KB금융그룹 | 국민의 평생 금융파트너

2024년 상반기 K-디지털 트레이닝

JDBC 프로그래밍

[KB] IT's Your Life

KB 국민은행

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;
```

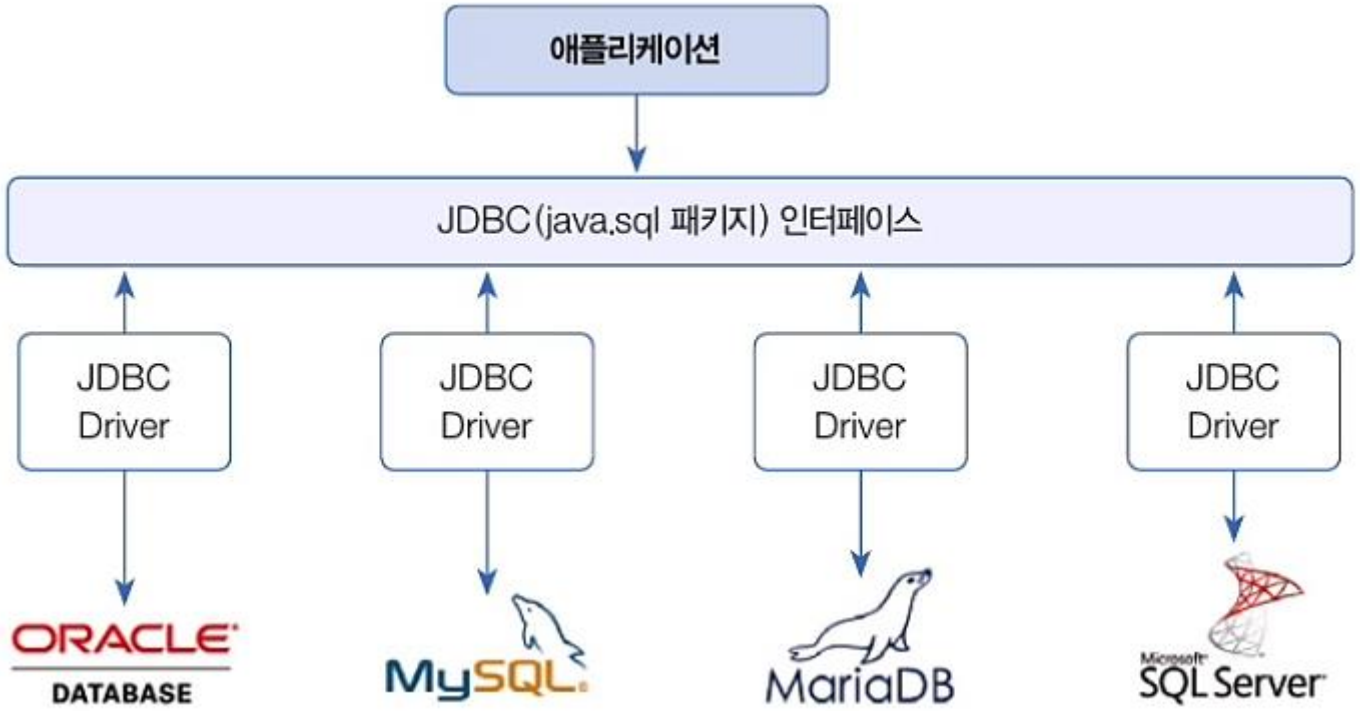
데이터베이스 준비

- 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;
```

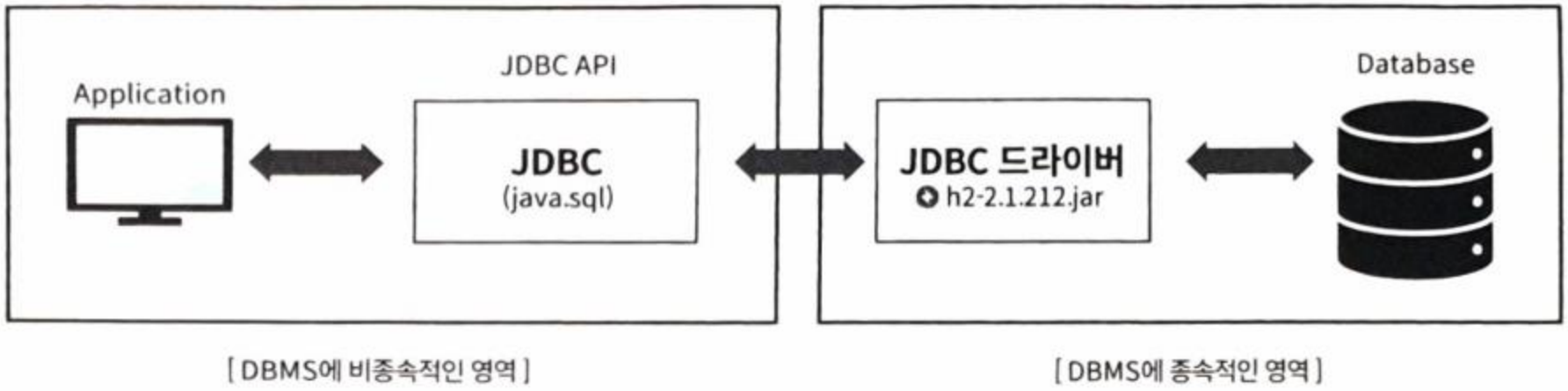
JDBC Java Database Connectivity

- 데이터베이스와 연결해서 입출력을 지원
- 데이터베이스 관리시스템(DBMS)의 종류와 상관없이 동일하게 사용할 수 있는 클래스와 인터페이스로 구성



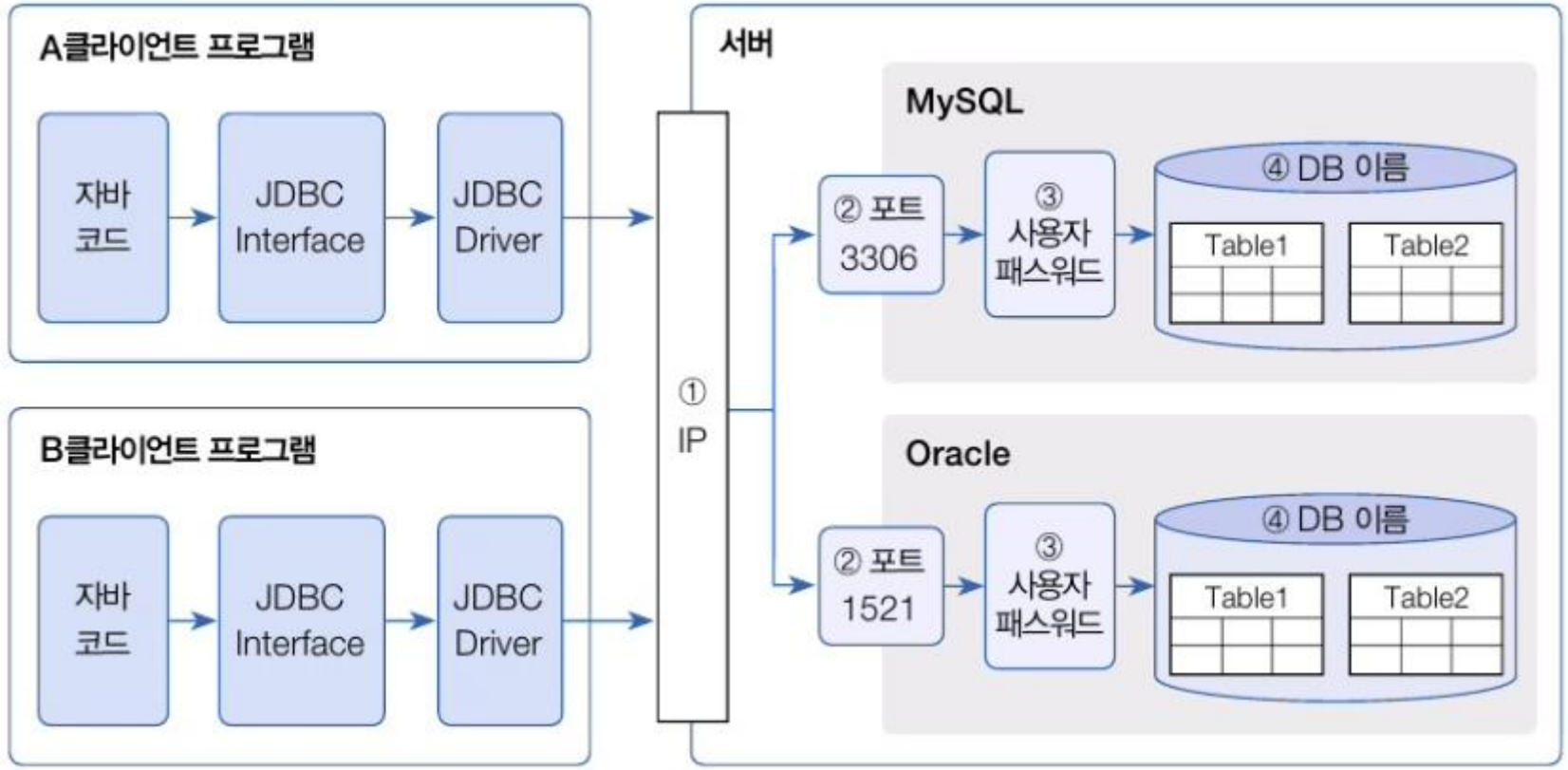
JDBC

JDBC 개념


Application
JDBC API
JDBC
(java.sql)
JDBC 드라이버
h2-2.1.212.jar
Database
[DBMS에 비종속적인 영역]
[DBMS에 종속적인 영역]

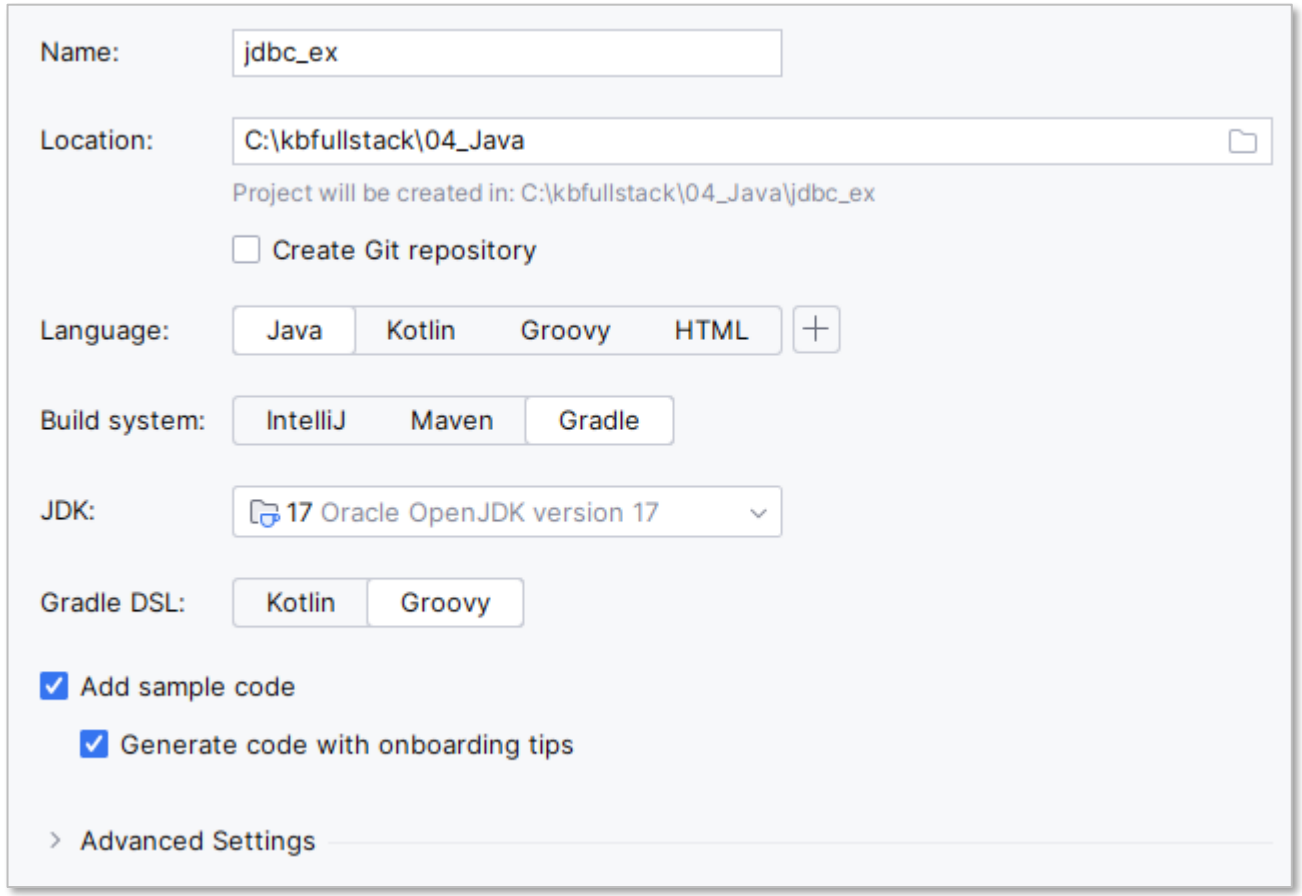
JDBC

JDBC 개념


A클라이언트 프로그램
자바 코드
JDBC Interface
JDBC Driver
B클라이언트 프로그램
자바 코드
JDBC Interface
JDBC Driver
① IP
서버
MySQL
② 포트 3306
③ 사용자 패스워드
④ DB 이름
Table1
Table2
Oracle
② 포트 1521
③ 사용자 패스워드
④ DB 이름
Table1
Table2

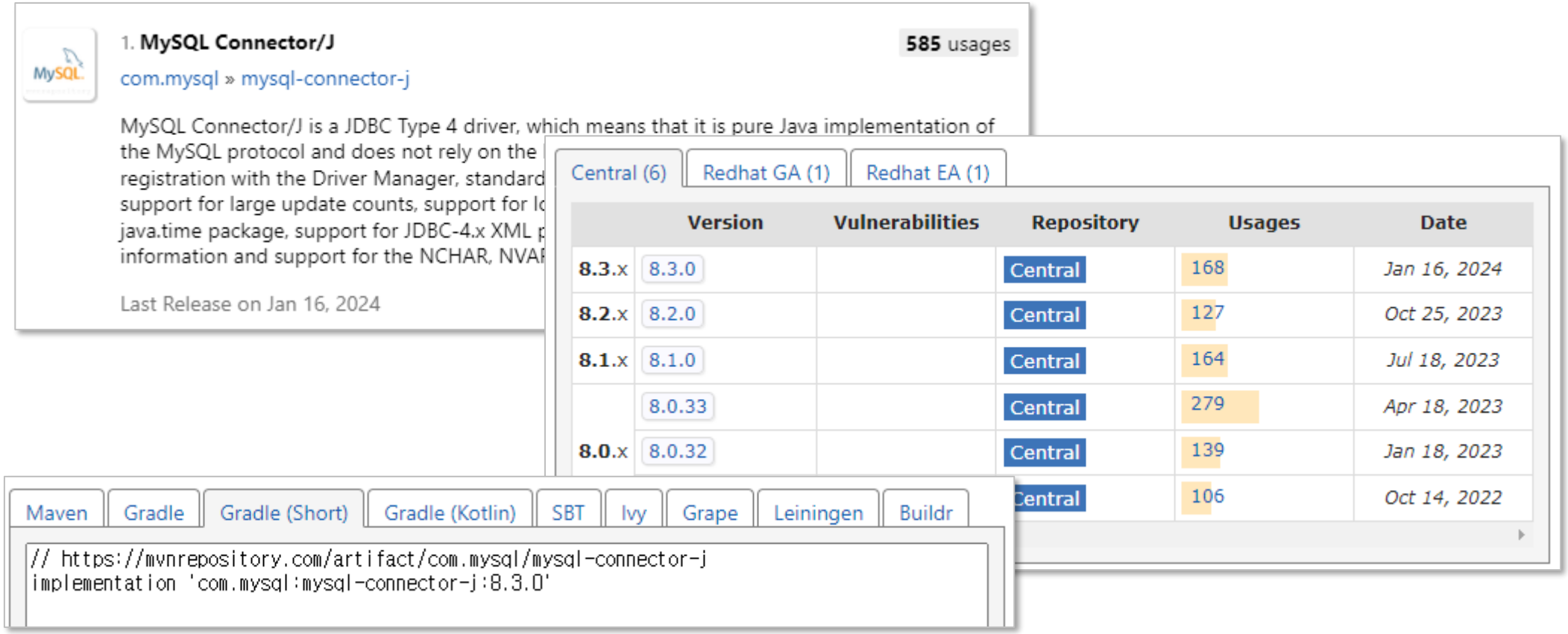
프로젝트 생성

- Name: jdbc_ex
- Build System : gradle



MySQL Connector

- https://mvnrepository.com/
- MySQL 검색



- Lombok도 추가

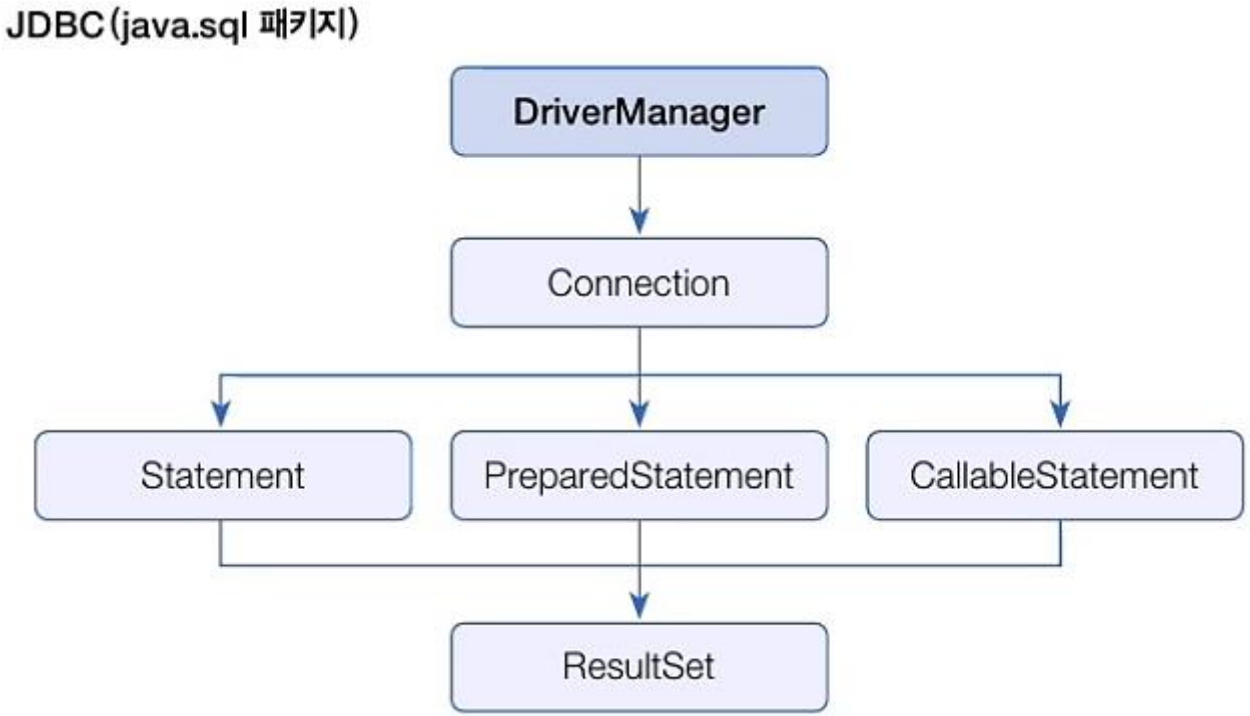
build.gradle

```
…
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

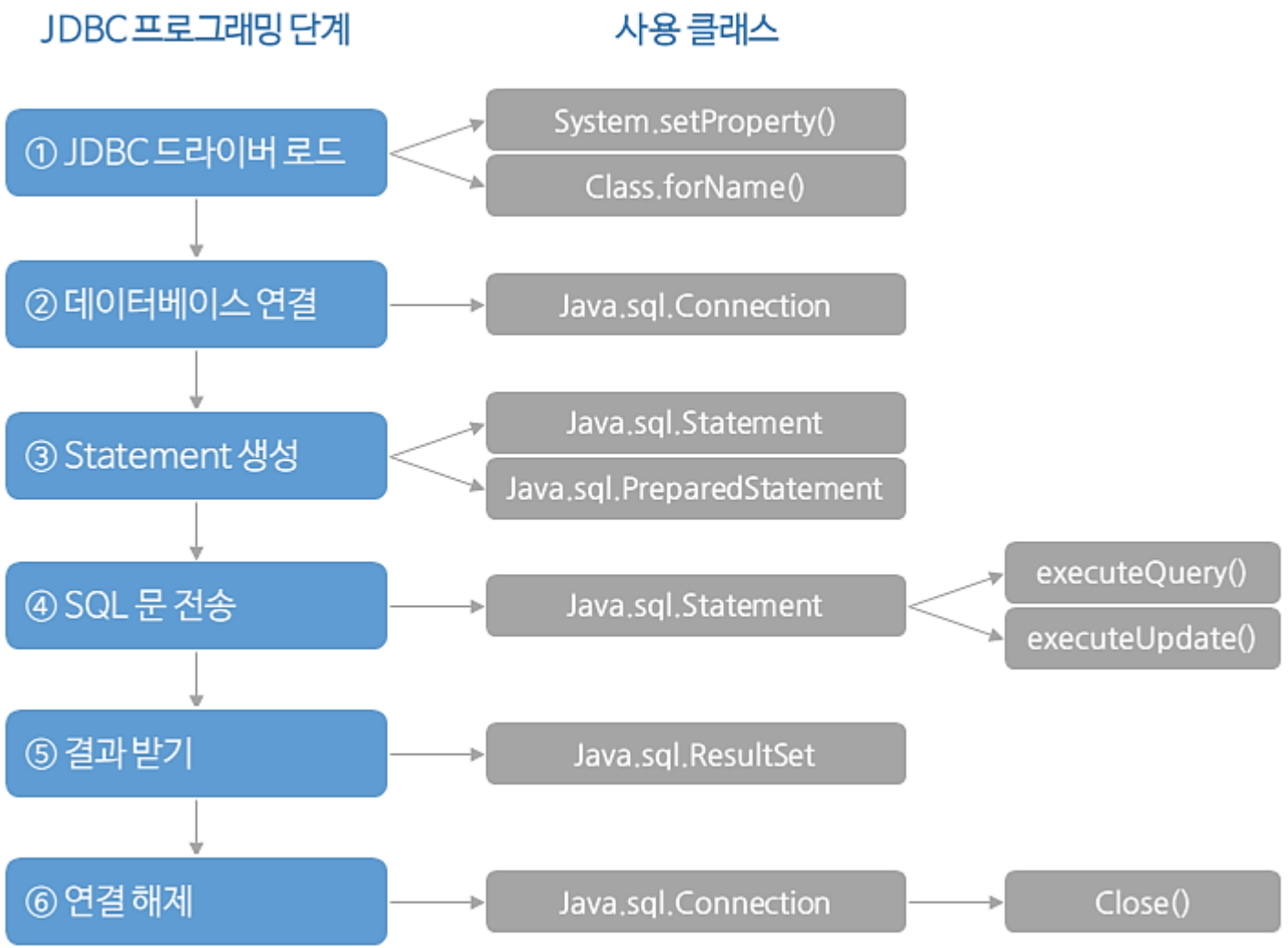
- 프로젝트 설정
  - Annotation Processing 활성화

JDBC의 핵심 인터페이스/클래스



JDBC

- JDBC 개발 절차



DB 연결

드라이버 확인

Class.forName("com.mysql.cj.jdbc.Driver");

→ 없으면 ClassNotFoundException 발생

Connection 객체

- 데이터베이스에 연결 세션을 만듦
- Connection conn = DriverManager.getConnection(
  "연결 문자열", "사용자", "비밀번호")

- 연결 문자열
  "jdbc:mysql://[host]:[포트]/[db이름]
  → jdbc:mysql://127.0.0.1:3306/jdbc_ex

- String url = "jdbc:mysql://127.0.0.1:3306/jdbc_ex";
- Connection conn = DriverManager.getConnection(url, "jdbc_ex", "jdbc_ex");

ConnectionTest.java

```
package org.example;

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

public class ConnectionTest {
  public static void main(String[] args) {
    Connection conn = null;
    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);

      System.out.println("DB 연결 성공");
    } catch (Exception e) {
      e.printStackTrace();
    }
```

ConnectionTest.java

```
    finally {
      try {
        if (conn != null) {
          conn.close();
          conn = null;
        }
          } catch (SQLException e) {
        e.printStackTrace();
      }
    }
  }
}
```

```
DB 연결 성공
```

모듈화해야 할 코드

- 데이터베이스 연결 및 닫기 작업은 항상 필요함
- 형태는 변하지 않음

→ common.JDBCUtil

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();
    }
  }

}
```

ConnectionTest.java

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

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

}
```

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
```

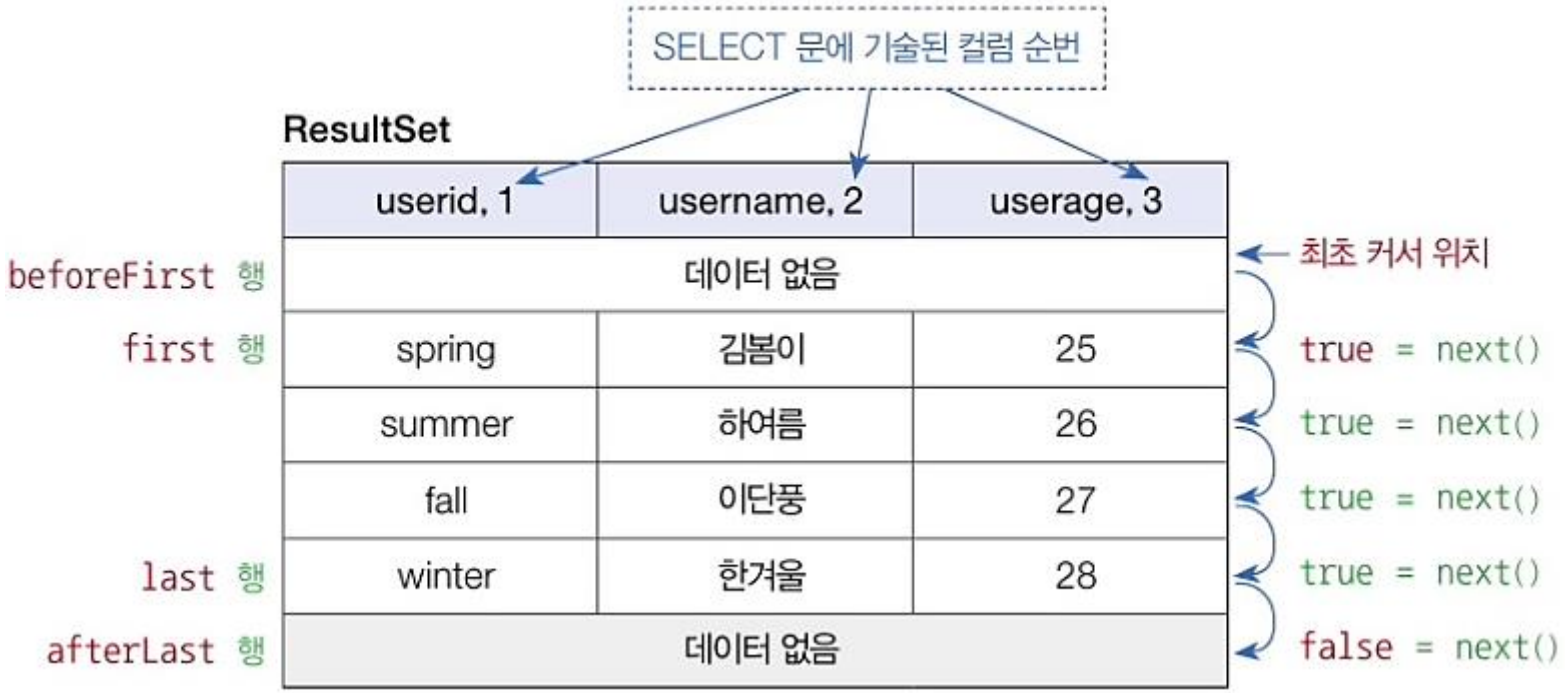
Statement

- SQL 문 실행 클래스
- Connection 객체를 통해 생성

```
Statement stmt = conn.createStatement();
```

- SQL 실행 메서드
  - ResultSet executeQuery(SQL문) : select문 실행
  - int executeUpdate(SQL문): insert, update, delete 문 실행

ResultSet



- 컬럼 값 추출
  - getXxxx("컬러명)
    - Xxx : 추출하고자하는 데이터 타입명
    - getString(), getInt(), getLong(), getDouble()

SelectUserTest.java

```
package org.example;
...
public class SelectUserTest {
  public static void main(String[] args) {

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

      String sql ="select * from users";
      Statement stmt = conn.createStatement();

      ResultSet rs = stmt.executeQuery(sql);
      while(rs.next()) {
        System.out.println(rs.getString("name"));
      }
      stmt.close();
      rs.close();
    } catch (SQLException e) {
      e.printStackTrace();
    }
  }
}
```

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

Statement로 Insert 문 실행하기

```
String sql ="INSERT INTO USERS(ID, PASSWORD, NAME, ROLE)" +
                    "VALUES('member2', 'member123', '일반회원', 'USER')";

int count = stmt.executeUpdate(sql);
```

- 값을 변수로 대체한다면 ?

```
String userId = "member2";
String password = "member123";
String name = "일반회원";
String role = "USER";

String sql ="INSERT INTO USERS(ID, PASSWORD, NAME, ROLE)" +
                    "VALUES('" + userId + "', '" + password + "', '" +
               name + "', '" + role + "')";
```

→ PreparementStatement로 처리

PreparedStatement

- SQL문에 값을 넣을 때 파라미터화 해서 처리

```
String sql ="INSERT INTO USERS(ID, PASSWORD, NAME, ROLE) " +
                    "VALUES(?, ?, ?, ?)";
```

- Connection 객체를 통해 생성

```
PreparedStatement pstmt = conn.prepareStatement(sql);
```

- 파라미터 설정
  - pstmt.setXxxx(파라미터번호, 값)
    - setString(), setInt(), setLong(), setDouble()

- SQL문 실행

```
int count = pstmt.executeUpdate();
```

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!
```

VO 패턴

VO 객체

- Value Object
- 특정 테이블의 한 행을 맵핑하는 클래스

클래스 정의 → 테이블
필드들 → 컬럼들
인스턴스 → 한 행

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;

}
```

DAO 패턴 적용

DAO 클래스

- Data Access Object
- 데이터베이스에 접근하여 실질적인 데이터베이스 연동 작업을 담당하는 클래스

- 테이블에 대한 CRUD 연산을 처리

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();
    }
  }
}
```

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();
  }
}
```