('org.junit:junit-bom:5.9.1')
    testImplementation 'org.junit.jupiter:junit-jupiter'
}
…
```

- 수정 후 Sync 실행

- 프로젝트 설정
  - Annotation Processing 활성화

- MySQL 데이터베이스에 대한 연결 관리 클래스를 작성하세요.

```
package org.example.common;

public class JDBCUtil {

}
```

- Try-with-resource 패턴으로 위에서 만든 클래스로 데이터베이스 접속 테스트 클래스를 작성하세요.

```
package org.example;

public class ConnectionTest2 {
  public static void main(String[] args) {

  }

}
```

jdbc_ex.common/JDBCUtil.java

```
package org.example.common;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBCUtil {
  static Connection conn = null;
  static {
    try {
      Class.forName("com.mysql.cj.jdbc.Driver");
      String url = "jdbc:mysql://127.0.0.1:3306/jdbc_ex";
      String id = "jdbc_ex";
      String password = "jdbc_ex";
      conn = DriverManager.getConnection(url, id, password);
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
```

jdbc_ex.comon/JDBCUtil.java

```
  public static Connection getConnection() {
    return conn;
  }

  public static void close() {
    try {
      if (conn != null) {
        conn.close();
        conn = null;
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }

}
```

ConnectionTest2.java

```
package org.example;

import java.sql.Connection;
import java.sql.SQLException;

public class ConnectionTest2 {
  public static void main(String[] args) {

    try(Connection conn = JDBCUtil.getConnection()) {
      System.out.println("DB 연결 성공");
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }

}
```

```
DB 연결 성공
```

users 테이블의 내용을 확인하는 테스트 클래스를 작성하세요.

- Statement 클래스를 이용해서 쿼리 실행함
- name 필드만 출력함

```
package org.example;

public class SelectUserTest {
  public static void main(String[] args) {

  }
}
```

```
관리자
방문자
일반회원
```

SelectUserTest.java

```
…

public class SelectUserTest {

  public static void main(String[] args) {
    String sql = "select * from users";

    try (Connection conn = JDBCUtil.getConnect();
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(sql) ) {

      while (rs.next()) {
        System.out.println(rs.getString("name"));
      }
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

관리자
방문자
일반회원

- PreparedStatement를 이용해서 users 테이블에 행을 하나 추가하세요.

```
public class InsertUserTest {
  public static void main(String[] args) {

  }
}
```

- 앞에서 추가한 데이터를 PreparedStatement를 이용해서 삭제하는 코드를 작성하세요.

```
public class DeleteUserTest {
  public static void main(String[] args) {

  }
}
```

InsertUserTest.java

```
public class InsertUserTest {
  public static void main(String[] args) {
    String sql = "insert into users(id, password, name, role) values(?, ?, ?, ?)";

    try (Connection conn = JDBCUtil.getConnection();
      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();

      System.out.println(count + "건 데이터 처리 성공!");
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

```
1건 데이터 처리 성공!
```

DeleteUserTest.java

```
public class DeleteUserTest {
  public static void main(String[] args) {
    String sql = "delete from users where id = ?";

    try (Connection conn = JDBCUtil.getConnection();
      PreparedStatement pstmt = conn.prepareStatement(sql)) {
      pstmt.setString(1, "scoula");

      int count = pstmt.executeUpdate();

      System.out.println(count + "건 데이터 처리 성공!");
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

```
1건 데이터 처리 성공!
```

- users 테이블 데이터를 처리하기 위한 VO 클래스를 작성하세요.

```
package org.example.domain;

public class UserVO {

}
```

UserVO.java

```
package org.example.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 CRUD를 수행할 수 있는 UserDAO 클래스를 작성하세요.

```
package org.example.dao;

public class UserDAO {

}
```

UserDAO.java

```
package org.example.dao;

import org.example.common.JDBCUtil;
import org.example.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;

public class UserDAO {
  // 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 users where id = ?";
```

UserDAO.java

```
// 회원 등록
public void insertUser(UserVO user) {
  Connection conn = JDBCUtil.getConnection();
  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());
    stmt.executeUpdate();
  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

UserDAO.java

```
// 회원 목록 조회
public List<UserVO> getUserList() {
  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 = new UserVO();
      user.setId(rs.getString("ID"));
      user.setPassword(rs.getString("PASSWORD"));
      user.setName(rs.getString("NAME"));
      user.setRole(rs.getString("ROLE"));
      userList.add(user);
    }
  } catch (SQLException e) {
    e.printStackTrace();
  }

  return userList;
}
```

UserDAO.java

```
// 회원 정보 조회
public UserVO getUser(String id) {
  Connection conn = JDBCUtil.getConnection();
  try (PreparedStatement stmt = conn.prepareStatement(USER_GET)) {
    stmt.setString(1, id);

    try(ResultSet rs = stmt.executeQuery()) {
      if(rs.next()) {
        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;
      }
    }
  } catch (SQLException e) {
    e.printStackTrace();
  }

  return null;
}
```

UserDAO.java

```
// 회원 수정
public void updateUser(UserVO user) {
  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());
    stmt.executeUpdate();
  } catch (SQLException e) {
    e.printStackTrace();
  }
}
```

UserDAO.java

```
  // USERS 테이블 관련 CRUD 메소드
  // 회원 삭제
  public void deleteUser(String id) {
    Connection conn = JDBCUtil.getConnection();
    try(PreparedStatement stmt = conn.prepareStatement(USER_DELETE)) {
      stmt.setString(1, id);
      stmt.executeUpdate();
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

- 앞에서 만든 UserDAO 기능을 확인할 수 있는 코드를 작성하세요.

```
package org.example;

public class UserDaoTest {
  public static void main(String[] args) {

  }
}
```

UserDaoTest.java

```
package org.example;

import org.example.common.JDBCUtil;
import org.example.dao.UserDAO;
import org.example.domain.UserVO;

import java.util.List;

public class UserDaoTest {
  public static void main(String[] args) {
    UserDAO dao = new UserDAO();
    UserVO user = new UserVO("ssamz3", "ssamz123", "쌤즈", "ADMIN");

    System.out.println("InsertUser =============");
    dao.insertUser(user);

    System.out.println("getUserList ============");
    List<UserVO> list = dao.getUserList();
    for(UserVO vo: list) {
      System.out.println(vo);
    }
```

UserDaoTest.java

```
    System.out.println("updateUser ============");
    user.setName("쌤즈3");
    dao.updateUser(user);

    System.out.println("getUserDetail =========");
    UserVO user2 = dao.getUser("ssamz3");
    System.out.println(user2);


    System.out.println("deleteUser ============");
    dao.deleteUser("ssamz3");

    JDBCUtil.close();
  }
}
```